Particle Methods for Geophysical Flow on the Sphere

by

Peter A. Bosler

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Applied and Interdisciplinary Mathematics)
in the University of Michigan
2013

Doctoral Committee:

Associate Professor Christiane Jablonowski, Co-Chair
Professor Robert Krasny, Co-Chair
Assistant Professor Brian K. Arbic
Professor John P. Boyd
Professor Divakar Viswanath

For Jerry and Adele.
I literally would not be here without them.

ACKNOWLEDGEMENTS

A project like this one cannot be completed alone; I am very fortunate to have had lots of support while completing my thesis.

I am especially grateful to my research advisors. Robert Krasny and Reza Malek-Madani have inspired me with their work and mentorship over the years that I have known them. Having the opportunity to work with mathematicians of their caliber has truly been an honor. Christiane Jablonowski showed me how to think like a modeler, as well as a forecaster, and kept our mathematics grounded in real-world applications. Her support and friendship has been invaluable, and I am also thankful for being included in her atmospheric dynamics research group, which has led to many lasting friendships, in addition to the occasional solution to a research problem.

James Kent provided a critical ear and a companion guitar over the course of my thesis. Aside from good counsel, he also provided the code and computations for the Lin-Rood model included in this work to provide comparison to the methods I have developed.

I would like to thank Brian Arbic, John Boyd, Smadar Karni, and Divakar Viswanath, whose courses I have enjoyed immensely, and for their help as committee members for this thesis. I am also very grateful to Peter Miller and Peter Smereka, for giving me the opportunity to study at Michigan. Sylvia Murphy and Cecilia DeLuca inspired me with their leadership and commitment to good science at NCAR, and conversations with Michael Duda, Kevin Reed, Paul Ullrich, and Jared Whitehead helped get my research unstuck on several occasions.

Lastly, I am forever grateful to my wife, Salika, whose support especially during the last phases of writing, when this thesis quite literally sat at the dinner table with us, helped make its completion possible.

TABLE OF CONTENTS

DEDICATION . . . . **ii**

ACKNOWLEDGEMENTS . . . . **iii**

LIST OF FIGURES . . . . **vii**

LIST OF TABLES . . . . **xi**

ABSTRACT . . . . **xii**

CHAPTER

I. Introduction 1

1.1 Background 4
1.2 Modeling the ocean and atmosphere : Dynamical core development 6
1.3 Lagrangian vs. Eulerian methods 14
1.4 Vortex methods 20
1.5 Spherical Geometry 22
1.5.1 Differential operators 25
1.5.2 Distance and area calculations 29
1.5.3 Measuring error 31

II. The Flow Map 33

2.1 Definition of the flow map 33
2.2 Discretizing the flow map 34
2.2.1 Computational implementation and data structures 40
2.2.2 Delaunay triangulations and Voronoi diagrams 49
2.3 Mesh distortion and adaptive refinement 52
2.4 Remeshing procedures 58
2.5 Dynamical core algorithms 64
2.6 Sources of error 67

III. Barotropic Vorticity Equation 71

3.1 Background 72
3.2 Inertial vs. rotating reference frame 73
3.3 Barotropic fluids 75
3.4 Lagrangian form 77
3.4.1 Inertial frame 80
3.4.2 Rotating frame 81
3.5 Numerical method 82
3.5.1 Interpretation 83

3.6 For comparison: An Eulerian model 85

IV. Barotropic Vorticity Equation: Test Cases and Results 89

4.1 Solid body rotation 90
4.2 Rossby-Haurwitz waves 95
4.3 Gaussian vortices 104
4.4 Jet dynamics 116
4.4.1 Steady zonal jet 118
4.4.2 Perturbed zonal jet 124
4.5 Stratospheric model of breaking Rossby waves 129

V. Advection Equation 142

5.1 Lagrangian form 144
5.2 Test cases and results 144

VI. Shallow Water Equations 158

6.1 Shallow water vorticity equation 160
6.2 Shallow water divergence equation 161
6.3 Lagrangian form 162
6.4 Proposed numerical methods 164

VII. Conclusions and Ongoing Work 168

7.1 Summary 168
7.2 Additional test cases 169
7.2.1 Merging vortices 169
7.2.2 Modeling the upper troposphere 173
7.2.3 Tripole vortices 177
7.3 Next steps 178
7.3.1 Improved quadrature 178
7.3.2 Lagrangian remeshing to intermediate times 180
7.3.3 Treecode algorithms 183
7.4 Future research 184
7.4.1 Shallow water test cases 184
7.4.2 Voronoi diagrams and Delaunay triangulations 185

APPENDIX **187**

BIBLIOGRAPHY **200**

LIST OF FIGURES

Figure

1.1 Hierarchy of equations for global circulation models 8

1.2 Development of Hurricane Sandy 12

1.3 “Super storm” Sandy 13

1.4 Locations of DART$^{®}$ buoys 15

1.5 Locations of Argo Floats 16

1.6 Spherical arc length calculation 29

1.7 Spherical area calculation 31

2.1 Particle-panel discretizations of the sphere 40

2.2 Dividing an edge 42

2.3 Initial data structures for icosahedral triangulations 44

2.4 Initial data structures for the cubed sphere 44

2.5 Refining a triangular panel 45

2.6 Refining a quadrilateral panel 46

2.7 Point query: Tree search failure 46

2.8 Point location query algorithm 47

2.9 Timing point location queries 49

2.10 Voronoi diagram of active particles 50

2.11 Refining a stretched panel 55

2.12 Refining a sheared panel 56

2.13 Asymmetry caused by panel shear 56

2.14 Mesh distortion due to a Rossby-Haurwitz wave 57

2.15 Saddle point in a Rossby-Haurwitz wave 59

2.16 Voronoi mesh distortion near a saddle point 59

2.17 Physical example of a saddle point: Extratropical cyclones near Iceland 60

2.18 Lagrangian remeshing 62

2.19 Edge approximation error 69

3.1 5 point stencil 86

4.1 Solid body rotation: position error vs. time 92

4.2 BVE solver Δt error convergence 93

4.3 Particle distance from sphere vs. time step 94

4.4 RH4: relative vorticity 98

4.5 RH4: relative vorticity error, no remesh 99

4.6 RH4, no remesh: Vorticity error, kinetic energy, and enstrophy 100

4.7 RH4, direct and Lagrangian remeshing: Vorticity error, kinetic energy, and enstrophy . . . 101

4.8 RH4: direct and Lagrangian remeshing: Vorticity error vs. N 102

4.9 RH4, direct and Lagrangian remeshing: Kinetic energy and enstrophy comparison with Lin-Rood 105

4.10 Gaussian vortex: Relative vorticity 106

4.11 Gaussian vortex: Comparison of Lin-Rood BVE solver to LPPM 107

4.12 Gaussian vortex: Loss of resolution 109

4.13 Gaussian vortex, $N = 81920$, Lagrangian remeshing: Evolution of relative vorticity and initial latitude 110

4.14 Gaussian vortex, $N = 81920$, Lagrangian remeshing: Evolution of relative vorticity and initial latitude (cont'd.) 111

4.15 Gaussian vortex: Total kinetic energy and enstrophy vs. time 112

4.16 Gaussian vortex, adaptive Lagrangian remeshing : Early time evolution of relative vorticity and initial latitude 114

4.17 Gaussian vortex, adaptive Lagrangian remeshing: Later time evolution of a relative vorticity and initial latitude 115

4.18 Gaussian vortex: Comparison of adaptive vs. uniform mesh at $t = 2.5$ 116

4.19 Velocity and vorticity of a steady zonal jet 118

4.20 Velocity and vorticity of a perturbed zonal jet 119

4.21 Steady zonal jet, $N = 24576$, Lagrangian remeshing: Relative vorticity at $t = 0, 1, 2$ 120

4.22 Steady zonal jet, $N = 24576$, Lagrangian remeshing: Total kinetic energy and total enstrophy 121

4.23 Steady zonal jet: Stretching of Lagrangian parameter 122

4.24 Steady zonal jet, $N = 10242$ Voronoi panels, no remesh: Relative vorticity time evolution . 125

4.25 Steady zonal jet, $N = 10242$ Voronoi panels, Lagrangian remeshing: Rel. vorticity at $t = 1$ 126

4.26 Zonal jet with perturbation wave number 4, quadrilateral panels and adaptive remeshing . . 128

4.27 Zonal jet with perturbation wave number 12, quadrilateral panels and adaptive remeshing . 130

4.28 Stratosphere model forcing functions 134

4.29 Stratosphere model: Breaking Rossby forcing function 134

4.30 Stratosphere model: Absolute and potential vorticity 137

4.31 Stratosphere model: relative vorticity 138

4.32 Stratosphere model: Adaptive meshes 139

4.33 Observed potential vorticity from a southern hemisphere sudden stratospheric warming . . 140

4.34 Observed vorticity from a northern hemisphere sudden stratospheric warming 141

5.1 Tracer distributions for deformational flow tests 145

5.2 Advection test case 1: Gaussian hills and non-divergent wind 146

5.3 Advection test case 1: Error convergence with respect to Δt 147

5.4 Advection test case 1: Relative change in tracer mass vs. N 149

5.5 Advection test case 1: Error comparison for panels with increased edge resolution 149

5.6 Advection test case 1: Comparison of Lagrangian remeshing with piecewise linear vs. cubic interpolation 150

5.7 Advection test case 1: l_2 and l_∞ error 151

5.8 Advection test case 1: Comparison of direct and Lagrangian remeshing 152

5.9 Discontinuous tracer advection: Comparison of direct and Lagrangian remeshing 154

5.10 Discontinuous tracer advection: Comparison of LPPM and Lin-Rood 155

5.11 Filament diagnostics for advection scheme 156

7.1 Freely decaying turbulence: initial condition 171

7.2 Freely decaying turbulence: Lin-Rood solution at $t = 2$ 172

7.3 Freely decaying turbulence: LPPM solution at $t = 1$ 173

7.4 Freely decaying turbulence: LPPM adaptive solution at $t = 1$ 174

7.5 Freely decaying turbulence: Plot of adaptive mesh at $t = 1$ 174

7.6 Annual-mean zonal-mean zonal wind profile 175

7.7 Model profiles of 300 mb zonal mean wind and relative vorticity 176

7.8 Midpoint rule and piecewise linear quadrature 180

A.1 Smooth approximation of a delta function on $\mathcal{S}^2$. 191

A.2 δ-convolution approximation error vs. N 192

A.3 δ-convolution approximation error vs. smoothing parameter 193

A.4 Barycentric linear interpolation on a panel subtriangle 196

LIST OF TABLES

Table

1.1 Schematic of Eulerian and Lagrangian representations 17

1.2 Advantages and disadvantages of Eulerian and Lagrangian numerical methods 17

2.1 Mesh size vs. *N* for quadrilateral and triangular panels 39

2.2 Passive particles data structure 43

2.3 Edges data structure 43

2.4 Panels data structure 43

4.1 Test case list 89

4.2 Parallel speedup and efficiency for BVE solver 95

4.3 RH4: Error norm comparison with Eulerian method 104

5.1 Advection test case 1: Error convergence rates 152

ABSTRACT

Particle methods for geophysical flow on the sphere

by
Peter A. Bosler

Co-Chairs: Jablonowski & Krasny

We present a Lagrangian Particle-Panel Method (LPPM) for geophysical fluid flow on a rotating sphere motivated by problems in atmosphere and ocean dynamics. We focus here on the barotropic vorticity equation and 2D passive scalar advection, as a step towards the development of a new dynamical core for global circulation models.

The LPPM method employs the Lagrangian form of the equations of motion. The flow map is discretized as a set of Lagrangian particles and panels. Particle velocity is computed by applying a midpoint rule/point vortex approximation to the Biot-Savart integral with quadrature weights determined by the panel areas. We consider several discretizations of the sphere including the cubed sphere mesh, icosahedral triangles, and spherical Voronoi tesselations. The ordinary differential equations for particle motion are integrated by the fourth order Runge-Kutta method. Mesh distortion is addressed using a combination of adaptive mesh refinement (AMR) and a new Lagrangian remeshing procedure. In contrast with Eulerian schemes, the LPPM method avoids explicit discretization of the advective derivative. In the case of passive scalar advection, LPPM preserves tracer ranges and both linear and nonlinear tracer correlations exactly.

We present results for the barotropic vorticity equation applied to several test cases including solid body rotation, Rossby-Haurwitz waves, Gaussian vortices, jet streams, and a model for the breakdown of the polar vortex during sudden stratospheric warming events. The combination of AMR and remeshing enables the LPPM scheme to efficiently resolve thin fronts and filaments that develop in the vorticity distribution. We validate the accuracy of LPPM by comparing with results obtained using the Eulerian based Lin-Rood advection scheme. We examine how energy and enstrophy conservation in the LPPM scheme are affected by the time step and spatial discretization. We conclude with a discussion of how the method may be extended to the shallow water equations.

CHAPTER I

Introduction

This thesis adapts the mathematical and numerical technique known as vortex methods to solve problems of geophysical fluid dynamics (fluid flow of the atmosphere and ocean). We represent the Earth as a sphere, and solve the equations of motion on the spherical surface as a model of the global circulation. From a weather and climate modeling perspective, we take the first steps toward using vortex methods as the basis of a *dynamical core*.

A dynamical core is the foundation of every weather and climate model. It encompasses the model equations that must be solved, their related spatial and temporal discretizations, and the numerical method employed to solve the equations [108]. In proposing vortex methods as an approach for discretizing and solving model equations, we develop an algorithm with four key features that could provide a new dynamical core to the weather and climate modeling communities.

The first and most notable feature of our method is that it is based on the Lagrangian form of the equations of motion. This implies that the equations are solved in a reference frame that is moving with the flow, or "flow-following." Numerically this means that our computational elements move as computations evolve in time. Dynamical cores have traditionally used an Eulerian form for the fluid equations, which solves the equations from the point of view of a stationary reference frame. Lagrangian formulations are thought to require less numerical viscosity [83], may be better at maintaining sharp interfaces such as frontal systems [97], and may have better conservation properties than their Eulerian counterparts [114]. We investigate these claims for geophysical applications throughout this work. Lagrangian methods must contend with the fact that, by definition, the arrangement of their computational elements will deform and distort over time which can lead to a loss of accuracy.

We address the challenge of distortion with the second key feature of our method: a combination of adaptive refinement and a new remeshing procedure. Adaptive refinement has been an area of active research in geophysical modeling in recent years [15]. This research is driven by the fact that models are solved by computers with finite resources. An adaptive refinement strategy that uses more resources in regions of the fluid that are most computationally challenging, while saving resources where they are not needed can make efficient use of expensive computer time. For a Lagrangian method like the one proposed by this thesis, adaptive refinement also helps to preserve accuracy [101]. As particles move, some regions of the domain may become depleted of computational elements; adaptively refining these regions as they occur can prevent computational accuracy from degrading. The new remeshing strategy we develop to maintain the Lagrangian flow map and preserve many of the beneficial properties of the Lagrangian representation of the fluid equations works in conjunction with our adaptive refinement procedures.

The third key feature of our method is that velocity is computed from vorticity with the Biot-Savart law [38, 30, 120]. From a mathematical perspective, this means that we solve the partial differential equations of motion with Green's function, rather than using spectral transforms, finite element methods, finite difference methods, or finite volume methods. Such an approach is new to the dynamical core community. The combination of a Lagrangian formulation and the Biot-Savart law is responsible for the success of vortex methods. It enables the spatial discretization to provide a quadrature rule instead of discrete derivatives, and is more tolerant to non-uniform resolutions as a result.

Lastly, we represent the sphere in 3D Cartesian coordinates, rather than 2D spherical coordinates. Cartesian coordinates do not have the polar singularities of the spherical coordinate system [188], and Green's function is simpler to evaluate in Cartesian coordinates [136].

The rest of this introductory chapter is devoted to background. First the background of numerical weather prediction and its relation to mathematics and scientific computation is presented in section 1.1. We discuss decisions modelers must make during the beginning stages of dynamical core development, including fundamental decisions like using the Lagrangian or Eulerian representation of the fluid equations in section 1.2. Section 1.3 focuses on the primary differences between the Lagrangian and Eulerian formulations, which are introduced from both physical and mathematical perspectives. The basis of our numerical techniques are vortex methods, which are introduced in section 1.4. The last section of this chapter reviews spherical geometry and the coordinate systems we use throughout this thesis.

The remainder of the thesis is divided into six chapters and an appendix. Chapter II introduces the flow map, which is the basis of all Lagrangian methods, and the specific algorithms we use to discretize it. Chapter II develops the key features of our algorithm, and distinguishes our approach from current dynamical cores and related works in vortex methods. We combine the flow map and the barotropic vorticity equation to derive our numerical method in chapter III. Chapter IV presents several test cases for barotropic vorticity problems and numerical solutions produced by our model as well as results from methods derived from an operational Eulerian dynamical core. The method we develop in chapters II and III can also be applied to the advection equation, which is done in chapter V for a suite of deformational flow test problems. The shallow water equations are more complex than the barotropic vorticity equation, and are frequently used as the proving ground for developing dynamical cores [187]. Chapter VI derives the Lagrangian form of the shallow water equations and introduces the strategies we intend to apply to that equation set. We present conclusions from our investigation of vortex methods for dynamical core applications in chapter VII, and identify several strategies for improving our method. The appendix provides a brief introduction to the problem of interpolating scattered data, which we confront in our remeshing procedures.

1.1 Background

Like most of fluid dynamics, the mathematics that underlies modern weather and climate models owes a great deal to the theorems of the late nineteenth century due to Kelvin and Helmholtz; Batchelor [12] provides a modern reference. Vilhem Bjerknes' 1898 paper adapted the fundamental results of Kelvin and Helmholtz to the specific application of geophysical fluid dynamics by considering variable density, rotating fluids. He used his primary result, now known as Bjerknes' Circulation Theorem, to explain land/sea breezes, cyclones, and monsoons [174]. In 1904 another paper by Bjerknes posed the forecasting problem as the solution of a set of nonlinear partial differential equations that could be calculated [18]. This paper is credited as the beginning of numerical meteorology [134].

Lewis Fry Richardson was the first to make a serious attempt at Bjerknes' goal [149, 134]. Richardson estimated that a workforce of 64,000 (human) computers, each working simultaneously in parallel could create a daily global forecast [149, 80]. Richardson's ideas were limited by poor observational data to serve as initial conditions and a set of equations that allowed noisy features such as high-speed gravity waves to undermine the accuracy of his numerical methods [80]. Today many are astounded at the similarities between Richardson's vision of human computers and the parallel supercomputers that run weather and climate models. At the time, however, the poor accuracy of his result and the fact that it took him 6 weeks to compute a 6 hour forecast stalled the development of numerical weather models [134].

The decades following Richardson's work, roughly 1915-1945, included the development of aircraft, two world wars with battles whose outcomes were significantly affected by weather, and the development of the first electronic computers. As the world's skies filled with aircraft, meteorologists suddenly had access to a vast source of observational data both from the planes themselves and from the growing number of airfields that launched weather balloons to write their own local forecasts. Governments were eager to minimize casualties due to weather like tropical cyclones and cold, and thus also eager to have accurate forecasts to accompany their operational plans. Computers made it possible to complete calculations with one machine that Richardson imagined would take 64,000 people.

At the same time, Jule Charney provided the mathematical approximations that allowed the filtering of noisy features such as high-speed waves from weather forecasting models' equations [28, 80]. Charney's approximations are now known as quasi-geostrophic theory. Strategically applying simplifying approximations to the equations of motion continues in today's models [177]. The physics and dynamics of global circulations (both atmosphere and ocean), as well as the most useful approximations, are described in the texts of Gill [68], Holton [80], and Vallis [183].

John von Neumann developed the modern computer in the early postwar period and, along with Charney, used it to forecast weather as early as 1950 using the barotropic vorticity equation [29]. Mathematics, meteorology, and computers have been linked ever since, with significant advances in one of these areas frequently having profound impact on the others. For example, in 1963 meteorologist Edward Lorenz discovered the beginnings of chaos theory by examining large differences between computations that began with very similar initial conditions in a model of atmospheric convection [119, 69].

Early models used finite difference schemes and spectral transform methods [80], but finite volume and finite element methods soon followed [111, 22, 108]. Each of these models were based on Eulerian (fixed grid) discretizations. Newer models use different horizontal discretizations of the sphere, to avoid the pole problem associated with latitude-longitude grids [188]. Lagrangian techniques such as vortex methods were relatively isolated from geophysical fluid dynamics applications until recently.

Modern scientists inhabit a world with supercomputers capable of running many forecasts concurrently (a technique known as "ensemble forecasting") and are able to compare the forecasts from a variety of different models that employ different approximations and numerical methods [86, 181]. These inter-comparison projects, and the continuous process of verifying a model's past forecasts with current observations, have led to a vast improvement in forecast accuracy of today's models over the original efforts of Richardson, Charney, and von Neumann. However, these projects have also identified some areas where contemporary models still fall short of desired capabilities. The two areas for potential improvement most relevant to this work, numerical conservation and multi-scale flow features, are discussed in the next section.

1.2 Modeling the ocean and atmosphere : Dynamical core development

As a starting point, a modeler must know what goal their model will help to achieve, and how to efficiently achieve it. For example, if the goal is to better understand local winds, a global simulation is unnecessary. If the goal is to forecast only a short time (7 to

10 days) ahead, the modeler may be able to treat sea surface temperatures as constant (and thereby avoid having to simulate the ocean circulation), since oceanic temperatures vary at a much slower rate than air temperatures. If the goal is to simulate the climate for many decades, these choices are less clear.

Over long periods of time oceanic currents will have significant impact on the atmosphere (and vice versa), and local effects in one area could potentially alter the weather hundreds or thousands of kilometers downstream. In such cases, modelers often choose the dominant features they wish their model to capture with a potential sacrifice in other areas. For example, modelers seeking to investigate the effect of the El Niño Southern Oscillation on precipitation in the Maritime Continent may choose to use a coarse representation of the upper atmosphere, in order to reserve computational resources for the troposphere where precipitation occurs. Each of these decisions leads to a different set of equations for the model regardless of the choice of numerical method.

The choice of equations almost always involves some approximations applied to the full set of Navier-Stokes equations [68, 80, 183]. It is very important to know the consequences of these approximations, to know what fluid flow features they exclude, and the significance of those exclusions relative to the validity of the model's results. It is common, for example, for global circulation models to employ mathematical approximations that prevent models from producing acoustic waves. These waves are prohibitively expensive to compute and have little effect on weather, so little is lost by excluding them. Another common approximation in global models is the hydrostatic approximation, which enforces a balance between the vertical pressure gradient and gravity. This approximation excludes the vertical propagation of acoustic waves, and a modifies the dynamics of gravity waves, which have relatively small impact on motion with scales greater than about 10 km [177]. If the goal of the model is to resolve flow features larger than that scale,

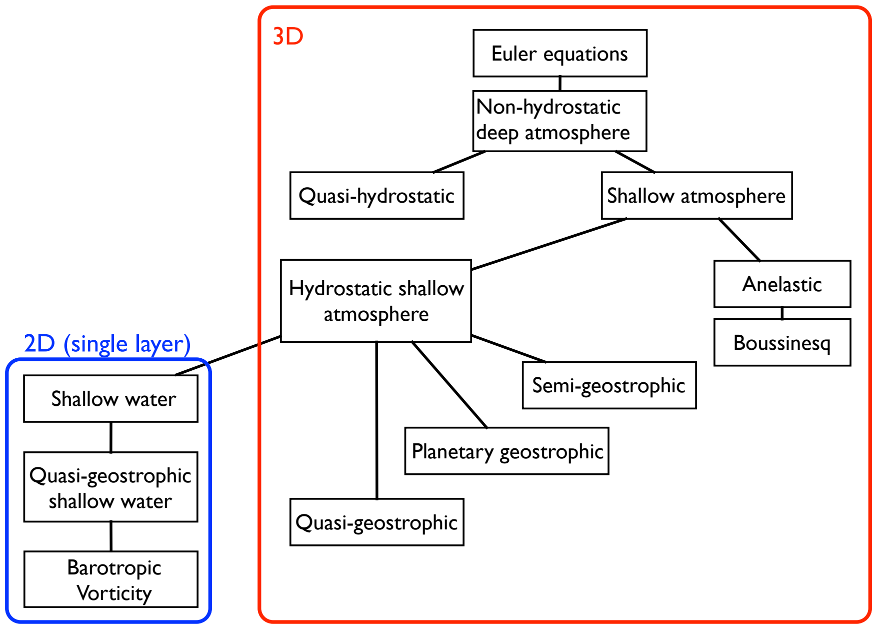


Figure 1.1: Hierarchy of equations for global circulations models, adapted from [186, 177].

this may be acceptable; if not, this approximation is an oversimplification of the equations and it should not be used. Atmospheric and oceanic modelers have developed a hierarchy of equations [186, 177] that concisely illustrates the relative complexity (or simplicity) of the various equations commonly used by dynamical cores. This hierarchy is presented in figure 1.1.

At the top of the model equation hierarchy are the Euler equations, which are the inviscid approximation of the Navier-Stokes equations. Combining the effect of gravity and centrifugal force into one potential function is a common technique that yields geopotential surfaces; treating these slightly ellipsoidal surfaces as though they were spheres results in the spherical geopotential approximation [68, 80, 183]. Applying the spherical geopotential approximation to the Euler equations gives the non-hydrostatic deep-atmosphere equations. In these equation sets, all types of motion including acoustic waves and gravity waves can exist. High-frequency waves require small time steps and high spatial resolution to resolve, but may have little effect on the meteorologic or oceanographic phenomena of interest; neglecting such motions in a physically appropriate manner is the goal of each of the subsequent approximations. A quasi-hydrostatic equation set can be derived by replacing the prognostic vertical momentum equation with a diagnostic balance equation that assumes vertical velocities are much smaller than horizontal velocities. This assumption prevents the vertical propagation of acoustic waves [186]. The shallow atmosphere approximation neglects metric terms from the momentum equation that are justified when the ratio $\mathfrak{R}$ of planetary rotation frequency to the buoyancy frequency is small $\mathfrak{R} \ll 1$, which is typical of Earth's current climate [186]. Applying the shallow atmosphere approximation in conjunction with the hydrostatic approximation removes acoustic waves entirely and results in a hydrostatic shallow atmosphere model, and an equation set also referred to as the primitive equations [177]. Anelastic and Boussinesq approximations also filter out acoustic wave modes, without requiring hydrostatic balance [80, 183]. Quasi-geostrophic approximations use the assumption that horizontal pressure gradients are nearly balanced by the Coriolis force; this assumption is valid for large scale motions and filters out another type of fast wave, inertio-gravity waves, in addition to acoustic waves [177]. The planetary geostrophic and semi-geostrophic approximations rely on assumptions more applicable to the ocean than the atmosphere, where density variations in the horizontal are more significant [183].

The atmosphere may also be approximated as a single, two-dimensional layer of fluid on a sphere. The fluid layer is assumed to have uniform density, and thus acoustic waves are not possible. The shallow water equations arise from this approximation and include 2D gravity waves; neglecting divergence by assuming the fluid also has a uniform depth results in the barotropic vorticity equation and removes gravity waves from the model [183]. Notably absent from the hierarchy presented in figure 1.1 is the advection equation. The advection equation is of such fundamental importance that it applies to each equation set, in both 2D and 3D. Some models treat the advection equation separately, because it must be solved no matter what equation set is used.

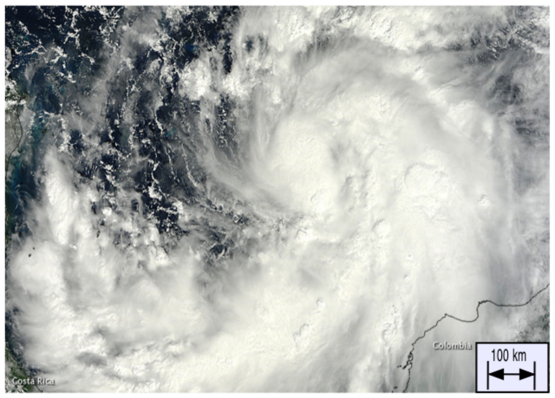
In this thesis our goal is to develop Lagrangian particle methods for global fluid dynamics problems. We examine their accuracy and efficiency with specific focus on multi-scale motion and conservation. For the investigatory purpose of this thesis, we start at the base of the equation hierarchy in figure 1.1, with the barotropic vorticity equation and the 2D advection equation. These equation sets are sufficiently complex for our purposes, but simple enough that fluid motions not relevant to our primary purpose are excluded. Despite their relative simplicity compared to operational forecast model equations, the advection equation and the barotropic vorticity equation are commonly used to investigate numerical methods for dynamical cores, *e.g.* [105, 113, 95]. These equations are also where Charney and von Neumann began their pioneering work [29].

The need for computational conservation is often framed from the point of view of conservation of mass. Recent examples of the need for accurate mass (or tracer) advection schemes in weather models are found in the forecasts of the distribution of ash from the eruption of Iceland's volcano Eyjafjallajökull in 2010 [59] and the forecast of radioactive particle dispersion following the tsunami damage to Japan's Fukushima Daiichi nuclear power plant in 2011 [171]. Climate models advect passive tracers that are used as input to chemistry parameterizations running in conjunction with the dynamical core. Errors caused by the advection scheme can amplify errors in chemical models or even cause them to crash if the advection scheme produces nonphysical negative values [109].

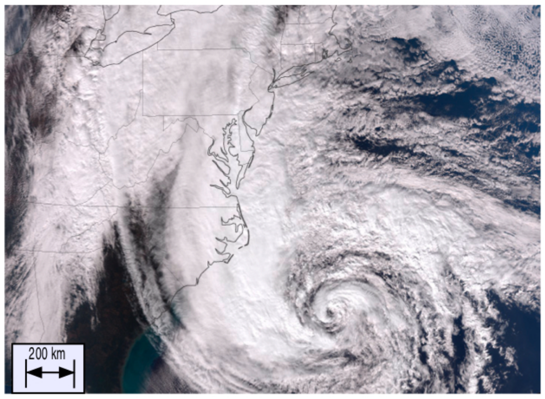
But the need for conservation applies to less obvious quantities than mass, like energy and potential vorticity, that are also conserved in the continuous equations [120]. In fact, energy and potential vorticity are perhaps more important to conserve than mass because these quantities affect the advecting velocity itself. In the barotropic vorticity equation, potential vorticity is equivalent to the absolute vorticity, and we examine its conservation in detail for various test cases in chapter IV.

Thuburn [175, 176] argues that some conserved quantities of the continuous equations should not be conserved in the discrete equations because quantities like enstrophy have a downscale cascade toward the model's unresolved scales. This cascade will create motion that the model is incapable of resolving which will then become a source of error; Thuburn's argument is that numerical dissipation avoids this source of error by preventing the development of motions too small for the model to handle. Numerical dissipation in a model is not necessarily similar to physical dissipation in the environment, however, and it is not clear how to discern which of the many dissipation techniques available to modelers are the best to use [95]. These techniques are the subject of active study in Eulerian dynamical cores, and their effects are documented in [95] and [88]. Analogous numerical dissipation strategies for vortex methods are addressed by Cottet and Koumoutsakos [38, chapter 5] but are beyond the scope of this work. Instead we attempt to resolve even the small scales that develop and document the conservation of enstrophy by our model. We leave the subject of dissipation as an item for future study in chapter VII.

To discuss the challenge that multi-scale motions pose for every dynamical core, we first make the obvious remark that the Earth is big: its mean radius is 6371 km [80]. But atmospheric motions can be significant even if they are relatively small. Consider the recent example of Hurricane Sandy. A satellite image of its beginning stages is shown in figure 1.2(a). At this point it is small in the sense that its diameter is on the order of 100 km and the Earth's circumference at 15N latitude is approximately 38600 km. Five days later, the storm has intensified to hurricane strength but more importantly, it has begun to interact with the trough of a planetary scale wave over the eastern United States, as shown in figure 1.2(b). The scale of the storm itself is bigger, as shown by the scale marker on



(a)



(b)

Figure 1.2: NASA satellite images of Hurricane Sandy. (a) Tropical Storm Sandy on October 23, 2012, off the coast of Colombia in the Caribbean Sea [160]. (b) Hurricane Sandy 575 miles south of New York City, October 28, 2012 [4].

the image, and the beginnings of the interaction with the mid-latitude trough are evidenced by the large cloud shield to the west of the storm. Two days after that, the interaction of the tropical cyclone with the planetary wave trough created "Super Storm Sandy", which affected the majority of eastern North America (figure 1.3).

Hurricane Sandy is an example of nonlinear interactions between features at different scales. In actuality, Hurricane Sandy intensified as a result of thermodynamic motions that are excluded from the barotropic vorticity equation, which has no mention of temperature. In our study of the barotropic vorticity equation in chapters III and IV we will confront situations where vortices of different scales interact with each other. Dynamic interactions of vorticity were also important to the development of Sandy; the midlatitude trough had a vorticity maximum along its axis, and the tropical cyclone was its own vorticity maximum. Despite the simplicity of the barotropic vorticity equation compared to the physical environment, it contains enough dynamics to still be relevant as a model of important atmospheric flow features at varying scales [52, 80, 95].

Current research in dynamical core development is aimed at creating models with variable spatial resolutions to handle the variety of spatial scales in geophysical problems [15].

Figure 1.3: NASA satellite image of "Super Storm" Sandy over eastern North America, October 30, 2012. The point of view looks southward; Canada is at the bottom of the image and Florida can be seen near the top [103].

Some regions of a model's computational space will have very fine resolution, while other regions deemed less important by some criteria will have coarser resolution. This allows modelers to use finite computational resources where they are most needed, but the choice of where to put the higher resolution is not trivial. Some models statically place higher resolution over regions where conditions are most apt to change rapidly, for example, over the Gulf Stream [151]. Others choose regions where accuracy is most crucial, for example, over the continental United States or Europe which have high concentrations of airports [162].

Another approach is to let the flow dictate, dynamically, where resolution is needed by having the grid adapt to regions where the flow is changing rapidly. This technique is known as Adaptive Mesh Refinement (AMR) [16, 173, 15, 87]. We use the dynamic, or adaptive, approach in this study. The difference between these two approaches, one static, one dynamic, are analogous to the differences between the Eulerian form and the Lagrangian form of the equations of motion, discussed in the next section.

1.3 Lagrangian vs. Eulerian methods

Eulerian methods are the predominant numerical technique used in dynamical cores today [80, 108]. They use a fixed grid for the spatial discretization and numerical methods such as finite differences, finite volume methods, spectral transforms, and finite element methods to solve the fluid equations. An Eulerian model measures and evolves the fluid at its fixed grid points. It is analogous to measuring the flow of a river or basin with an array of anchored buoys. These buoys would measure the properties of the fluid, such as its temperature, pressure and velocity, etc., as it flows past their locations. In this way Eulerian methods can be described as modeling the fluid as it flows over or past a fixed mesh. A modeler could then build a picture of the fluid's behavior from the discrete measurements of each individual buoy, using the knowledge of each buoy's fixed (anchored) position to give the spatial dependence of the measured quantities. A physical example of such a system are the buoys of the Deep-Ocean Assessment and Reporting of Tsunamis (DART$^{®}$) project [70] (see figure 1.4).

To continue the buoy analogy, a Lagrangian model would correspond to an array of drifting buoys. The spatial discretization is known at $t = 0$, but is allowed to evolve with the fluid velocity so that the positions of each buoy at $t > 0$ depend on the flow. Velocity is determined by measuring the changes in the position of the buoys themselves. The drifting buoys still measure state variables of the fluid, but since they are moving with the fluid, the buoys' meters record no change in variables that are conserved materially. Each Lagrangian buoy can be thought of as a tag for a particular parcel of fluid which can be identified by its invariant properties as it moves with the current. Lagrangian models are often described as flow-following because of this analogy. A physical example is the Argo project [170], which monitors thousands of drifting floats across the Earth's oceans (figure

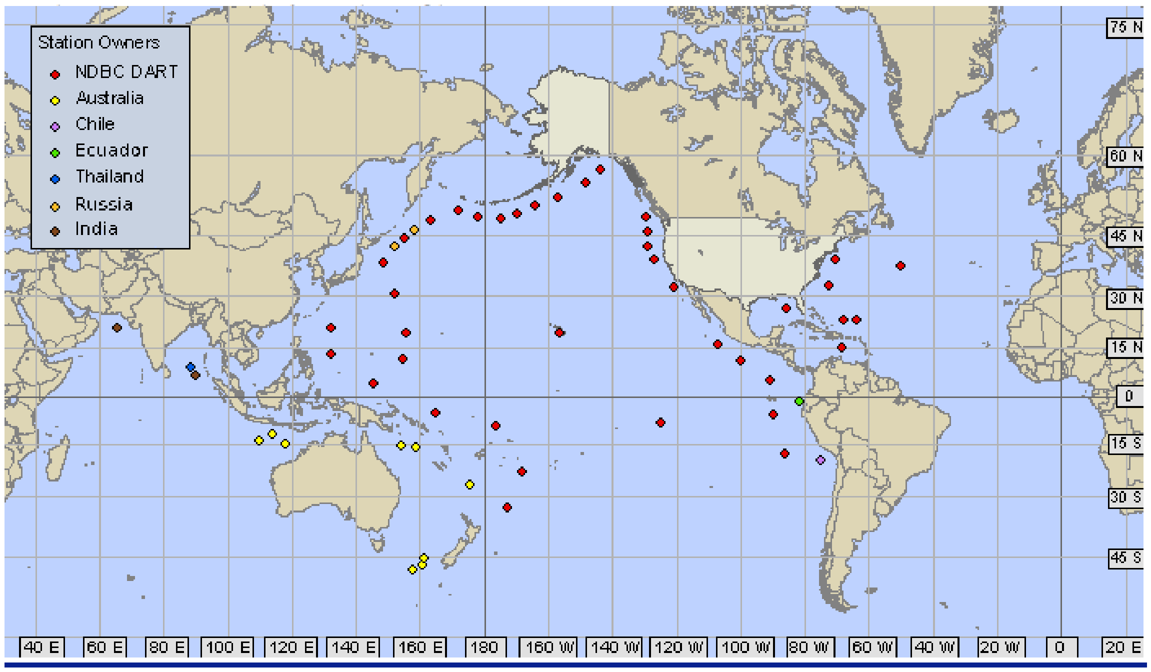


Figure 1.4: The locations of DART® buoys used for tsunami monitoring [70].

1.5).

In the computational space of a model, buoys are replaced by locations where the fluid's properties are defined – the computational elements of the model's spatial discretization. These are grid points in an Eulerian model, and particles in a Lagrangian model.

Let the vector $\boldsymbol{x}(t)$ represent the position of the computational elements of the model fluid. Let the vector $\boldsymbol{\Phi}(t)$ represent the state variables of the fluid, such as its vorticity, for example. Knowledge of both $\boldsymbol{\Phi}$ and $\boldsymbol{x}$ are required to solve the equations of motion. At any one time step $t = t_0$ in a calculation, both $\boldsymbol{\Phi}$ and $\boldsymbol{x}$ are known. A dynamical core's task is to use that information to evolve the model fluid to the next time step, $t = t_0 + \Delta t$.

To keep things simple we imagine a fluid whose state variables are preserved exactly in motion following the flow (this is precisely the case with the barotropic vorticity equation). Then these quantities (which comprise the vector $\boldsymbol{\Phi}$) are "stationary" in the Lagrangian frame so that $\boldsymbol{\Phi}(t_0+\Delta t) = \boldsymbol{\Phi}(t_0)$, but the position of the computational elements at $t = t_0+\Delta t$

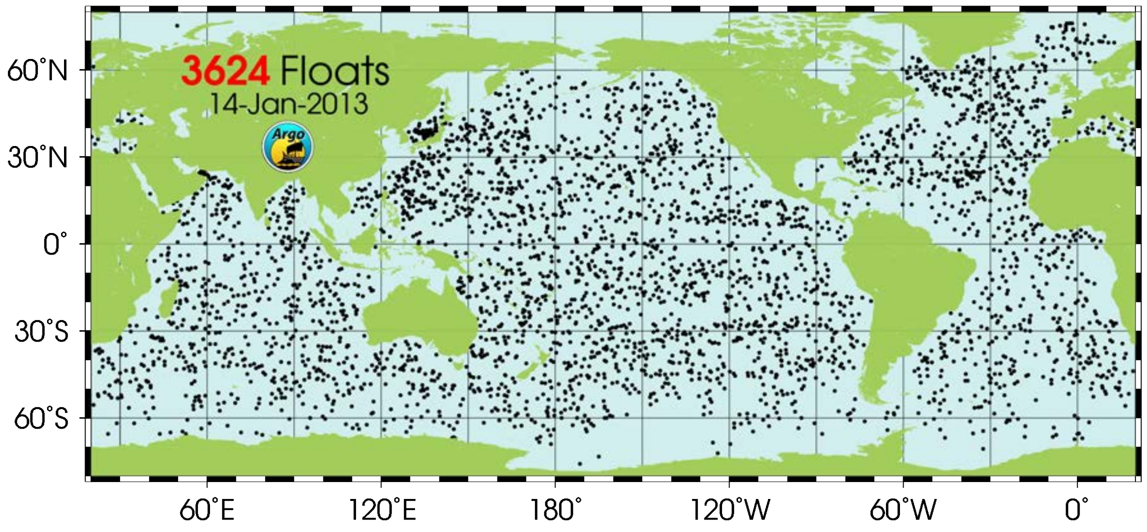


Figure 1.5: Global locations of 3264 Argo Float drifting buoys on Janurary 14, 2013 [170].

must be determined,

$$\boldsymbol{x}(t_0 + \Delta t) = \mathcal{L}_L\left(\boldsymbol{x}(t_0), \boldsymbol{\Phi}(t_0 + \Delta t)\right). \tag{1.1}$$

By contrast, the computational elements $\boldsymbol{x}$ are stationary in an Eulerian frame, *i.e.,* $\boldsymbol{x}(t_0 + \Delta t) = \boldsymbol{x}(t_0)$, and the fluid's physical quantities $\boldsymbol{\Phi}$ must be computed,

$$\boldsymbol{\Phi}(t_0 + \Delta t) = \mathcal{L}_E\left(\boldsymbol{x}(t_0 + \Delta t), \boldsymbol{\Phi}(t_0)\right). \tag{1.2}$$

where $\mathcal{L}_L$ and $\mathcal{L}_E$ represent the differential operators appropriate to the model equations in their Lagrangian and Eulerian forms, respectively.

This conceptual model is summarized in table 1.1. Although this idea holds for the case of the barotropic vorticity equation, in the more complex shallow water equations there are several terms that are not stationary in either representation. This discussion is therefore not meant as an absolute principle but rather as an illustration of the fundamental differences between the Eulerian and Lagrangian forms of the fluid equations.

Eulerian and semi-Lagrangian methods have been successful for many years, and are used in all operational weather and climate models today [80]. Lagrangian methods are

Eulerian	t_0	$t_0 + \Delta t$
$\boldsymbol{\Phi}$	known	computed
$\boldsymbol{x}$	known	known

Lagrangian	t_0	$t_0 + \Delta t$
$\boldsymbol{\Phi}$	known	known
$\boldsymbol{x}$	known	computed

Table 1.1: Conceptual schematic of Eulerian and Lagrangian solution techniques. $\boldsymbol{\Phi}$ represents the fluid's state vector. $\boldsymbol{x}$ represents the positions of the computational elements.

Form	Advantages	Disadvantages
Eulerian	Straightforward	Advective derivative must be calculated
Lagrangian	Implicit advection	Distortion of fluid particles

Table 1.2: Advantages and disadvantages of Eulerian and Lagrangian numerical methods.

relatively rare by comparison, due in large part to the difficulty of implementing them in a computationally robust manner across the whole globe [80, 150]. This thesis addresses those difficulties by leveraging techniques from the field of vortex methods [30, 38, 120] to solve for fluid velocity and developing an adaptive refinement strategy and a new remeshing algorithm that preserves the beneficial properties of Lagrangian methods.

Existence and uniqueness theorems guarantee that Lagrangian and Eulerian formulations are mathematically equivalent [12, 30], but computational implementations of them are very different. There are practical advantages and disadvantages to both forms of the equations (table 1.2).

All dynamical cores are dependent upon the supercomputers that run their software. Eulerian methods take advantage of their fixed spatial discretizations by predetermining memory layouts within a computer. Additionally, because of their fixed spatial discretizations, discrete versions of spatial derivatives are much more straightforward to compute in Eulerian methods than in Lagrangian methods.

Unfortunately Eulerian schemes are subject to truncation error in the fundamentally important advective derivative. This derivative is part of all model equations, including the advection equation. The advection equation states that a materially invariant quantity, say $Q(\boldsymbol{x}, t)$, is conserved in the absence of sources or sinks.

$$\frac{d}{dt}Q(\boldsymbol{x}, t) = \frac{\partial}{\partial t}Q(\boldsymbol{x}, t) + \boldsymbol{u} \cdot \nabla Q(\boldsymbol{x}, t) = 0 \tag{1.3}$$

In modern models, Q may represent dynamic quantities like potential vorticity and thermodynamic variables, suspended particles (such as pollutants or volcanic ash), or carbon species used by chemistry parameterizations within a climate model. From a stationary (Eulerian) reference frame, equation (1.3) states that the partial time derivative $\partial Q/\partial t$ is balanced by the advective derivative, $\boldsymbol{u} \cdot \nabla Q$.

The advective derivative is a nonlinear operator that causes many Eulerian models to have difficulty achieving both high accuracy and conservation within their computations [109]. Discrete versions of the advective derivative may introduce high frequency numerical noise into a model's solutions, which must be filtered out in order for Eulerian models to remain stable. Typically, this is done with dissipative numerical viscosity or hyper-viscosity [88], but these numerical necessities may bear little resemblance to the physically dissipative processes that operate in the environment [95].

In a Lagrangian method fluid particles are advected by the flow so there is no need to discretize the advective derivative explicitly. The total derivative in (1.3), $d/dt = \partial/\partial t + \boldsymbol{u} \cdot \nabla$, becomes an ordinary time derivative. The advection equation in a Lagrangian reference frame simply states that Q is constant following the flow: $dQ/dt = 0$. By avoiding the advective derivative, Lagrangian methods can use less artificial numerical viscosity than their Eulerian counterparts [114]. Lagrangian models may have better conservation properties than Eulerian models as a consequence. In this thesis, for example, we avoid the use of numerical diffusion altogether.

Some model developers seek to incorporate the advantages of both techniques by combining aspects of each into *semi-Lagrangian* methods that interpolate Lagrangian particle data to and from an Eulerian grid at each time step [166, 132, 104, 107]. Fully Lagrangian methods have been avoided in the geophysical fluid dynamics community because collections of moving fluid particles develop regions of non-uniform spatial resolution as computations evolve in time [80, 150]. However, Eulerian and semi-Lagrangian techniques do not yet satisfy all of the accuracy and conservation requirements that modelers desire in a dynamical core [109]. A Lagrangian method that is able to overcome the challenge of non-uniform spatial resolution would therefore be a significant contribution to the atmospheric modeling community.

Alam and Lin [2] use a Lagrangian particle method to solve a two-dimensional (vertical cross section) model of a land-sea breeze system. They propose that Lagrangian models may not suffer as much from spurious oscillations or physically unrealistic negative concentrations, which can occur in Eulerian advection schemes. Based on their results, which show similar accuracy as semi-Lagrangian and Eulerian models with smaller computational cost for their test problem, they advocate the development of a fully Lagrangian atmosphere model.

Mesinger [130] attempted such a model in 1971, solving the shallow water equations on the sphere, but was unable to overcome errors caused by the irregular distribution of particles as his computations evolved [6]. In 1985 Augenbaum [6, 7] solved the shallow water equations on the sphere using a moving Voronoi mesh. Augenbaum used the properties of Voronoi diagrams to give an optimal discretization of irregularly spaced particles, and a local remeshing algorithm to handle particles that moved too close to each other. The accuracy of Augenbaum's study was limited by low-order spatial derivative calculations that were less accurate than competing Eulerian methods [6].

Although they each attempted a Lagrangian method, neither Mesinger nor Augenbaum used vorticity as a prognostic variable. Vortex methods, which provide the basis of this work, are based on the vorticity-stream form of the fluid equations, and rely on an integral formulation of velocity rather than a differential form. It is possible, therefore, to achieve higher-order accuracy by improving the quadrature scheme of the discretization [5, 14], which may prove to be simpler than calculating high order derivatives on collections of irregularly spaced fluid particles.

1.4 Vortex methods

Vortex methods are summarized in the texts of Cottet & Koumoutsakos [38], and Majda & Bertozzi [120]. Leonard [110] and Koumoutsakos [97] provide review articles. Conceptually vortex methods rely on the idea of vortices as material fluid entities; each vortex moves in a velocity field induced by the other vortices. Likewise, each vortex contributes to the motion of the others with its own vorticity. Fujiwhara's experiments [63] in the 1920's approached this idea from a meteorological perspective.

Mathematically, vortex methods are founded upon the fluid dynamics theorems of Kelvin and Helmholtz [38, 30, 120], like the model equations discussed in section 1.1. As a technique for solving fluid dynamics problems, they were first studied by Rosenhead in the 1930s [154, 38]. They are based on the Helmholtz decomposition of the fluid velocity and consequences of Kelvin's circulation theorem which will be discussed in more detail in chapter III.

Vortex methods represent the fluid's vorticity field with a collection of singular or regularized point vortices [145, 97]. These point vortices are the fluid particles in a vortex method computation, and each particle is affected by interactions with every other point vortex through the singular or regularized integral kernel of the Biot-Savart law [120]. Vortex methods are an attractive alternative to contemporary fluid dynamics solvers because of their Lagrangian formulation. Conservation of fundamental fluid properties like circulation and vorticity is built into their algorithms [38, 120]. However, their singular integral kernels caused skepticism about their numerical accuracy through the 1970s.

Vortex blobs (smooth approximations of point vortices) were introduced by Chorin in 1973 [31, 32] as a means of smoothing the singular integral kernels and restoring accuracy. Analysis of numerical convergence of vortex blob methods to solutions of the Euler equations were provided by Hald [74] in 1979 and Beale & Majda [14] in 1985 for smooth vorticity fields. These studies quantitatively related the magnitude of smoothing required to the initial particle spacing. A small study of this relationship is given in the appendix, section A.1. The convergence studies from Hald and Beale & Majda provided a theoretical foundation for the analysis of vortex blob methods [38].

Perlman [142] performed a numerical study in 1985 that verified the results predicted by Hald and Beale & Majda for three test problems in planar Cartesian geometry without boundaries. Krasny [99] provided a numerical study of vortex blob methods for vortex sheets in 1986 that showed the methods' convergence as both inter-particle spacing converged to zero and as the smoothing parameter converged to zero. These numerical results validated the practical use of the vortex blob approximation.

While vortex blobs gained acceptance, some controversy still surrounded the use of point vortices. Leonard [110] reported the use of point vortices in some studies even though convergence proofs only existed for vortex blobs [71]. In particular, Christiansen [33] used point vortices for hydrodynamics problems similar to the problems we consider in this thesis. However, Beale & Majda [14] reported growing errors with time for simulations that used point vortices, and Perlman [142] showed that the rate of error growth with time increased as the amount of smoothing decreased. As smoothing decreases, vortex blobs become closer approximations of point vortices. Based on these results, Beale & Majda [14] and Majda & Bertozzi [120] advocate for vortex blobs and discourage the use of point vortices for flow simulations.

The argument for using point vortices was reinforced by a separate study from Krasny [100] that reported numerical convergence of calculations using point vortices for non-singular vortex sheets. By the 1990s, convergence of numerical methods using point vortices to solutions of the Euler equations was proven for smooth vorticity distributions [71, 81, 37] and for vortex sheets [82]. Despite this advance, point vortices are still not as widely used in numerical simulations as vortex blobs [38, 120]. Some analysis suggests that methods using point vortices may have slower convergence rates than a similar treatment with vortex blobs [14, 120], but there are few numerical studies comparing the two approaches. With a smooth vorticity field, the proof given by [71] and the numerical studies in [33, 100] show that point vortices are capable of providing the basis of a convergent numerical method.

Vorticity distributions for the geophysical fluid dynamics problems in this study do not form singularities. Each calculation begins with a smooth vorticity distribution; the two-dimensional character of the fluid guarantees that the vorticity will remain smooth throughout, as a consequence of Kelvin's circulation theorem. In this investigation, one of our guiding principles is to keep our algorithms as simple as possible. We use point vortices in this work in order to avoid introducing an additional parameter (the vortex blob smoothing parameter) into our algorithms. However, we have designed our code with the capability of using vortex blobs as well; a comparison of the results of calculations using point vortices to the results of vortex blob computations is planned for a future study.

1.5 Spherical Geometry

We use a sphere to represent the Earth. For simplicity, we choose the unit sphere. While it is true that the Earth's mean sea level surface is an ellipsoid because of its rotation and the centrifugal force, its departure from a spherical shape is negligible for this study (its equatorial radius is approximately 21 km larger than its polar radius; this difference is two orders of magnitude smaller than its mean radius of 6371 km [80]). Further analysis and justification for the spherical approximation can be found in most geophysical fluid dynamics texts, e.g., [80, 183].

Definition I.1. The domain $\mathcal{S} \subset \mathbb{R}^3$ is the set of all $\boldsymbol{x} \in \mathbb{R}^3$ such that $|\boldsymbol{x}| = 1$.

$$\mathcal{S} = \left\{ \boldsymbol{x} \in \mathbb{R}^3 : |\boldsymbol{x}| = 1 \right\}$$

We use both Cartesian and spherical coordinate systems to represent $\mathcal{S}$; depending on the context, one is often more convenient than the other. Unlike most mathematical texts which use co-latitude as a spherical coordinate, we use a latitudinal coordinate system which is more aligned with our application of atmospheric modeling. Unfortunately this choice is somewhat arbitrary, even in the literature. For example, [168] and [96] use co-latitude while [187] and [183] use latitude. Choosing the latitude coordinate allows a more direct comparison with the x and y directions of the typical tangent plane approximations of geophysical problems.

Thus we represent a vector $\boldsymbol{x} \in \mathbb{R}^3$ as either $\boldsymbol{x} = [x, y, z]^T$ or as $\boldsymbol{x} = [\rho, \lambda, \theta]^T$, where $\rho \geq 0$ is the radial distance from the origin, $\lambda \in [0, 2\pi)$ is longitude, and $\theta \in [-\pi/2, \pi/2]$ is latitude; the superscript T denotes the transpose. If $\boldsymbol{x}$ is in $\mathcal{S}$, its Cartesian representation will satisfy $x^2 + y^2 + z^2 = 1$ and its spherical coordinate representation will have $\rho = 1$.

The two coordinate systems are related by the equations

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \rho\cos\theta\cos\lambda \\ \rho\cos\theta\sin\lambda \\ \rho\sin\theta \end{bmatrix}, \tag{1.4a}$$

$$\rho = \sqrt{x^2 + y^2 + z^2}, \tag{1.4b}$$

$$\lambda = \arctan\left(\frac{y}{x}\right), \tag{1.4c}$$

$$\theta = \arctan\left(\frac{z}{\sqrt{x^2 + y^2}}\right). \tag{1.4d}$$

When a vector $\boldsymbol{x} \in \mathcal{S}$ has longitude λ, latitude θ, and Cartesian coordinates given by equation (1.4a), we write $\boldsymbol{x} \mapsto (\lambda, \theta)$ to convey this correspondence.

The Cartesian unit vectors $\hat{\boldsymbol{\imath}}$, $\hat{\boldsymbol{\jmath}}$, and $\hat{\boldsymbol{k}}$ are related to the spherical unit vectors $\hat{\boldsymbol{e}}_r$, $\hat{\boldsymbol{e}}_\lambda$, and $\hat{\boldsymbol{e}}_\theta$ by

$$\begin{bmatrix} \hat{\boldsymbol{e}}_\lambda \\ \hat{\boldsymbol{e}}_\theta \\ \hat{\boldsymbol{e}}_r \end{bmatrix} = \boldsymbol{Q} \begin{bmatrix} \hat{\boldsymbol{\imath}} \\ \hat{\boldsymbol{\jmath}} \\ \hat{\boldsymbol{k}} \end{bmatrix}, \tag{1.5}$$

where $\boldsymbol{Q}$ is an orthogonal coordinate change matrix from Cartesian to spherical coordinates [168, 187],

$$\boldsymbol{Q} = \begin{bmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\sin\theta\cos\lambda & -\sin\theta\sin\lambda & \cos\theta \\ \cos\theta\cos\lambda & \cos\theta\sin\lambda & \sin\theta \end{bmatrix}. \tag{1.6}$$

If $\boldsymbol{v}_0$ is represented in Cartesian coordinates, then $\boldsymbol{Q}\boldsymbol{v}_0$ is the same vector in spherical coordinates. Since $\boldsymbol{Q}$ is orthogonal, its inverse is simply $\boldsymbol{Q}^T$.

This matrix is particularly useful with velocity vectors which are tangent to $\mathcal{S}$ in $\mathbb{R}^3$. Let $\boldsymbol{u} \in \mathbb{R}^3$ be a vector tangent to the sphere at $\boldsymbol{x} \in \mathcal{S}$ where $\boldsymbol{x} \mapsto (\lambda, \theta)$. If $\boldsymbol{u}_C$ is the representation of $\boldsymbol{u}$ in Cartesian coordinates, and $\boldsymbol{u}_S$ is its representation in spherical coordinates, then

$$\boldsymbol{u}_S = \boldsymbol{Q}\boldsymbol{u}_C \tag{1.7a}$$

$$\boldsymbol{u}_C = \boldsymbol{Q}^T\boldsymbol{u}_S, \tag{1.7b}$$

where $\boldsymbol{Q}$ is evaluated at $\boldsymbol{x}$. In the work that follows, we will use $\boldsymbol{u}$ to represent a velocity vector, typically in Cartesian coordinates. Scalar variables u and v will be used to denote the zonal (longitudinal) and meridional (latitudinal) components of $\boldsymbol{u}$, respectively.

1.5.1 Differential operators

Equations (1.4) and (1.5) provide the necessary relations to derive spherical coordinate representations of differential operators. Let $f(x, y, z)$ and $F(\rho, \lambda, \theta)$ be scalar functions such that $F = f(\rho\cos\lambda\cos\theta, \rho\sin\lambda\cos\theta, \rho\sin\theta)$. Then

$$\nabla f = \frac{\partial f}{\partial x}\hat{\boldsymbol{\imath}} + \frac{\partial f}{\partial y}\hat{\boldsymbol{\jmath}} + \frac{\partial f}{\partial z}\hat{\boldsymbol{k}}, \tag{1.8a}$$

$$\nabla f = \nabla F = \frac{\partial F}{\partial \rho}\hat{\boldsymbol{e}}_r + \frac{1}{\rho\cos\theta}\frac{\partial F}{\partial \lambda}\hat{\boldsymbol{e}}_\lambda + \frac{1}{\rho}\frac{\partial F}{\partial \theta}\hat{\boldsymbol{e}}_\theta. \tag{1.8b}$$

Similarly, let $\boldsymbol{v} = [v_1(x, y, z), v_2(x, y, z), v_3(x, y, z)]^T = [V_1(\rho, \lambda, \theta), V_2(\rho, \lambda, \theta), V_3(\rho, \lambda, \theta)]^T$. Then

$$\nabla\cdot\boldsymbol{v} = \frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y} + \frac{\partial v_3}{\partial z} \tag{1.9a}$$

$$\nabla\cdot\boldsymbol{v} = \frac{\partial V_1}{\partial \rho} + \frac{2}{\rho}V_1 + \frac{1}{\rho\cos\theta}\frac{\partial V_2}{\partial \lambda} + \frac{1}{\rho}\frac{\partial V_3}{\partial \theta} - \frac{\tan\theta}{\rho}V_3 \tag{1.9b}$$

and

$$\nabla\times\boldsymbol{v} = \left(\frac{\partial v_3}{\partial y} - \frac{\partial v_2}{\partial z}\right)\hat{\boldsymbol{\imath}} + \left(\frac{\partial v_1}{\partial z} - \frac{\partial v_3}{\partial x}\right)\hat{\boldsymbol{\jmath}} + \left(\frac{\partial v_2}{\partial x} - \frac{\partial v_1}{\partial y}\right)\hat{\boldsymbol{k}} \tag{1.10a}$$

$$\begin{aligned}\nabla\times\boldsymbol{v} = &\left(\frac{1}{\rho}\frac{\partial V_2}{\partial \theta} - \frac{\tan\theta}{\rho}V_2 - \frac{1}{\rho\cos\theta}\frac{\partial V_3}{\partial \lambda}\right)\hat{\boldsymbol{e}}_r + \\ &\left(\frac{1}{\rho}V_3 + \frac{\partial V_3}{\partial \rho} - \frac{1}{\rho}\frac{\partial V_1}{\partial \theta}\right)\hat{\boldsymbol{e}}_\lambda + \\ &\left(\frac{1}{\cos\theta}\frac{\partial V_1}{\partial \lambda} - \frac{1}{\rho}\frac{\partial V_2}{\partial \rho} - \frac{1}{\rho}V_2\right)\hat{\boldsymbol{e}}_\theta.\end{aligned} \tag{1.10b}$$

The Laplacian operator is found through a composition of the above operators, $\nabla^2(\,) = \nabla \cdot \nabla(\,)$.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} \tag{1.11a}$$

$$\nabla^2 F = \frac{1}{\rho^2 \cos\theta}\left(\frac{1}{\cos\theta}\frac{\partial^2 F}{\partial \lambda^2} + \frac{\partial}{\partial \theta}\left(\cos\theta \frac{\partial F}{\partial \theta}\right) + \cos\theta \frac{\partial}{\partial \rho}\left(\rho^2 \frac{\partial F}{\partial \rho}\right)\right) \tag{1.11b}$$

The above differential operators are valid on the whole space $\mathbb{R}^3$. To find the form of the operators for $\mathcal{S}$, we note that on the sphere there is no dependence on the radial direction because the radius is unity. It is trivial to implement this restriction to the spherical coordinate representations of the differential operators; all radial derivatives simply vanish and ρ is replaced by its value, $\rho = 1$. To find the Cartesian form of the above differential operators restricted to $\mathcal{S}$ we note that

$$P_0(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{x}^T \tag{1.12}$$

is a projection matrix whose image is the span of the radial unit vector at the point $\boldsymbol{x} \in \mathcal{S}$ [179]. Its complementary projector,

$$P(\boldsymbol{x}) = I - \boldsymbol{x}\boldsymbol{x}^T = \begin{bmatrix} 1 - x^2 & -xy & -xz \\ -xy & 1 - y^2 & -yz \\ -xz & -yz & 1 - z^2 \end{bmatrix} \tag{1.13}$$

is therefore is a projector onto the plane tangent to the sphere at $\boldsymbol{x}$. If $\boldsymbol{v}$ is tangent to $\mathcal{S}$ at $\boldsymbol{x}$, then $\boldsymbol{v} \cdot \boldsymbol{x} = 0$, since the position vector $\boldsymbol{x}$ is also the unit normal on the unit sphere; in this case, $P\boldsymbol{v} = \boldsymbol{v}$.

The matrix P helps to represent the tangential derivatives' Cartesian forms on $\mathcal{S}$ [50, 189]. Let the subscript S denote a differential operator on $\mathcal{S}$.

$$\nabla_S f = P(\boldsymbol{x})(\nabla f) = \nabla f - (\nabla f \cdot \boldsymbol{x})\boldsymbol{x} \tag{1.14a}$$

$$= \frac{1}{\cos\theta}\frac{\partial f}{\partial \lambda}\hat{\boldsymbol{e}}_\lambda + \frac{\partial f}{\partial \theta}\hat{\boldsymbol{e}}_\theta \tag{1.14b}$$

$$\nabla_S \cdot \boldsymbol{v} = [P(\boldsymbol{x})\nabla] \cdot \boldsymbol{v} = \nabla \cdot \boldsymbol{v} - \boldsymbol{x} \cdot \nabla(\boldsymbol{v} \cdot \boldsymbol{x}) + \boldsymbol{x} \cdot \boldsymbol{v} \tag{1.14c}$$

$$= \frac{1}{\cos\theta}\frac{\partial V_1}{\partial \lambda} + \frac{\partial V_2}{\partial \theta} - V_3 \tan\theta \tag{1.14d}$$

$$(\nabla_S \times \boldsymbol{v}) \cdot \hat{\boldsymbol{n}} = (\nabla \times \boldsymbol{v}) \cdot \boldsymbol{x} \tag{1.14e}$$

$$\nabla_S^2 f = \frac{1}{\cos^2\theta}\frac{\partial^2 f}{\partial \lambda^2} - \tan\theta\frac{\partial f}{\partial \theta} + \frac{\partial^2 f}{\partial \theta^2} \tag{1.14f}$$

Equations (1.14a) and (1.14b) define the tangential gradient operator; equations (1.14c) and (1.14d) define the divergence operator on $\mathcal{S}$, and equation (1.14e) extracts the scalar component of the curl. Note that the curl of a vector on the sphere will have only a radial component because of orthogonality. Equation (1.14f) is the horizontal Laplacian operator, also called the Laplace-Beltrami operator, which will appear in our work with the barotropic vorticity equation (chapter III) and the shallow water equations (chapter VI). Derivations and symbolic calculations are much simpler to carry out in spherical coordinates. Results may be translated back into an equivalent Cartesian representation if necessary. The Cartesian forms of the gradient and divergence operators (1.14) are presented to illustrate the nonlinearities inherent to the spherical domain (as seen in the matrix P) and the effect of the curved domain on the form of the differential operators.

The entirety of this work takes place on $\mathcal{S}$, so for convenience in the following chapters we drop the subscript S from the horizontal differential operators. Unless specifically stated otherwise, equations (1.14) are implied.

In chapters IV and V we will confront problems that are posed in spherical coordinates and need to translate them into Cartesian coordinates, or vice versa. It is helpful to note that the following relations hold on the unit sphere.

(1.15a) $$\boldsymbol{x} = \hat{\boldsymbol{e}}_r = [x, y, z]^T$$

(1.15b) $$\hat{\boldsymbol{e}}_\lambda = \frac{1}{\sqrt{1-z^2}}[-y, x, 0]^T$$

(1.15c) $$\hat{\boldsymbol{e}}_\theta = \frac{1}{\sqrt{1-z^2}}[-xz, -yz, 1-z^2]^T$$

(1.15d) $$\cos\lambda = \frac{x}{\sqrt{x^2+y^2}}$$

(1.15e) $$\sin\lambda = \frac{y}{\sqrt{x^2+y^2}}$$

(1.15f) $$\cos\theta = \sqrt{x^2+y^2} = \sqrt{1-z^2}$$

(1.15g) $$\sin\theta = z$$

(1.15h) $$|\boldsymbol{x} - \tilde{\boldsymbol{x}}|^2 = 2(1 - \boldsymbol{x}\cdot\tilde{\boldsymbol{x}})$$

Equation (1.15a) is the unit normal vector and equations (1.15b) and (1.15c) are the unit tangent vectors to the sphere. The norm in equation (1.15h) is the Euclidean chord distance between two vectors $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{S}$. Each of these equations can be derived easily from the geometry of the sphere and equations (1.4) and (1.5).

Mathematically, the spherical coordinate system is very convenient for computing analytical results and exact solutions to equations, when it is possible to do so. However, spherical coordinate representations of the sphere, whether they use latitude or co-latitude, have singularities at the poles. This problem can be seen in (1.14) by noting the cosine terms in the denominators (or, equivalently, the tangent terms in the numerators). These singularities lead to indeterminate terms in the fluid equations' spherical coordinate forms. Of course, the are no physical singularities in the spherical surface; it is smooth everywhere. Instead of using spherical coordinates, we perform numerical computations in Cartesian space with position vectors $\boldsymbol{x}$ represented in x, y, z coordinates. We therefore need to be able to calculate basic geometric quantities such as distance and area in Carte-

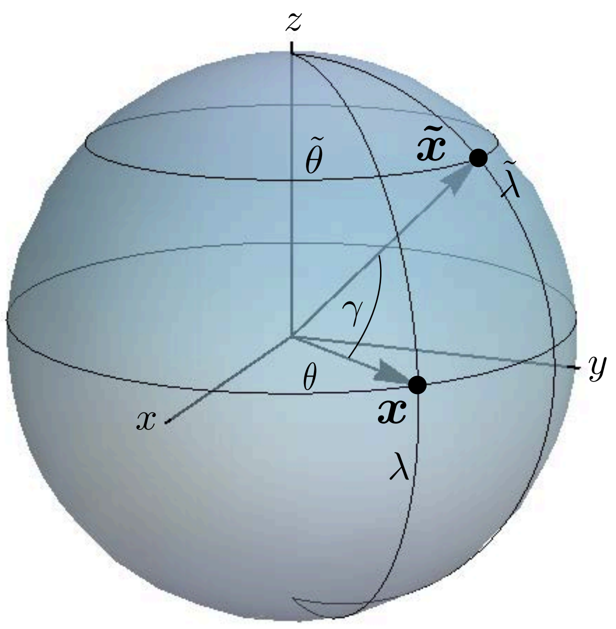


Figure 1.6: Two particles $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ with latitudes $\theta, \tilde{\theta}$ and longitudes $\lambda, \tilde{\lambda}$, respectively. The central angle γ is shown between the two position vectors.

sian coordinates.

1.5.2 Distance and area calculations

Let γ denote the central angle between two points on the sphere, $\boldsymbol{x} \mapsto (\lambda, \theta)$ and $\tilde{\boldsymbol{x}} \mapsto (\tilde{\lambda}, \tilde{\theta})$, as shown in figure 1.6.

The cosine of γ is given by

$$\cos\gamma = \boldsymbol{x} \cdot \tilde{\boldsymbol{x}} = \cos\theta\cos\tilde{\theta}\cos(\lambda - \tilde{\lambda}) + \sin\theta\sin\tilde{\theta}. \tag{1.16}$$

Therefore the angle γ could be obtained by applying the inverse cosine function to equation (1.16),

$$\gamma = \arccos(\boldsymbol{x} \cdot \tilde{\boldsymbol{x}}) = \arccos\left(\cos\theta\cos\tilde{\theta}\cos(\lambda - \tilde{\lambda}) + \sin\theta\sin\tilde{\theta}\right). \tag{1.17}$$

However, the inverse cosine function is ill-conditioned for small angles, and will not give sufficient numerical accuracy when $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ are close to each other. We note that $\sin\gamma =$

$|\boldsymbol{x} \times \tilde{\boldsymbol{x}}|$, and use the numerically stable formula for arc length,

$$\gamma = \arctan\left(\frac{|\boldsymbol{x} \times \tilde{\boldsymbol{x}}|}{\boldsymbol{x} \cdot \tilde{\boldsymbol{x}}}\right), \tag{1.18}$$

instead [66].

On the unit sphere, geodesic distance is equivalent to arc length in magnitude, but has units of length,

$$\operatorname{dist}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \arctan\left(\frac{|\boldsymbol{x} \times \tilde{\boldsymbol{x}}|}{\boldsymbol{x} \cdot \tilde{\boldsymbol{x}}}\right). \tag{1.19}$$

We also note that, with equation (1.16), we can recast the Euclidean distance relation (1.15h) in spherical coordinates. Let $\boldsymbol{x} \mapsto (\lambda, \theta)$ and $\tilde{\boldsymbol{x}} \mapsto (\tilde{\lambda}, \tilde{\theta})$; the square of the chord distance between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ is

$$|\boldsymbol{x} - \tilde{\boldsymbol{x}}|^2 = 2(1 - \boldsymbol{x} \cdot \tilde{\boldsymbol{x}}) = 2\left(1 - \cos\theta \cos\tilde{\theta} \cos(\lambda - \tilde{\lambda}) - \sin\theta \sin\tilde{\theta}\right). \tag{1.20}$$

To find the area of an arbitrary spherical polygon, we first decompose the polygon into spherical triangles. Let p_n be a spherical polygon with n vertices and one interior point, as in figure 1.7. Then p_n can be decomposed into a union of n sub triangles, T_i, $i = 1, \ldots, n$, and its area is the sum of the areas of those sub triangles.

$$A_{p_n} = \sum_{i=1}^{n} A_{T_i} \tag{1.21}$$

To find the area of each spherical triangle, A_{T_i}, we use a formula that relates the area of a spherical triangle to the arc lengths of its sides [191]. Let the triangle's vertices be

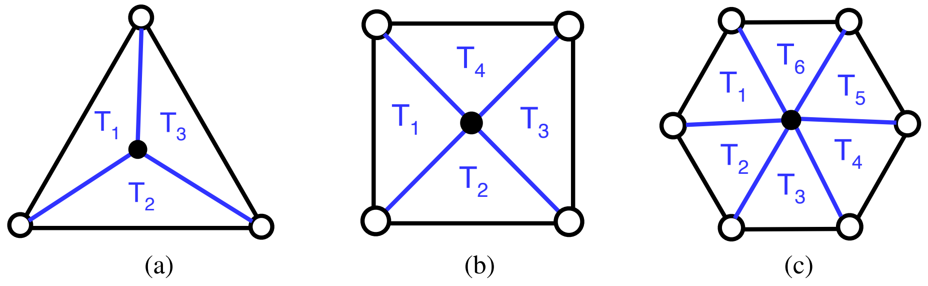


Figure 1.7: Triangular decomposition of arbitrary spherical polygons with one interior particle and particles at each vertex. Examples: (a) triangle, (b) quadrilateral, (c) hexagon.

$\boldsymbol{x}_A, \boldsymbol{x}_B, \boldsymbol{x}_C \in \mathcal{S}$. Its area, A_{T_i}, is given by

$$a = \text{dist}(\boldsymbol{x}_A, \boldsymbol{x}_B) \tag{1.22a}$$

$$b = \text{dist}(\boldsymbol{x}_B, \boldsymbol{x}_C) \tag{1.22b}$$

$$c = \text{dist}(\boldsymbol{x}_C, \boldsymbol{x}_A) \tag{1.22c}$$

$$s = \frac{a+b+c}{2} \tag{1.22d}$$

$$z = \tan\left(\frac{s}{2}\right)\tan\left(\frac{s-a}{2}\right)\tan\left(\frac{s-b}{2}\right)\tan\left(\frac{s-c}{2}\right) \tag{1.22e}$$

$$A_{T_i} = \text{Area}(\boldsymbol{x}_A, \boldsymbol{x}_B, \boldsymbol{x}_C) = 4\arctan\sqrt{z}. \tag{1.22f}$$

1.5.3 Measuring error

We adopt the standard measures of error from the suite of dynamical core tests introduced by Williamson *et al.* [187]. Let $\phi = \phi(\boldsymbol{x}, t)$ represent a quantity of interest in a calculation. Let $\phi_E = \phi(\boldsymbol{x}, t)$ represent its "true" value; this may be exact if an analytical solution is known, or it may be a reference value from a high-resolution computation. We use three error measures, l_1, l_2, and l_∞ to quantify the accuracy of our solutions.

$$l_1(\phi, t) = \frac{\int_{\mathcal{S}} |\phi(\boldsymbol{x}, t) - \phi_E(\boldsymbol{x}, t)|\, dA}{\int_{\mathcal{S}} |\phi_E(\boldsymbol{x}, t)|\, dA} \tag{1.23a}$$

$$l_2(\phi, t) = \frac{\left(\int_{\mathcal{S}} (\phi(\boldsymbol{x}, t) - \phi_E(\boldsymbol{x}, t))^2 dA\right)^{1/2}}{\left(\int_{\mathcal{S}} \phi_E^2(\boldsymbol{x}, t) dA\right)^{1/2}} \tag{1.23b}$$

$$l_\infty(\phi, t) = \frac{\max_{\boldsymbol{x} \in \mathcal{S}} |\phi(\boldsymbol{x}, t) - \phi_E(\boldsymbol{x}, t)|}{\max_{\boldsymbol{x} \in \mathcal{S}} |\phi_E(\boldsymbol{x}, t)|} \tag{1.23c}$$

In chapter V, we complete a test case with additional error measures that apply to those specific problems; they will be described in that chapter, and can be found in the test case description [105].

CHAPTER II

The Flow Map

2.1 Definition of the flow map

The *flow map* is the mathematical object that defines the flow-following trait of Lagrangian fluid particles [30, 120]. The flow map is commonly applied in the field of differential equations [72], and is related to the method of characteristics [73, 111, 122]. The flow map relates the position of each fluid particle, $\boldsymbol{x}$, to the fluid velocity, $\boldsymbol{u}$.

Definition II.1. The flow map $\boldsymbol{x}(\boldsymbol{\alpha}, t)$ is defined by the equations

$$\frac{d\boldsymbol{x}}{dt}(\boldsymbol{\alpha}, t) = \boldsymbol{u}(\boldsymbol{x}(\boldsymbol{\alpha}, t), t) \tag{2.1a}$$

$$\boldsymbol{x}(\boldsymbol{\alpha}, 0) = \boldsymbol{x}_0(\boldsymbol{\alpha}) \tag{2.1b}$$

where $\boldsymbol{\alpha}$ is a Lagrangian parameter which is independent of time.

Equation (2.1a) defines the Lagrangian reference frame, where particles travel along trajectories defined by the fluid velocity. Equation (2.1b) defines the initial conditions for the ODE in (2.1a). Via the flow map, particle positions $\boldsymbol{x}$ are a function of both time t and a Lagrangian parameter $\boldsymbol{\alpha}$.

The Lagrangian parameter serves as a tag for each fluid particle. It provides a means of identifying each particle from the others in the fluid. The choice of what to use as the Lagrangian parameter is up to the modeler. Vortex methods may use circulation [99, 101] or potential vorticity [47] as the Lagrangian parameter, for example. Particles' initial positions are also a common choice (e.g. [57]). In this study we use the latter option, and define $\boldsymbol{\alpha}$ as each particle's initial position

$$\boldsymbol{\alpha} = \boldsymbol{x}_0(\boldsymbol{\alpha}) = \boldsymbol{x}(\boldsymbol{\alpha}, 0). \tag{2.2}$$

Equations (2.1) are responsible for the simplified statement of the continuity equation's Lagrangian form (1.3). When all particles travel with the fluid velocity, there is no flux across particle boundaries due to fluid motion. Hence, with a collection of particles satisfying (2.1), the advective derivative is accounted for implicitly, and need not be calculated.

The flow map (2.1) is the fundamental equation we solve throughout this study. As the context changes, the methods for finding the terms of (2.1) may change, but in all cases the objective is to integrate the flow map forward in time. For example, when solving the barotropic vorticity equation, we use a Biot-Savart law to give the velocity for the right-hand side of (2.1); with the advection equation, the velocity is prescribed; the shallow water equations require two Biot-Savart type integrals to find $\boldsymbol{u}$. In each case, we use the right hand side of (2.1) to advance the fluid particles to their physical positions at the next time step. This process is described symbolically by equation (1.1). The details of this process are governed by the discretization of (2.1) and the numerical methods that accompany that discretization.

2.2 Discretizing the flow map

Solving (2.1) numerically requires a discrete representation of both space and time. Space in this case refers to the spherical domain $\mathcal{S}$ and time is the interval $t \in [0, T]$, where $T > 0$ is the calculation's final time.

We use fourth order Runge-Kutta to integrate the flow map forward in time. A high-order time-integration scheme is necessary when using Cartesian coordinates to represent

$\mathcal{S}$. Numerical experiments confirm that a simple forward Euler time stepping scheme causes fluid particles to rapidly develop $|\boldsymbol{x}| > 1$. Low order methods, like the forward Euler time discretization used by [6], have to re-project fluid particles back to the sphere after each time step so that they satisfy $|\boldsymbol{x}| = 1$. In order to avoid having to apply this correction, we instead use a high-order time stepping method. We show numerical results in chapters IV and V that confirm that our particles satisfy $||\boldsymbol{x}| - 1| \sim O\left(\Delta t^4\right)$ as $\Delta t \to 0$ without any re-projection back to the sphere.

Additionally, choosing a high order time stepping scheme ensures that error due to the time discretization will be much smaller than error due to the spatial discretization. This allows us to focus our efforts on the primary challenge facing a Lagrangian method, the moving spatial discretization.

To begin the discussion of our spatial discretization, we note that since vortex methods rely on a Biot-Savart integral to calculate fluid velocity, the discretization must enable a straightforward quadrature scheme.

Spectral methods (e.g. [169]) also rely on quadrature schemes for their discrete analogs of integral transforms, and many use grids made up of specially chosen Gaussian quadrature points to enable high accuracy integration [5, 62]. A Lagrangian method cannot take advantage of such a discretization, because the positions of Lagrangian particles will change at each time step, which would destroy the error-canceling nature of the original point distribution. A quadrature scheme for a Lagrangian method therefore cannot rely on fixed (Eulerian) quadrature points.

A simple midpoint rule Riemann sum may not be as sophisticated as a high-order Gaussian quadrature scheme, but has been shown to give comparable accuracy for quasi-uniform spatial distributions on $\mathcal{S}$ [5]. The midpoint rule is easy to implement on moving sets of particles using a particle-panel method [57, 184], can handle non-uniform resolutions, and has already been applied to many problems in vortex methods (e.g. [102, 57, 184]). The choice of a midpoint rule quadrature scheme provides the basis for a discretization of $\mathcal{S}$ derived from the particle-panel discretizations from studies of vortex sheet dynamics (e.g. [101, 57]) and spherical fluid flow problems [184].

In this particle-panel method, the sphere is represented as a union of disjoint panels, $\mathcal{S} = \cup_{j=1}^{N} P_j$, which initially corresponds to a discretization of the Lagrangian parameter $\boldsymbol{\alpha}$. These panels are built upon two sets of fluid particles. One set are *active particles* located at panel centers, denoted by $\boldsymbol{x}_j(t) = \boldsymbol{x}(\boldsymbol{\alpha}_j, t), \; j = 1, \ldots, N$. The other set are *passive particles* located at panel vertices, denoted $\boldsymbol{y}_i(t) = \boldsymbol{y}(\boldsymbol{\alpha}_i, t), \; i = 1, \ldots, M$. Both sets of particles $\boldsymbol{x}_j(t)$ and $\boldsymbol{y}_i(t)$ are advected by numerical approximations of the flow map (2.1). In the numerical method developed in the following chapters, active and passive particles must sometimes be treated slightly differently; the use of different variable names will help illustrate those differences when they occur.

Each panel is a polygon on $\mathcal{S}$ with exactly one interior point (its active particle). Both active and passive particles track their Lagrangian coordinate and any physical data relevant to the problem, such as tracer mass or vorticity. Active particles also track the area of their associated panel, and use this area to approximate integrals. Let ϕ be a quantity defined on $\mathcal{S}$ whose integral we wish to compute.

$$\int_S \phi(\boldsymbol{x})\, dA = \sum_{j=1}^{N} \int_{P_j} \phi(\boldsymbol{x})\, dA \tag{2.3a}$$

$$\int_{P_j} \phi(\boldsymbol{x})\, dA \approx \phi(\boldsymbol{x}_j) A_j \quad \text{as } N \to \infty \tag{2.3b}$$

$$\Longrightarrow \int_S \phi(\boldsymbol{x})\, dA \approx \sum_{j=1}^{N} \phi_j A_j \quad \text{as } N \to \infty \tag{2.3c}$$

where $\boldsymbol{x}_j$ is the interior active particle of panel P_j, and A_j is the panel's area calculated with formula (1.22) and the triangular decomposition of the panel shown in figure 1.7. $\phi_j = \phi(\boldsymbol{x}_j)$ is the value of ϕ defined on active particle $\boldsymbol{x}_j$.

The particle-panel discretization provides a helpful interpretation of the flow map discretization, in addition to the basic quadrature scheme. In many applications particle methods can be mesh-free, collections of moving particles only [38, 120]. This approach is attractive for simulations of vortex sheet dynamics, where vorticity is concentrated on a small subset of the spatial domain (e.g. [158]). On the sphere, however, a mesh-free approach makes it difficult to answer a fundamental question: How much fluid does each particle represent? In several studies of moving vortices on the sphere [158, 137, 136] circulations are arbitrarily assigned to each particle at the beginning of a computation in a manner that is helpful at illustrating the dynamics of vorticity, but is not directly related to a specific observed phenomenon of the atmosphere or ocean. Panels provide a notion of area and an answer to the question of how much fluid each particle represents; this allows a more direct association of individual particles' vorticity with observed global vorticity distributions via Stokes' theorem.

Circulation on a panel Γ_j is defined as the line integral of velocity around the boundary of panel P_j. Let the differential $d\boldsymbol{r}$ represent an increment of arc length along the perimeter of P_j; then circulation is defined as

$$\Gamma_j = \oint_C \boldsymbol{u} \cdot d\boldsymbol{r}, \tag{2.4}$$

where C is the closed boundary of P_j. Stokes' theorem allows us to represent Γ_j in terms of relative vorticity, $\zeta = \hat{\boldsymbol{n}} \cdot (\nabla \times \boldsymbol{u})$. Similarly to equation (2.3), as more and more panels are used to represent the sphere, midpoint rule quadrature becomes a better approximation. Let ζ_j denote the vorticity at the active particle associated with panel P_j, and A_j be the panel's area.

$$\Gamma_j = \int_{P_j} \zeta(\boldsymbol{x})\, dA \tag{2.5a}$$

$$\Gamma_j \approx \zeta_j A_j \quad \text{as } N \to \infty \tag{2.5b}$$

Equation (2.5a) is the result of applying Stokes' theorem to (2.4). Equation (2.5b) provides a means of interpreting the flow map discretization. As our polygonal panels become smaller and smaller, each panel becomes a better approximation of the circulation around that panel's active particle.

An important consequence of this interpretation for dynamical core applications is the fact that vorticity is commonly used as a prognostic variable in atmospheric dynamics. Many problems and test cases are defined in terms of vorticity, and it is more intuitive for modelers to analyze than circulation. We can assign each active particle a vorticity value by sampling a vorticity distribution, and recover the circulation Γ_j of each active particle by finding the area of each polygonal panel and multiplying with the formula given by equation (2.5b).

The choice of what variety of polygon to use for the Lagrangian panels is similar to the question of what variety of grid is best for an Eulerian model. Global models began with latitude-longitude grids [188]. Latitude-longitude grids are locally orthogonal and a natural choice because of their correspondence to the latitude-longitude coordinates of geography. Despite those advantages, latitude-longitude grids are hampered by problems of anisotropy and converging grid lines at the poles. Longitude lines converge to a single point at both poles; as a result, grid boxes (panels) are much smaller near the poles than at the equator. This effect intensifies as grids are refined and polar grid boxes have longitudinal separations that converge toward zero. The singularity of the latitude-longitude grid is known as the *pole problem* and many other tilings of the sphere have been developed to avoid it; these alternatives are summarized in [188, 108, 167]. A common approach is to use regular polyhedra whose faces have constant angular separation regardless of grid resolution.

The idea of using regular polyhedrons to approximate the sphere was proposed by

Cubed sphere quadrilaterals					
N	1536	6144	24576	98304	393216
$\Delta\lambda$	5.2	2.6	1.4	0.7	0.35
Δx (km)	578	289	155	77	39

Icosahedral triangles					
N	1280	5120	20480	81920	327680
$\Delta\lambda$	8.6	4.3	2.2	1.1	0.59
Δx (km)	956	478	244	122	65

Table 2.1: Mesh size $\Delta\lambda$ (in degrees) as a function the number of panels N in a uniform mesh. Arc lengths Δx (in kilometers) correspond to approximate panel edge lengths on the Earth (mean radius 6371 km).

Sadourny [157]. Other choices, such as overset grids, are possible [188], but are less applicable to this work. Two common choices for discretizing the sphere using polyhedra are based on the icosahedron [13] and the cube [153]. We use these two polyhedra as the basis for two different kinds of Lagrangian panels. The icosahedron leads to triangular panels while the cube produces quadrilateral panels; examples are shown in figure 2.1 (a) and (b). It is common in the dynamical core community to use mesh size to discuss the resolution used by a computation. While is it more natural for our method to use the number of panels N, in order to compare our results to other dynamical cores, we define mesh size $\Delta\lambda$ as the average length of a panel edge at $t = 0$. The correspondence of $\Delta\lambda$ to N for uniform meshes, as well as the approximate length of a panel edge on an Earth-sized sphere, are shown in table 2.1.

The dual of the icosahedral triangle mesh is a grid of primarily hexagonal panels, with a few pentagonal panels at the vertices of the original icosahedron [152]. Such a mesh on its own adds little to a discussion that already includes triangles and quadrilaterals; the midpoint rule has straightforward implementations on any type of polygon (see figure 1.7). However, when combined with algorithms for computing Voronoi diagrams of arbitrary point distributions on $\mathcal{S}$, hexagonal grids can be quite useful to our Lagrangian method. Voronoi grids are discussed in section 2.2.2. A set of hexagonal Voronoi panels is shown in figure 2.1 (c).

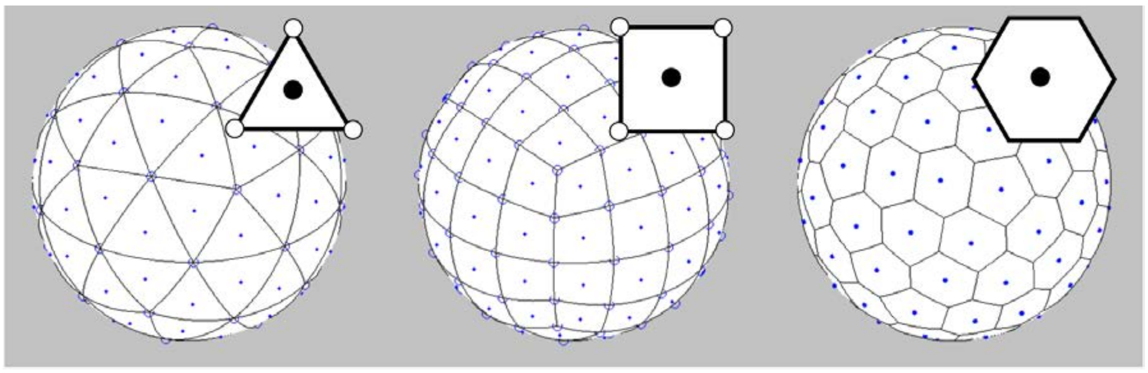


Figure 2.1: Examples of particle-panel discretizations of the sphere. Active particles are shown as filled circles. Passive particles are hollow circles. (a) Icosahedral triangle mesh with $N = 96$ panels. (b) Cubed sphere mesh, $N = 80$. (c) Voronoi mesh, $N = 92$. Passive particles are not shown in (c) because they are not stored separately for Voronoi grids.

Having decided on a quadrature rule and a discretization strategy, we next need a computational data structure that is capable of implementing these discretizations on moving sets of fluid particles.

2.2.1 Computational implementation and data structures

The primary purpose of the particle-panel discretization is to facilitate quadrature computations over the discrete domain. Consequently, we need a data structure that is capable of quickly calculating integrals over $\mathcal{S}$ such as (2.3). Passive particles may belong to two or more panels, and for the sake of conserving memory, their data should not be duplicated at each active particle whose panel happens to connect to them. As a first guess, we may decide that we only need a means of identifying which passive particles define the vertices of the panel associated with each active particle. This would lead to a data structure that maintains the two sets of particles, with active particles also tracking pointers to the passive particles that define the vertices of their associated panel. In the field of computational geometry, this is known as a face-vertex relation [159]. Our Lagrangian panels correspond to topological faces, and our passive particles are the vertices of those faces.

A face-vertex relation would provide a means of quickly calculating quadrature, but not much else. A review of data structures and algorithms for computational geometry [159, 39] reveals several fundamental procedures that good data structures should be able to complete quickly. Point query methods are a primary example. In this work, a point query procedure would output a pointer to the panel that contains an arbitrary input point $\boldsymbol{x}_0 \in \mathcal{S}$. A simple face-vertex relation does not provide a means of solving this problem quickly, and would have to rely on a brute-force search algorithm that looped over the entire panel array to find the closest active particle to $\boldsymbol{x}_0$.

Point query procedures are not necessary for computing the Biot-Savart integrals we will use to calculate velocity, but are vital for the remeshing algorithm introduced in section 2.4, which relies on local interpolation methods. Additionally, point query methods are essential for representing data from Lagrangian particles and panels on other grids, such as a latitude-longitude grid for contour plotting, or for coupling with other models or data sets. Fortunately, computational geometry texts provide a means of accelerating point query procedures: the introduction of edges [159, 39]. A recent paper [41] provides an interactive JavaScript that illustrates several different point query algorithm designs that rely on edges. Several data structures provide face-edge-vertex relations, but we adopt a variant of the *winged-edge* data structure due to its simultaneous representation of both a primary mesh and its dual [159, 140].

Each winged edge has pointers to the passive particles at its starting vertex $\boldsymbol{x}_A$ and its ending vertex $\boldsymbol{x}_B$, in an edge-vertex relation (figure 2.2). This induces a direction on the edge, which enables it to also point to its left and right panels, P_L and P_R, in an edge-face relation.

We enforce a strict counter-clockwise ordering of panel edges and panel vertices relative to each active particle. By relying on the fact that our panels are either always triangu-

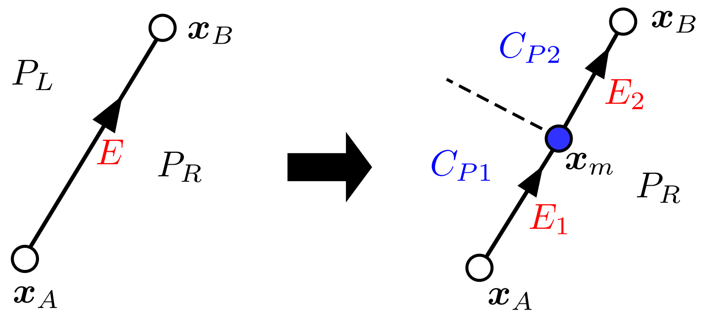


Figure 2.2: Dividing edge E while refining panel P_L. Child edges E_1 and E_2 inherit their orientation from parent edge E and connect to new passive particle $\boldsymbol{x}_m$ at the parent edge's midpoint. Child edges $E_{1,2}$ connect to new child sub-panels C_{P1} and C_{P2} as well as original panel P_R. Dashed line represents a new interior edge.

lar or always quadrilateral, we are able to quickly return pointers to the edges and vertices around a particular panel and that panel's adjacent panels. These methods are simplified versions of the algorithms described by [140, chapter 4] and [159, section 2.2.1.2] which do not assume a fixed polygon type.

For additional simplicity in coding, we combine active particles and their associated panels into one structure. Our algorithms are implemented using three primary data structures: *Particles*, *Edges*, and *Panels*. Passive particles are kept in the *Particles* data structure, and active particles are kept in the *Panels* data structure. The topology and connectivity of the mesh are maintained with *Edges*, which connect two passive particles and track left and right panels, and *Panels* which maintain area and track edges. We vectorize data structures for passive particles, edges, and panels with Fortran 90/95's column-major memory striping in mind. Our basic data structures for triangular or quadrilateral panel sets are presented in tables 2.2, 2.3 and 2.4.

Grids on $\mathcal{S}$ are generated by initializing particles, edges, and panels with either the icosahedron (figure 2.3) or the cubed sphere (figure 2.4). Vertices of the polyhedron and the centroids of its faces are normalized so that each satisfies $|\boldsymbol{x}| = 1$. The initial data structures are then recursively divided to a desired resolution as illustrated in figures 2.5 and 2.6.

Particles Data Structure			
Variable	Kind	Size	Description
`N`	integer	scalar	current number of passive particles
`N_max`	integer	scalar	maximum number of passive particles allowed in memory
`x`	double precision	$3 \times N_{\max}$	Physical position, $\boldsymbol{y}$
`x0`	double precision	$3 \times N_{\max}$	Lagrangian parameter, $\boldsymbol{\alpha}$
`tracer`	double precision	$N_{\max} \times N_T$	tracers, ϕ
`absVort`	double precision	$N_{\max}$	absolute vorticity, ω
`relVort`	double precision	$N_{\max}$	relative vorticity, ζ
`edges`	integer	$p \times N_{\max}$	Pointers to edges incident to each passive particle

Table 2.2: Passive particles data structure. N_T is the number of tracers (set by user). p is the type of panel: $p = 3$ for triangular panels, $p = 4$ for quadrilateral panels.

Edges Data Structure			
Variable	Kind	Size	Description
`N`	integer	scalar	current number of edges
`N_max`	integer	scalar	maximum number of edges allowed in memory
`verts`	integer	$2 \times N_{\max}$	edge vertices (pointers to particles)
`leftPanel`	integer	$N_{\max}$	pointers to panels
`rightPanel`	integer	$N_{\max}$	pointers to panels
`hasChildren`	logical	$N_{\max}$	true if edge has been divided
`children`	integer	$2 \times N_{\max}$	pointers to child edges

Table 2.3: Edges data structure.

Panels Data Structure			
Variable	Kind	Size	Description
`N`	integer	scalar	current number of panels
`N_active`	integer	scalar	current number of undivided panels
`N_max`	integer	scalar	maximum number of panels allowed in memory
`x`	double precision	$3 \times N_{\max}$	Physical position of active particles, $\boldsymbol{x}$
`x0`	double precision	$3 \times N_{\max}$	Lagrangian parameter of active particles, $\boldsymbol{\alpha}$
`tracer`	double precision	$N_{\max} \times N_T$	tracers at active particles, ϕ
`absVort`	double precision	$N_{\max}$	absolute vorticity at active particles, ω
`relVort`	double precision	$N_{\max}$	relative vorticity at active particles, ζ
`area`	double precision	$N_{\max}$	area of panels, A
`edges`	integer	$p \times N_{\max}$	Pointers to edges around each panel
`vertices`	integer	$p \times N_{\max}$	Pointers to particles at panel vertices
`hasChildren`	logical	$N_{\max}$	true if panel has been divided
`children`	integer	$4 \times N_{\max}$	pointers to child sub-panels
`nest`	integer	$N_{\max}$	recursion level of each panel

Table 2.4: Combined data structure for active particles and panels. N_T is the number of tracers (set by user). p is the type of panel: $p = 3$ for triangular panels, $p = 4$ for quadrilateral panels.

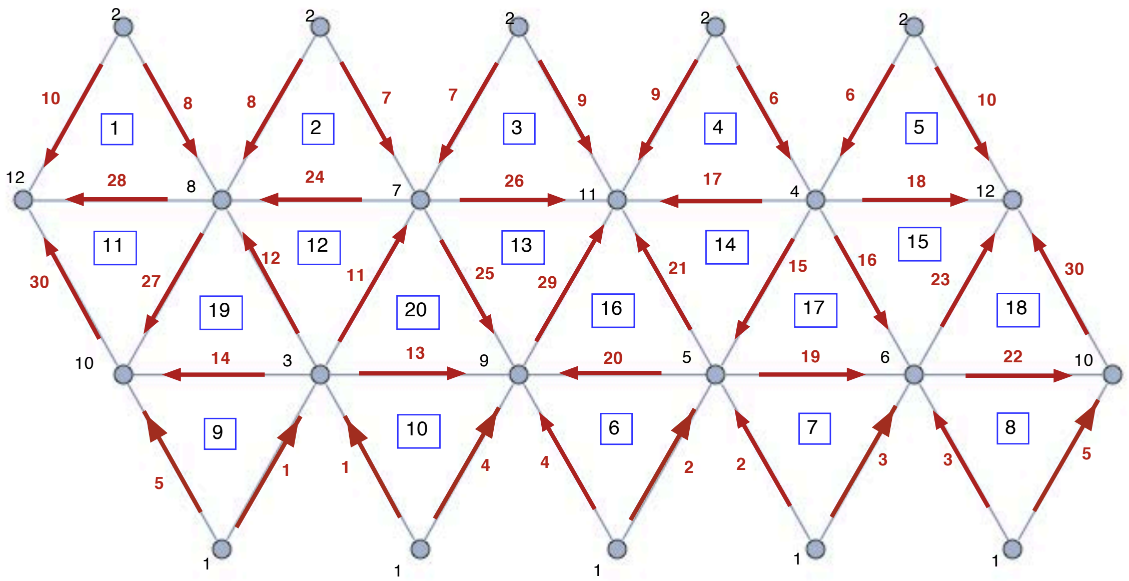


Figure 2.3: Initial data structures for icosahedral triangulations. Passive particles (black numbers, blue filled circles) are the vertices of each triangular panel. Active particles (black numbers, blue square outlines) are initially at the centroid of each panel. Edges (red) connect passive particles while separating panels. Numbering corresponds to the output of Mathematica's `PolyhedronData["Icosahedron"]` functions.

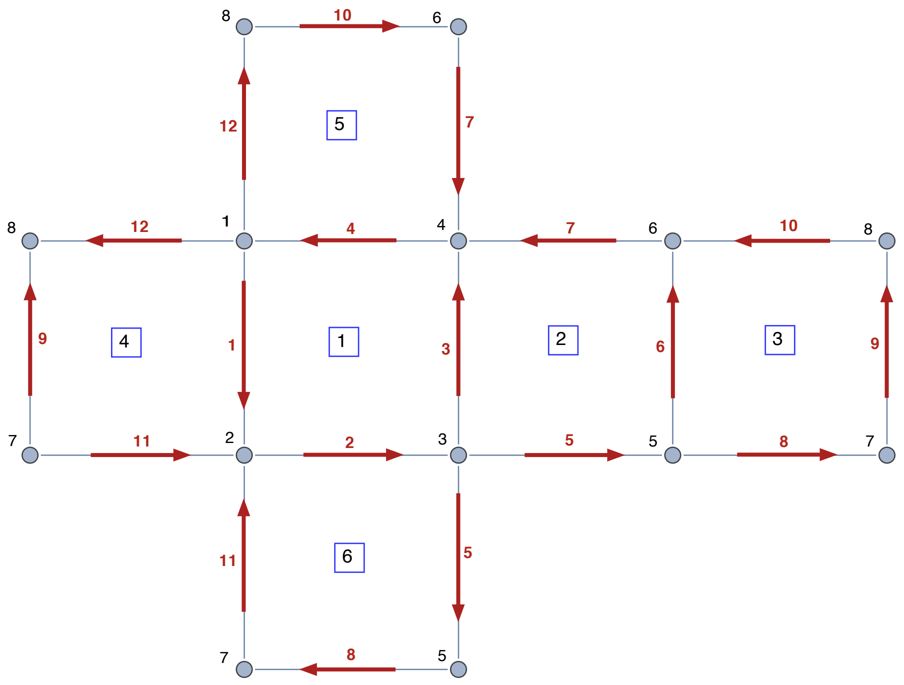


Figure 2.4: Initial data structures for the cubed sphere. Passive particles, edges, and active particles are as indicated in figure 2.3

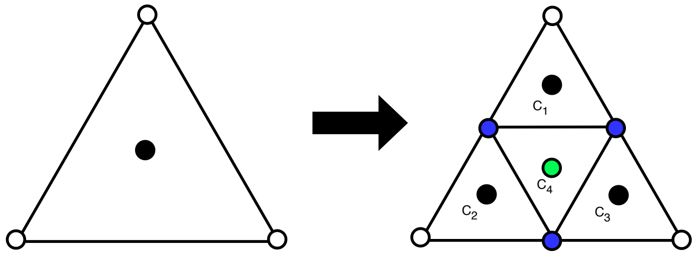


Figure 2.5: Refining a triangular panel. White open circles are existing passive particles. Blue filled circles indicate new passive particles. C_i denotes the ith child of the parent panel. Active particles are shown as solid black circles. The parent panel's active particle becomes the active particle of the 4th child panel (green circle).

Dividing triangular panels is nearly the same process as dividing quadrilateral panels. Each edge of the parent panel is divided, and a new passive particle is created at its midpoint. New edges retain the orientation of their parent edge (figure 2.2). New interior edges connect the new particles with arbitrarily determined orientation that is consistent across the entire panel set. New active particles are created at the centroids of the child sub-panels. The only significant difference between dividing a triangular panel and a quadrilateral panel lies in how the parent panel's active particle is handled. These points are indicated by the green circles in figures 2.5 and 2.6. For triangular panels, the parent panel's active particle is simply relocated to the centroid of the inner child panel, and becomes that panel's active particle as well. In quadrilateral panels, the parent's active particle becomes a passive particle in each of the child panels.

It is possible that an adjacent panel has already been divided and a new passive particle already exists at the midpoint of the edge shared by the two panels. In this case, the panel division procedure simply connects the sub panels to that existing particle and to the two child edges that had been previously created.

Each set of panels has a natural tree structure due to this recursive refinement procedure. The tree structure can be used to accelerate a point query algorithm, but a tree search is insufficient on its own because the Lagrangian particles may move outside the boundaries

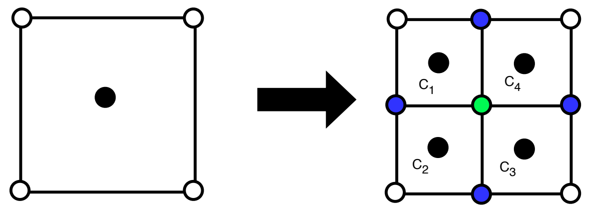


Figure 2.6: Refining a quadrilateral panel. Passive particles are as indicated in figure 2.5. The parent panel's active particle becomes a passive particle of all 4 child panels (green circle).

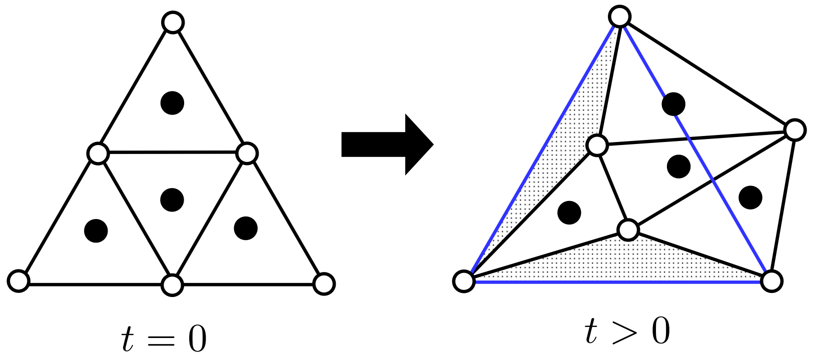


Figure 2.7: A situation where a tree search point query would give an incorrect result due to particle motion. Parent panel is outlined in blue. A query point located in the stippled area would be incorrectly located within one of the four child panels shown.

of their parent panels (see figure 2.7). Consequently, we use the tree structure to provide an initial guess to an adjacent panel search algorithm. This method is similar to the "jump and walk" strategy described in [41]. It assumes panels are convex polygons on $\mathcal{S}$. A schematic of the algorithm is presented in figure 2.8.

The point query procedure takes as input an existing set of `Particles`, `Edges`, and `Panels`, and the coordinates of the query point $\boldsymbol{x}_0$. To initialize a tree search, the panel of the original polyhedron (either the cubed sphere or the icosahedron) that contains $\boldsymbol{x}_0$ is found by finding the minimum distance between $\boldsymbol{x}_0$ and the centroids of the root polyhedron's faces. This is the root of the tree search, and its index is stored as P in figure 2.8. The tree search is a recursive search beginning at the root panel and returning the index P of the panel at the leaf of tree whose centroid is closest to the query point. Since this leaf panel may not contain the query point, the walk search commences at this panel. For each edge of P, the panel on the other side of the edge, P^*, is adjacent to P. If an adjacent panel

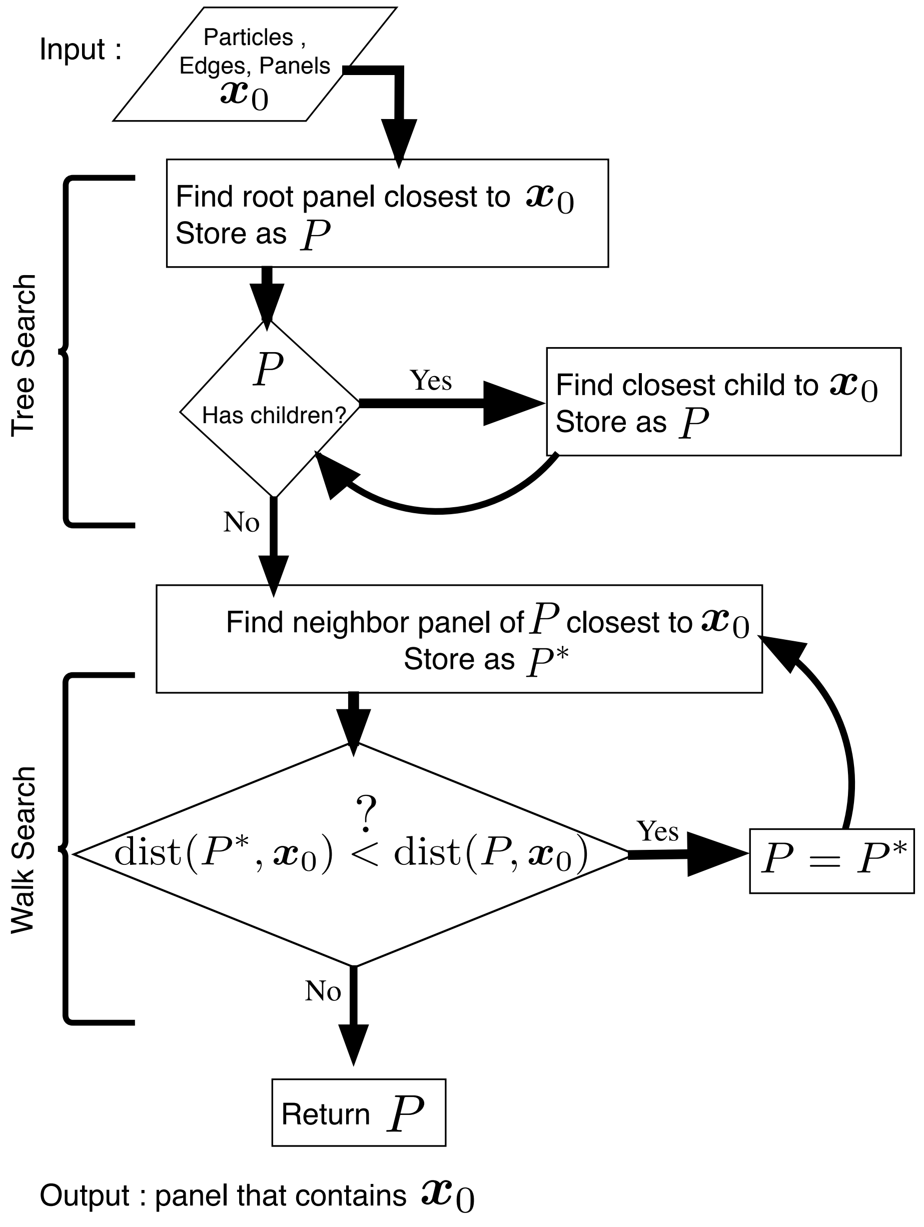

Input :
Particles , Edges, Panels $\boldsymbol{x}_0$
Find root panel closest to $\boldsymbol{x}_0$ Store as P
Tree Search
P Has children?
Yes
No
Find closest child to $\boldsymbol{x}_0$ Store as P
Find neighbor panel of P closest to $\boldsymbol{x}_0$ Store as P^*
Walk Search
$\mathrm{dist}(P^*, \boldsymbol{x}_0) \overset{?}{<} \mathrm{dist}(P, \boldsymbol{x}_0)$
Yes
$P = P^*$
No
Return P
Output : panel that contains $\boldsymbol{x}_0$

Figure 2.8: Diagram of point location query algorithm. The closest panel is defined to be the panel whose centroid is the minimum distance from the query point $\boldsymbol{x}_0$. The first loop is a tree search; the second loop is a walk search.

has a centroid that is closer to $\boldsymbol{x}_0$ than the centroid of P, the walk search restarts at that panel; otherwise, the search has converged and returns the index of panel P.

Asymptotically, the worst-case timing of a tree search is $\mathcal{O}(\log N)$ and worst-case timing of a walk search is $\mathcal{O}\left(\sqrt{N}\right)$. Walk search performance is entirely dependent on the initial guess; using the tree search to provide a good initial guess accelerates the walk search portion of our search algorithm. Additionally, since the tree depth is relatively short, we find that in practice the timing of the point query algorithm shown in figure 2.8 is $\mathcal{O}(1)$, which is a significant improvement over an $\mathcal{O}(N)$ brute force approach. The remeshing algorithm introduced in section 2.4 will need to locate N new active particles and M new passive particles within an existing mesh of N distorted panels. Thus we expect the overall search time for the remeshing algorithm would scale as $\mathcal{O}\left((N+M)^2\right)$ for a brute force search and as $\mathcal{O}(N+M)$ for the fast search. Empirical timing results that verify these expectations for both the "fast" algorithm (figure 2.8) using the winged edges and a brute force "slow" search are shown in figure 2.9.

The winged edges that connect passive particles and enable the fast search are also a common data structure for Delaunay Triangulation and Voronoi Diagram algorithms [140, 159]. Voronoi diagrams have recently been used as horizontal discretizations in some Eulerian dynamical cores (e.g. [151, 162]). In the context of this study, Voronoi diagrams yield a collection of polygons on $\mathcal{S}$ centered on each active particle, similar to the triangular and quadrilateral panels we have just described. Voronoi diagrams have several geometric properties that may be advantageous in Lagrangian particle-panel methods [7, 140]. We discuss their properties and potential use for our algorithms in the following section.

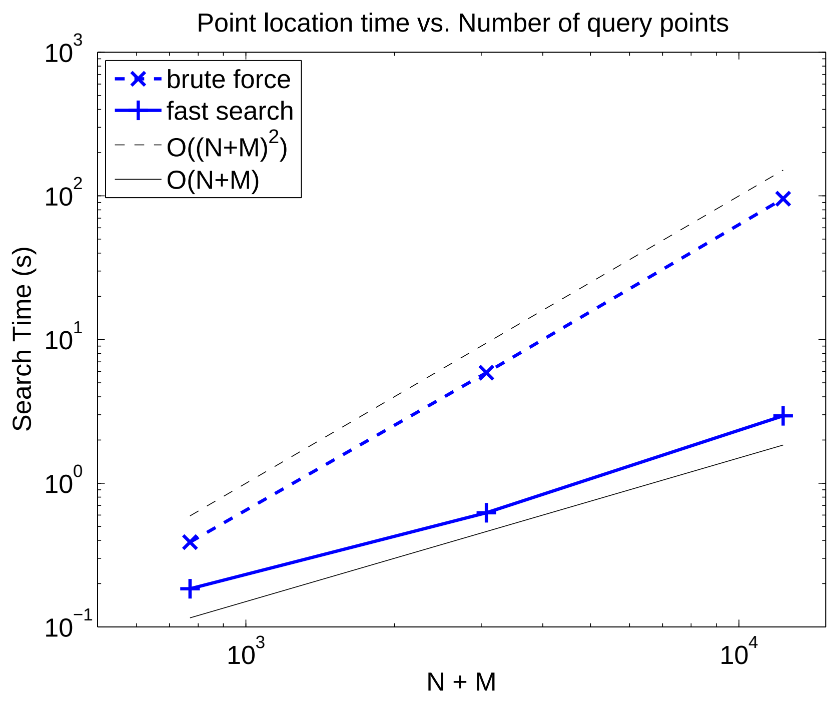


Figure 2.9: Time required to locate N new active particles and M new passive particles within a set of N distorted panels. Brute force search time (blue dashed line) scales as $O\left((N+M)^2\right)$ as $N \to \infty$. Fast search time (solid blue line) using the algorithm from figure 2.8 scales as $O(N+M)$.

2.2.2 Delaunay triangulations and Voronoi diagrams

Okabe *et al.* [140] provides a thorough review of the properties, definitions and applications of Voronoi diagrams for applications ranging from coding theory to stochastic processes. The fundamental importance of Voronoi diagrams and their dual, the Delaunay triangulation, is evidenced by their prominent inclusion in most introductory texts in computational geometry (e.g. [39, 159]). These texts introduce a terminology that is a mixture of combinatorics, topology, and graph theory; we present the definition in the context of particles and panels.

Definition II.2. The *Voronoi polygon* V_j surrounding active particle $\boldsymbol{x}_j$ is the set of all $\boldsymbol{x} \in \mathcal{S}$ closer to $\boldsymbol{x}_j$ than any other active particle,

$$V_j = \left\{\boldsymbol{x} \in \mathcal{S} \;:\; \operatorname{dist}(\boldsymbol{x} - \boldsymbol{x}_j) \leq \operatorname{dist}(\boldsymbol{x} - \boldsymbol{x}_{k \neq j})\right\}, \tag{2.6}$$

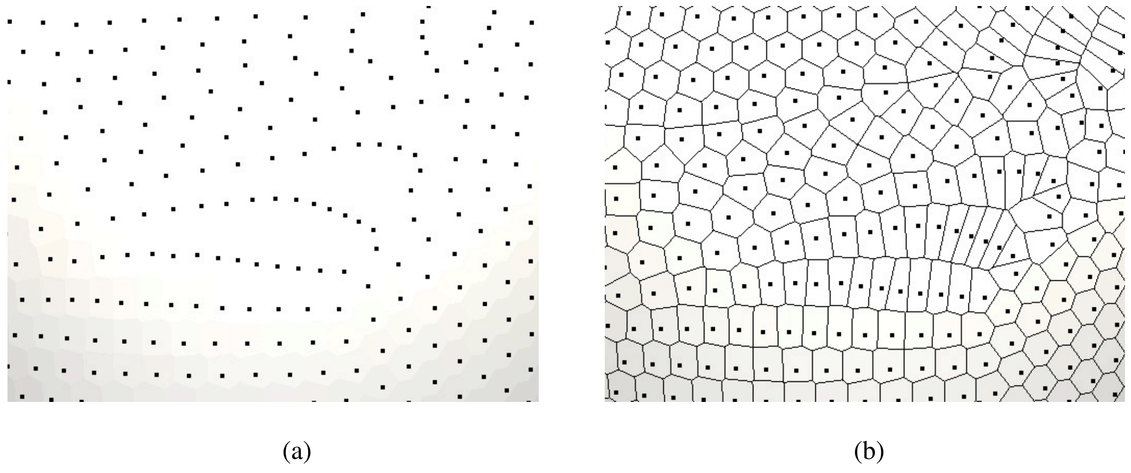


Figure 2.10: Example of a Voronoi diagram. A set of active particles (a) and their Voronoi polygons (b).

where we assume that

$$\mathcal{S} = \cup_{j=1}^{N} V_j.$$

Equality in (2.6) holds only on the edge between two neighboring polygons. Thus each Voronoi polygon overlaps by at most one edge with any other Voronoi polygon [140]. An illustration is shown in figure 2.10.

If the Voronoi polygons for an arbitrary set of active particles can be calculated efficiently at each time step, the set of Voronoi polygons provides a tiling of the sphere which can be used for quadrature calculations. Such an algorithm would have no need to track passive particles separately; their locations would be implicitly calculated as the vertices of each Voronoi polygon at each time step.

Geometric properties of both the Voronoi diagram and its dual, the Delaunay triangulation, are given in [140, chapter 2], [159, section 2.2.1.4], and [39, chapters 7 and 9]. These properties define a locally optimal discretization of $\mathcal{S}$ for a given set of active particles. By definition the Voronoi polygon that defines each active particle's panel provides the answer to the question of how much fluid each active particle represents. As the active particles move, we may simply recalculate their Voronoi diagram on $\mathcal{S}$ and avoid maintaining a separate set of passive particles.

To generate Voronoi grids, such as the one shown in figure 2.1 (c), we initialize an icosahedron as if we were going to use a triangular panel set. Instead of using the triangles, we use their dual; the vertices of each triangle, which would be the passive particles in a triangular panel set, become the active particles of the Voronoi panels. This collection of triangular vertices from the icosahedral mesh becomes the input to the Voronoi diagram algorithm. That algorithm outputs the vertices of the Voronoi polygons, which are the passive particles for a Voronoi mesh, and panel areas are assigned as usual. Time integration for each active particle is handled the same way as the cubed sphere and icosahedral triangle meshes. However, since the passive particles of a Voronoi grid are defined as the vertices of the Voronoi polygons, they need not be stored (or advected) separately; they are determined at each time step.

This idea of recalculating the Voronoi diagram at each time step for a set of Lagrangian fluid particles was proposed for spherical geophysical fluid dynamics problems by Augenbaum and Peskin [7] in 1985. Their paper also provides an algorithm that computes a Voronoi diagram on $\mathcal{S}$ for N active particles in $\mathcal{O}(N)$ time; it is a spherical implementation of the planar incremental algorithm presented in Okabe *et al.* [140, section 4.3] and the Voronoi algorithm for periodic domains described by [20]. Voronoi diagrams may also be computed by first finding the Delaunay triangulation of a set of active particles. Algorithms that find Delaunay triangulations in $\mathcal{O}(N)$ time are given by [159, 39]. The circumcenter of each Delaunay triangle corresponds to the vertex of a Voronoi polygon; this correspondence can be used to generate a Voronoi diagram from a Delaunay triangulation. Renka [146] provides an ACM TOMS Fortran 77 software library called STRIPACK that implements this algorithm on the sphere. STRIPACK is the predominant software used for spherical Delaunay triangulations today [25, 90].

The locally optimal set of panels defined by the Voronoi diagram for a set of active particles provides an attractive basis for a Lagrangian method, as introduced by [7]; however, such techniques are limited by the difficulty of computing Voronoi diagrams accurately. Even in the planar case, Okabe *et al.* report that double precision arithmetic may be insufficient to represent a Voronoi diagram with topologically consistent properties [140, section 4.6]. Problems of finite numerical accuracy are amplified on the sphere, where geometric calculations are complicated by metric terms that involve transcendental functions. Augenbaum and Peskin [7, 6] showed that their algorithm worked for a relatively small number of active particles, $N \sim O\left(10^3\right)$. But as the number of particles increases, the difficulty of computing both the Voronoi diagram and the Delaunay triangulation also increases [25]. Algorithms that rely on software capable of arbitrary precision arithmetic are currently being developed [25, 90].

For this work we rely on STRIPACK [146], whose Delauany triangulation algorithms have proven capable for use in existing dynamical cores [151, 162]. STRIPACK's focus is on the Delaunay triangulation, which will be used as part of the remeshing algorithm (section 2.4). Although we have had no problems with STRIPACK's Delaunay triangulations, we find that STRIPACK's Voronoi algorithm occasionally outputs an error or freezes; this behavior has also been noted by [25]. Consequently, while we introduce the capability to use Voronoi panels in this chapter, and discuss some of their results in sections 2.4, 4.2, and 4.4, the majority of our work will use the triangular and quadrilateral panels of the icosahedron and cubed sphere instead.

2.3 Mesh distortion and adaptive refinement

The problem of computing motion at a variety of interacting scales has been a topic of research since the beginning of atmospheric modeling. Behrens [15, section 1.1] cites the example of a mid-latitude cyclone, whose vortex filaments are on the order of 5-10km wide, but whose dynamics are driven by planetary Rossby waves with characteristic length scales on the order of 1000-10000 km as evidence of the inherent challenge. Such filamentary features require closely spaced computational elements to resolve, but these features are often small relative to the surface area of the sphere. Consequently, uniformly high-resolution computations can be inefficient; away from the cyclone, the fluid may not have such small-scale features. Flow away from the cyclone may therefore be resolved with a much coarser spacing of computational elements at a much lower cost. An adaptive method that is able to increase resolution where it is needed, and decrease resolution where it is able, could make much more efficient use of computational resources. Behrens's work is focused on adaptive finite element methods; other Eulerian techniques (e.g. finite volume methods [85]) are also developing adaptive methods to efficiently resolve small-scale features without having to resort to uniformly high resolution computations.

In a Lagrangian setting, the need for adaptivity is increased by the fact that the computational elements are moving. Adaptivity is not only necessary to resolve small scales that may develop, but also to maintain accuracy as particles may move away from some areas of the domain. Regions of the domain that become depleted of fluid particles rapidly lose accuracy in a Lagrangian computation. An algorithm that detects these regions as they develop, then adaptively refines the region to add resolution before accuracy is lost is a natural strategy. This strategy was employed with vortex methods by Krasny [101] and Feng *et al.* [57] for vortex sheet dynamics, and by Wang [184] for the barotropic vorticity equation on the sphere.

Our adaptive refinement goals, therefore, are based on two principles. We want to resolve small scales of motion that develop in geophysical flows and to maintain the spatial accuracy of our Lagrangian method. In order to do so, we must decide on what criteria to use to trigger adaptive panel refinement.

Behrens [15, section 2.6] suggests several different refinement criteria that are based on either estimates of numerical error or the physics of the developing model fluid. In a Lagrangian simulation, there is a third option: the state of the mesh itself. Krasny and his collaborators [101, 57, 184] develop refinement criteria based on the amount panels (in 1D) or panel edges (2D) have been stretched, or how much curvature a panel has developed.

In this thesis, we focus on physics-based criteria. The use of either error estimates or measurements of the Lagrangian mesh for refinement criteria are topics for possible future research. We have developed two physics-based criteria to use for adaptive panel refinement, based on our experience with uniform-mesh simulations.

The first criterion measures either the total amount of mass in a panel for advection and transport problems,

$$M_j = \phi_j A_j < \epsilon_1 \tag{2.7}$$

or the magnitude of circulation on a panel for barotropic vorticity problems,

$$\left|\Gamma_j\right| = \left|\zeta_j\right| A_j < \epsilon_1, \tag{2.8}$$

where ϵ_1 is a user-specified tolerance. If either M_j or $\left|\Gamma_j\right|$ exceed ϵ_1, then panel j is refined. This criterion refines regions of high tracer mass density in advection problems and vortex cores in barotropic vorticity problems.

The second criterion measures the total variation of key physical quantities on each panel. The total variation in vorticity is defined as

$$V_\zeta = \max_{P_j} \zeta - \min_{P_j} \zeta < \epsilon_2, \tag{2.9}$$

where $\max_{P_j}$ and $\min_{P_j}$ give the maximum or minimum value on panel j across all of its particles (active and passive), and ϵ_2 is a separate user-specified tolerance. The total

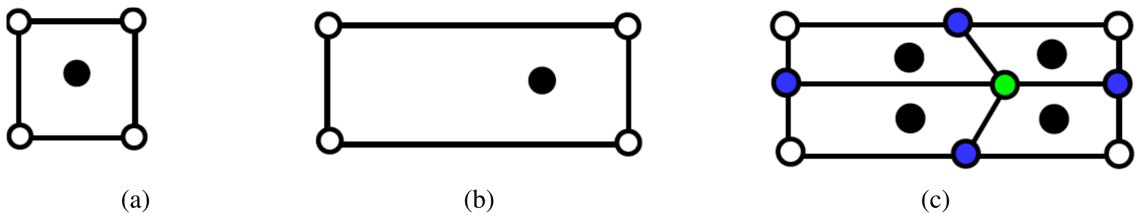


Figure 2.11: Refining a stretched quadrilateral panel. $t = 0$, (a). Panels may stretch due to particle motion (b). By refining stretched panels (c), particles are inserted into the sparse area of domain and a quasi-regular mesh is maintained.

variation in the Lagrangian parameter $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \alpha_3]^T$ is defined as

$$V_\alpha = \sum_{i=1}^{3} \left(\max_{P_j} \alpha_i - \min_{P_j} \alpha_i \right) < \epsilon_2. \tag{2.10}$$

Either V_ζ or V_α is used as criterion 2; if it exceeds ϵ_2, then panel j is refined. Criterion 2 refines developing filaments as the flow evolves.

The variation of the Lagrangian parameter V_α is a generalization of the variation of the vorticity V_ζ. Using V_α returns the focus to the flow map, rather than to the specific problem, and can therefore be used in all applications (advection, barotropic vorticity, shallow water, etc.).

Unfortunately, for problems such as a vortex sheet in three-dimensional space [57] or geophysical problems on the surface of a sphere [184], adaptive refinement alone is not enough to maintain accuracy in Lagrangian simulations. Adaptive refinement is very good at maintaining accuracy on stretched panels (figure 2.11) by inserting particles and panels into regions that have become depleted of computational elements. But adaptive refinement does not do as well when the problem is shearing, not stretching. In extreme cases, sheared panel's interior angles may approach 0 or π; this distortion degrades computations of panel area and introduces error. More frequently, shearing causes error by creating asymmetric interactions between nearby particles. Simply refining a sheared panel does not alleviate these problems (figures 2.12 and 2.13). Sheared panels are common near vorticity gradients, such as the periphery of a jet or along a vorticity front. Examples of

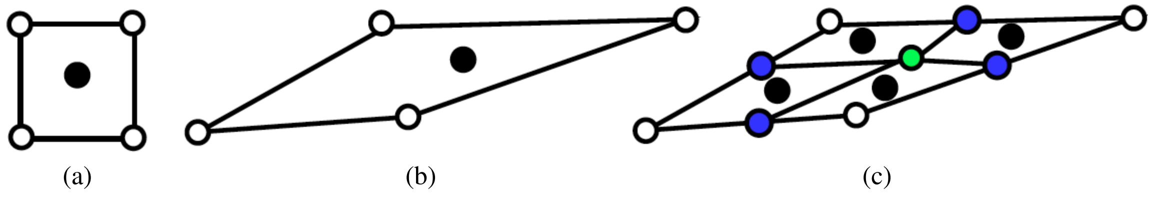


Figure 2.12: Refining a sheared quadrilateral panel. $t = 0$, (a). Panels may shear due to particle motion (b). Inserting particles by refining a sheared panel (c) does not improve the problem of small interior angles or the asymmetric arrangement of particles (see figure 2.13).

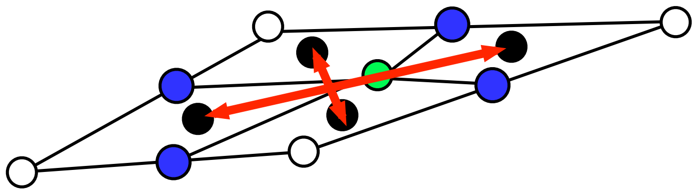


Figure 2.13: Asymmetric interaction of particles caused by shear. The green passive particle should receive similar influence from each of its adjacent active particles (black circles); instead it receives too much influence from its northwest and southeast neighbors (connected by the short red arrows) and not enough from its southwest and northeast neighbors (connected by long red arrows).

mesh distortion in Rossby-Haurwitz wave computations (section 4.2) are shown in figure 2.14 .

One of the optimal properties of a Delaunay triangulation is that it maximizes each triangle's minimum angle [159, 140, 39], which is an ideal property to have for flows that develop sheared panels. A possible strategy to mitigate shearing error would rely on Voronoi diagrams and Delaunay triangulations to optimally reconnect active particles to passive particles as the flow evolves. This approach is similar to the untangling algorithm proposed by [182] and is examined in figure 2.15 and section 4.4. Figure 2.15 shows the evolution of a set of $N = 2562$ Voronoi panels in the same Rossby-Haurwitz wave simulation as figure 2.14. Comparing $t = 0.3$ results from the Voronoi mesh of figure 2.15 (c) to the quadrilateral and triangular meshes in figure 2.14 (b) and (d), we see that while it is still distorted from its initial configuration, the Voronoi mesh appears to have maintained better uniformity than the triangular and quadrilateral panels. However, we find that even these strategies are not sufficient to maintain accuracy in regions of high shear, particularly

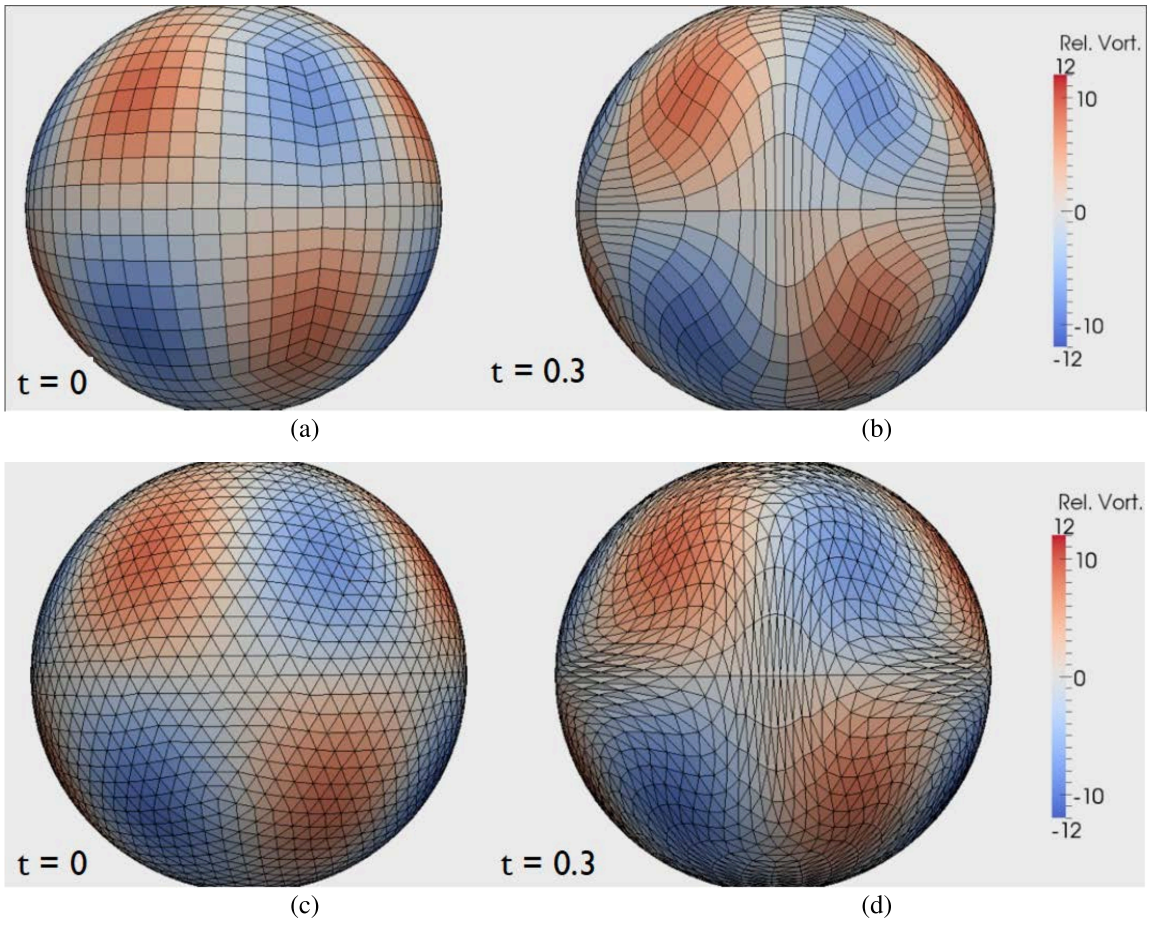


Figure 2.14: Mesh distortion due to a Rossby-Haurwitz wave. Quadrilateral panels (a) and (b) with $N = 1536$. Triangular panels (c) and (d) with $N = 5120$. Color scale shows relative vorticity; red indicates positive vorticity and counterclockwise rotation; blue shows negative vorticity and clockwise rotation.

along Lagrangian manifolds of the flow.

For example, these manifolds are drawn on the sphere in figure 2.15 (a) for the case of a stationary Rossby-Haurwitz wave. In figure 2.15 (d), region A lies along an unstable manifold and has become depleted of fluid particles. Accuracy could be restored to this region using adaptive refinement approaches similar to those employed by [101, 57, 184]. However, region B, which lies along a stable manifold, will also lose accuracy due to the extreme shearing of panels as they approach the saddle point. A close-up view of the saddle point and the Voronoi mesh is shown in figure 2.16. As an additional difficulty associated with shear, as particles approach each other the local mesh spacing approaches zero, which will cause instability with a fixed time step.

Rossby-Haurwitz waves are highly idealized models, but their similarity to realistic flows is precisely the point of Haurwitz's paper [77]. Indeed, saddle points can be observed in the actual atmosphere with regularity; an example is shown in figure 2.17. Although Voronoi grids reduce the effects of shearing distortion compared to panel sets of fixed polygonal types, they do not remove it entirely. Thus, we conclude that we will need to restore regularity to the spatial distribution of fluid particles with a remeshing procedure.

2.4 Remeshing procedures

Particle distortion is an inherent challenge in all Lagrangian methods, and many techniques have been developed to mitigate the spatial accuracy problems that result from distortion. In section 2.3, two such techniques were presented: adaptive refinement, e.g., [101, 57, 184], and reconnecting particles in a way that minimizes panel shearing, e.g., [140, 182]. However, when confronted with the many varieties of distortion error (stretching, shearing, and convergence), we found that these two techniques were inadequate to maintain accuracy in long time calculations. Other Lagrangian methods have confronted

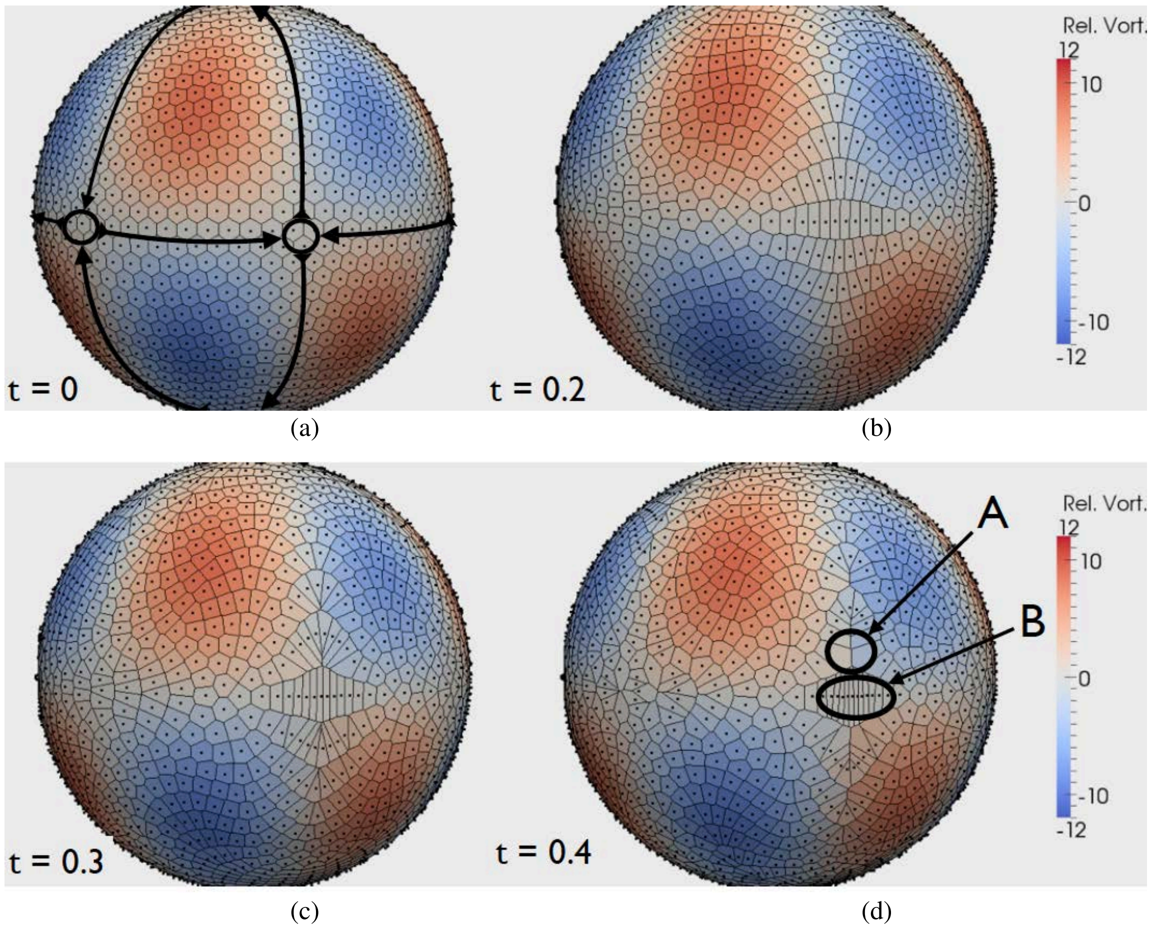


Figure 2.15: Voronoi mesh ($N = 2562$) distortion due to a steady Rossby-Haurwitz wave. Color scale is same as figure 2.14. (a) $t = 0$, Initial mesh. Active particles are black dots; passive particles are panel vertices. Lagrangian manifolds are drawn as thick black curves; saddle points are open circles. (b) $t = 0.2$. Manifolds not drawn to show mesh. (c) $t = 0.3$. (d) $t = 0.4$. Region A has become depleted of particles; region B shows sheared panels approaching a saddle. A close-up view of regions A and B at $t = 0.4$ is shown in figure 2.16.

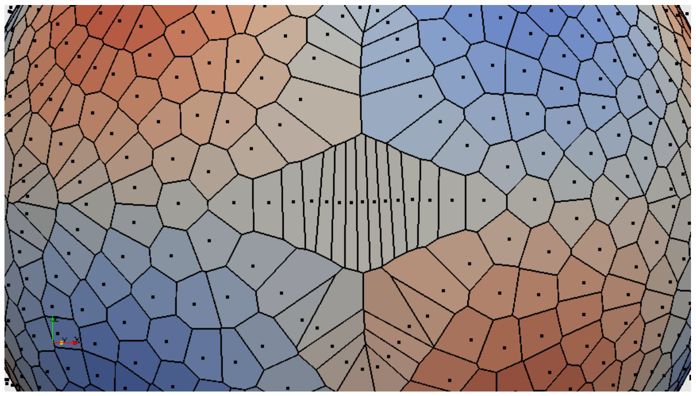


Figure 2.16: Close-up view of saddle point showing regions A and B from figure 2.15 (d).

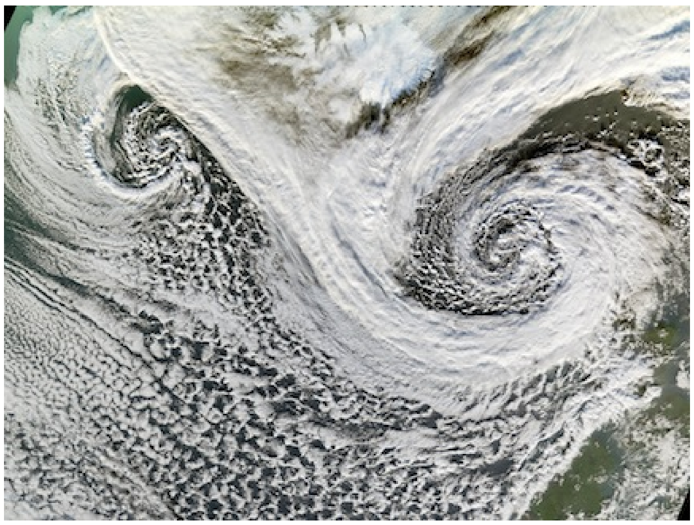


Figure 2.17: MODIS satellite image showing two cyclones south of Iceland (top middle) from November 20, 2006 [3]. Air flows into the pair from the northeast along the cloud band from the upper-right corner toward the center of the image. A nonlinear saddle exists near the center; as the inflowing air approaches it, the flow diverges toward one cyclone or the other.

these difficulties and also concluded that remeshing is necessary to maintain both computational accuracy and efficiency [98, 9, 156]. We adopt the same point of view in this thesis, and develop a new remeshing procedure in this section.

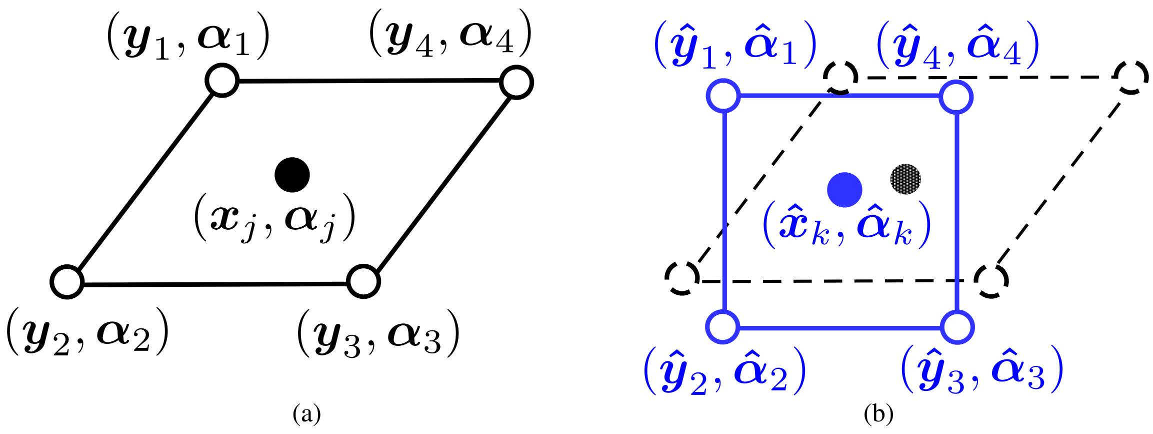
Remeshing is the process of transferring flow information from a distorted set of particles to a new, more regularly distributed set of particles. All remeshing procedures involve interpolation. The method's success depends on the quality of the interpolation scheme, which generally involves scattered source data since the locations of the distorted particles are not known in advance. A brief discussion of scattered data interpolation methods is provided in the appendix. Piecewise polynomials [98, 147], radial basis functions [9, 10], and general interpolation kernels [156] have all been successfully used as interpolation schemes for remeshing Lagrangian computations.

Typically, the interpolated quantities are the problem data– the tracers ϕ in advection problems or vorticity ζ in a barotropic vorticity problem. We refer to this strategy as *direct remeshing*; the data is interpolated directly from a distorted set of particles to a new set.

Depending on the specific details of the interpolation source and destination points, and how often the remeshing procedure is applied during a calculation, the distinction between Lagrangian methods and semi-Lagrangian methods may become blurred. Semi-Lagrangian methods (discussed briefly in section 1.3) perform interpolation at every time step, and do not store Lagrangian quantities separately [166]. Hybrid Eulerian-Lagrangian methods tend to rely more directly on Lagrangian data; for example, the scheme in [75] uses Lagrangian parcels to interpolate a correction term to an Eulerian forecast at each grid point at each time step. Arbitrary Lagrangian-Eulerian (ALE) methods may use moving meshes but frequently do not use the fluid velocity as the mesh velocity [125, 79, 83] and are therefore a different variety of algorithm.

We seek to maintain the Lagrangian nature of our computations by focusing again on the flow map, equation (2.1). For incompressible fluids, the flow map is invertible [71, 120]. In the shallow water equations, which do not have an incompressible horizontal velocity field, it is possible that the flow map may become non-invertible, but such situations correspond either to shock formation or a violation of the shallow water approximation [111, 129]. These conditions do not develop in the context of global circulation models, so we do not expect to face this problem for the types of flows solved by dynamical cores. The flow map's invertibility is the foundation of our remeshing procedure.

The flow map tracks the evolution in time of the correspondence between particles' physical positions $\boldsymbol{x}$ and their Lagrangian parameter $\boldsymbol{\alpha}$. It is the solution of the primary equations we solve, regardless of the specific problem, whether barotropic vorticity, advection, or shallow water. Thus, a remeshing procedure based on preserving the flow map would be versatile enough to apply to each of these equation sets. To begin the development of such a procedure, we reemphasize that the flow map provides a stationary reference frame for materially invariant quantities. This implies that material invariant



$(\boldsymbol{y}_1, \boldsymbol{\alpha}_1)$
$(\boldsymbol{y}_4, \boldsymbol{\alpha}_4)$
$(\boldsymbol{x}_j, \boldsymbol{\alpha}_j)$
$(\boldsymbol{y}_2, \boldsymbol{\alpha}_2)$
$(\boldsymbol{y}_3, \boldsymbol{\alpha}_3)$
(a)
$(\hat{\boldsymbol{y}}_1, \hat{\boldsymbol{\alpha}}_1)$
$(\hat{\boldsymbol{y}}_4, \hat{\boldsymbol{\alpha}}_4)$
$(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{\alpha}}_k)$
$(\hat{\boldsymbol{y}}_2, \hat{\boldsymbol{\alpha}}_2)$
$(\hat{\boldsymbol{y}}_3, \hat{\boldsymbol{\alpha}}_3)$
(b)

Figure 2.18: Lagrangian remeshing. (a) A set of active particles $(\boldsymbol{x}_j, \boldsymbol{\alpha}_j)$ and passive particles $(\boldsymbol{y}_i, \boldsymbol{\alpha}_i)$ have evolved in a flow, and their panels have deformed. (b) A new, regularly distributed mesh of particles $(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{\alpha}}_k)$ & $(\hat{\boldsymbol{y}}_l, \hat{\boldsymbol{\alpha}}_l)$ and panels (blue) is overlaid on the old mesh. The Lagrangian parameter $\boldsymbol{\alpha}$ is interpolated from the old mesh to assign values $\hat{\boldsymbol{\alpha}}$ to the new particles.

quantities, like tracers ϕ in advection problems, absolute vorticity ω in the barotropic vorticity equation, and potential vorticity q in the shallow water equations, are functions only of the Lagrangian parameter and independent of time.

Suppose that a set of fluid particles $\boldsymbol{x}_j(t)$ with Lagrangian parameters $\boldsymbol{\alpha}_j$ have evolved from $t = 0$ to $t = t_{rm}$, where t_{rm} is a remeshing time step, as illustrated by figure 2.18 (a). At $t = t_{rm}$ the time stepping portion of our algorithm is paused, and a new set of particles $\hat{\boldsymbol{x}}_k$ are created, shown in blue in figure 2.18 (b). A direct remeshing algorithm would simply interpolate all problem data from the distorted particles $\boldsymbol{x}_j$ to the new particles $\hat{\boldsymbol{x}}_k$. Instead, we interpolate the Lagrangian parameter $\boldsymbol{\alpha}_j$ from the old mesh to assign new values $\hat{\boldsymbol{\alpha}}_k$ to the new particles. The interpolation scheme uses the old mesh $(\boldsymbol{x}_j, \boldsymbol{\alpha}_j)$ and the physical positions of the new mesh $\hat{\boldsymbol{x}}_k$ to interpolate the Lagrangian parameter $\hat{\boldsymbol{\alpha}}_k$ from the old mesh to the new mesh. Following the interpolation, we renormalize all $\hat{\boldsymbol{\alpha}}_k$ so that each particle satisfies $|\hat{\boldsymbol{\alpha}}_k| = 1$. This procedure preserves, within interpolation error, the numerical representation of the flow map (2.1); we refer to it as *Lagrangian remeshing*.

Lagrangian remeshing inverts the flow map at t_{rm}; each new particle $\hat{\boldsymbol{x}}_k$ now has Lagrangian information $\hat{\boldsymbol{\alpha}}_k$ that relates its current position $\hat{\boldsymbol{x}}_k(t_{rm})$ to the position it would have had at $t = 0$,

$$\hat{\boldsymbol{x}}_k(0) \approx \hat{\boldsymbol{\alpha}}_{\boldsymbol{k}}, \tag{2.11}$$

where $\hat{\boldsymbol{\alpha}}_k$ are the interpolated values of the Lagrangian parameter. These data can then be used to assign values of any materially invariant quantities to the new set of particles. For example, in an advection problem, new particles' tracer values $\hat{\phi}_k$ would be assigned in the following manner:

$$\frac{d}{dt}\phi(\boldsymbol{x}, t) = 0 \implies \phi(\hat{\boldsymbol{x}}_k(t_{rm})) = \phi(\hat{\boldsymbol{x}}_k(0)) \tag{2.12a}$$

$$\hat{\boldsymbol{x}}_k(0) \approx \hat{\boldsymbol{\alpha}}_{\boldsymbol{k}} \tag{2.12b}$$

$$\hat{\phi}_k = \phi(\hat{\boldsymbol{\alpha}}_{\boldsymbol{k}}). \tag{2.12c}$$

Assignment of other material invariants proceeds in the same way. At this point the old particles $\boldsymbol{x}_j$ are discarded, and the new set $\hat{\boldsymbol{x}}_k$ are evolved forward in time until the next remeshing step, when this process is repeated.

The interval between remeshing time steps $R_I = t_{rm} - t_{rm-1}$ must be selected carefully. Choosing too large an R_I will lead to more distortion in the interpolation scheme's source data, which can cause higher error; choosing too small an R_I will result in unnecessary remeshing which can also introduce more interpolation error. Currently we select R_I for a specific problem based on experiments; we hope to develop a more quantitative approach in the future.

The choice of $\boldsymbol{\alpha} = \boldsymbol{x}(\boldsymbol{\alpha}, 0)$ also has implications. This choice of Lagrangian parameter has the advantage of providing analytic formulas for the assignment of materially invariant quantities to particles in equation (2.12c). However, as computations evolve, the distribution of each particle's initial position becomes more complex and develops finer scales.

These fine scales become more difficult to interpolate as computational time increases, particularly on uniform panel sets that are not adaptively refined. A better approach would be to reset the Lagrangian parameter at intermittent times, perhaps at every remeshing step, so that the source data for each interpolation is smoother and easier to use for interpolation. This requires maintaining a numerical approximation of the materially invariant data to use instead of the analytic formula from the initial time in equation (2.12c). We revisit this idea as a topic for future development in chapter VII.

Lagrangian remeshing has several advantages over the direct remeshing procedures typically used in particle methods. The definition of the Lagrangian parameter α guarantees that the source data for the interpolation is smooth, which helps minimize interpolation error (tracer distributions may be discontinuous, which adversely affects direct remeshing procedures). Additionally, interpolation error is not transferred to the flow data; it is kept in the flow map. The result is that there are no overshoots, undershoots, oscillations, or smoothing errors introduced into the flow. Tracer quantities and vorticity distributions therefore develop no new extrema, and current maxima and minima are not dissipated. Chapters IV and V present comparisons of Lagrangian and direct remeshing procedures for a variety of test cases. The Lagrangian remeshing procedure is the last piece we need to build a fluid dynamics solver for the sphere. Our main algorithms are outlined in the next section.

2.5 Dynamical core algorithms

In the previous sections we described the flow map and the discretizations we use for the sphere, the challenge of mesh distortion, and the adaptive refinement and remeshing strategies we use to maintain accuracy. In this section we put those pieces together to develop a dynamical core algorithm, which we refer to as a Lagrangian Particle-Panel

Method, LPPM. The general discrete equations we solve are the particle-panel discretization of the flow map (2.1).

$$\frac{d}{dt}\boldsymbol{x}_j(t) = \boldsymbol{u}(\boldsymbol{x}_j(t), t), \quad j = 1, \ldots, N \tag{2.13a}$$

$$\boldsymbol{x}_j(0) = \boldsymbol{\alpha}_j \tag{2.13b}$$

$$\frac{d}{dt}\boldsymbol{y}_i(t) = \boldsymbol{u}(\boldsymbol{y}_i(t), t), \quad i = 1, \ldots, M \tag{2.13c}$$

$$\boldsymbol{y}_i(0) = \boldsymbol{\alpha}_i \tag{2.13d}$$

where $\boldsymbol{x}_j(t)$ are active particles, N is the number of panels, $\boldsymbol{y}_i(t)$ are passive particles and M is the number of panel vertices. The initial representation of $\mathcal{S}$, icosahedral triangles or the cubed sphere, gives the initial discretization of $\boldsymbol{\alpha}$, which provide the initial conditions (2.13b), (2.13d) for the ODEs (2.13a), (2.13c).

The procedure for solving fluid equations on the sphere with particle-panel discretization and adaptive remeshing is summarized in pseudocode by algorithm II.1. As input, our programs require the desired panel kind (icosahedral triangles or cubed sphere), an initial uniform resolution, the remesh interval R_I, the time step Δt and the final time T. If adaptive refinement is to be used, users must also specify the values of ϵ_1 and ϵ_2 for the refinement criteria (section 2.3). The software takes this input and generates the initial particle-panel sets, and time stepping follows.

Particles and panels are advanced one time step at a time until a remeshing point is reached, that is, until $t = t_{rm}$. At a remeshing step, the remeshing procedure is called and a new mesh is returned. In an adaptive computation, the adaptive refinement subroutines are called again for the new sets of particles and panels. When the remeshing is complete, the program returns to the time stepping loop and continues. Time-dependent data and plotting data can be output at each time step. At the final time, $t = T$, any final data are output and the data structures are deleted to free memory and end the program.

Algorithm II.1: Main Algorithm for LPPM

```
Input     : $n_i$ =initial resolution, $p_k$ =panel kind, $\epsilon_1$, $\epsilon_2$, $\Delta t$, $T$, $R_I$
Output    : Flow simulation (plots and data files)
// Initialize panel set to initial resolution and desired panel kind
[Particles, Edges, Panels ] = GridInit($p_k, n_i$)
if $AMR$ then
    [Particles, Edges, Panels ] = AdaptiveRefine($\epsilon_1$,$\epsilon_2$)
end
// Output initial data
[DataFile,PlotFile ]=Output(Particles,Edges,Panels)
// Time stepping
$t = 0$
while $t \leq T - \Delta t$ do
    if $t = t_{rm}$ then
        [Particles, Edges, Panels ] = Remesh( )
        if $AMR$ then
            [Particles, Edges, Panels ] = AdaptiveRefine($\epsilon_1$,$\epsilon_2$)
        end
    end
    [Particles, Edges, Panels ] = AdvanceOneTimestep($t$, $\Delta t$)
    $t = t + \Delta t$
    // Output intermediate data
    [DataFile,PlotFile ]=Output(Particles,Edges,Panels)
end
// Output final data
[DataFile,PlotFile ]=Output(Particles,Edges,Panels)
// Finalize and clear memory
[Particles, Edges, Panels ] = Delete()
```

The discrete flow map equations (2.13) are integrated in time with fourth order Runge-Kutta in parallel computations using a simple copy algorithm [121] for distributed memory machines. We assume a computing environment with p processors. Each processor maintains a complete copy of the entire system in local memory: all passive particles, edges, and panels. For each stage of Runge-Kutta integration, each process only computes the evolution of N/p active particles and M/p passive particles, then broadcasts their results to the other $p-1$ processes. This approach is also known as data replication, as all problem data is replicated on each process.

Parallel speed up and efficiency results for the barotropic vorticity equation solver are presented in table section 4.1. The copy algorithm is limited by each process's local memory, but even for very high resolution panel sets with $N \sim O\left(10^6\right)$ panels, we have not approached this memory limit. Even at that resolution, our data structures occupy less than 1 GB of memory, which is much lower than today's CPUs have available in DRAM. Other approaches described in [121] can be used when the problem size grows too large to hold in one process's local memory.

Algorithm II.1 and its subroutines were developed with Matlab [126] and are currently implemented in Fortran 90/95 [1, 127] with OpenMPI [141] for parallelization. The source code is compiled with both Intel and GNU compiler suites. Graphs of data are plotted in Matlab. 3D spherical plots are made with the Visualization Toolkit C++ library (VTK) [84]; 2D contour plots are done with NCAR's Command Language (NCL) [180].

2.6 Sources of error

Analyzing error in a Lagrangian method is somewhat different than in an Eulerian method because the solutions of a Lagrangian method also involve the particle positions, not just the tracer concentrations or vorticity. A steady Eulerian solution, for example, a stationary Rossby-Haurwitz wave (section 4.2), is not a steady Lagrangian solution. Error measures typically used to judge accuracy in a dynamical core such as (1.23) do not account for particle position error. To begin a discussion of error associated with algorithm II.1, we list the sources of error that we will encounter:

- Spatial discretization error
- Time discretization error
- Distortion error
- Remeshing error

Spatial discretization error and time discretization error are part of any numerical solution of a partial differential equation, regardless of numerical method. In our case, spatial discretization error refers to errors due to quadrature. This error can be reduced with the use of higher resolution panel sets, and error convergence as $N \to \infty$ is demonstrated for a variety of test problems in the following chapters. Theoretical analysis of the midpoint rule suggests that this error should decay as $O(1/N)$ as $N \to \infty$, or equivalently, as $O\left(\Delta\lambda^2\right)$ as $\Delta\lambda \to 0$, where $\Delta\lambda$ is the average initial edge length of our Lagrangian panels. Time stepping error is error from the time integration, which here is fourth order Runge-Kutta. Time stepping error can be reduced by using smaller Δt; analysis suggests that time stepping error will decay as $O\left(\Delta t^4\right)$ as $\Delta t \to 0$. Numerical convergence with respect to N and Δt for both the barotropic vorticity equation and the advection equation are shown in chapters IV and V, respectively.

Distortion error refers to the distortion that will occur due to particle motion between remeshing steps. This distortion error is due to the approximation of panel edges by great circles (straight lines on the sphere), as shown in figure 2.19. Informally, this error arises from the dA portion of a surface integral $\int_S \phi(\boldsymbol{x})\, dA$, rather than the integrand.

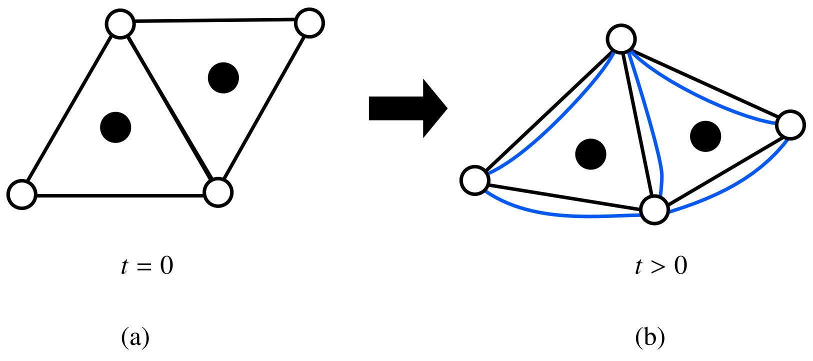


Figure 2.19: Distortion error due to approximating panel edges as great circles. (a) At $t = 0$ the panel edges are great circles. (b) For $t > 0$ continuous edges (blue) do not necessarily lie along the great circles connecting two passive particles (black) which are used for panel area calculations.

In actuality, passive particles are connected by continuous material curves, but our area calculations assume that passive particles are connected by great circles. This approximation is exact at $t = 0$ and each remeshing step, but as particles move the edges connecting them no longer lie along a great circle arc. Great circle edge approximation error is related to the spatial discretization error but is unique to Lagrangian methods. It can be reduced by increasing spatial resolution to shorten panel edges, but it can also be reduced by increasing the number of passive particles per edge, which we discuss later, in chapter V. While adding additional particles to panel edges can decrease distortion error, it also introduces the possibility that panels may become non-convex, which would violate an assumption of our point location query algorithm, figure 2.8. Increasing edge resolution would require a new point query procedure that does not assume convex panels. We have therefore kept the great circle arc approximation and use edges with passive particles only at edge endpoints.

Remeshing error is the most significant source of error in our algorithm, and it is entirely dependent on the quality of the interpolation method used to carry out the remeshing for both direct and Lagrangian remeshing procedures. Initially we used a piecewise linear interpolation scheme on each panel. In this procedure each new particle $\hat{\boldsymbol{x}}_k$ is located within the distorted panel set $\boldsymbol{x}_j$ using the fast point location algorithm, figure 2.8. New Lagrangian parameter data $\hat{\boldsymbol{\alpha}}_k$ were then assigned to particle $\hat{\boldsymbol{x}}_k$ with linear interpolation in barycentric coordinates using data from three points in the distorted panel that contained $\hat{\boldsymbol{x}}_k$. Details are presented in the appendix, section A.2.

We have since upgraded to a cubic Hermite interpolation method provided by the SSRFPACK software library [147]. Remeshing with SSRFPACK begins by building a Delaunay triangulation of the active and passive particles using STRIPACK [146]. SSRFPACK takes the triangulation as input, along with the data to be interpolated. Each new particle $\hat{\boldsymbol{x}}_k$ is located within that triangulation using software provided by the STRIPACK library. Lagrangian data $\hat{\boldsymbol{\alpha}}_k$ are assigned to each new particle using SSRFPACK's cubic Hermite splines on the Delaunay triangulation. Details are presented in [147, 148].

In chapter V we document the improvement in both l_2 and l_∞ error norms due to switching from piecewise linear interpolation to cubic Hermite interpolation for an advection test problem. Other interpolation methods may be used, and some are discussed in the appendix. Although remeshing error is dominated by the interpolation scheme, it is also affected by the remesh interval R_I. Too long an interval will cause a distorted interpolation source data set, while too short an interval will lead to unnecessary interpolation.

In chapters IV and V, we present solutions to a variety of test problems for the barotropic vorticity equation and the advection equation. The four sources of error just listed will be used to interpret our results relative to exact solutions, when they are available. We also use these categories of error to discuss comparisons between our results and results from an Eulerian scheme for the same test problems. Lastly, we also use these categories of error to identify issues that could be changed to improve the accuracy to our fluid equation solver.

CHAPTER III

Barotropic Vorticity Equation

In this chapter we derive the barotropic vorticity equation (BVE) for a rotating sphere and present our numerical method. The solution of the barotropic vorticity equation provides the fluid velocity which we use to integrate the flow map (2.1) from one time step to the next using algorithm II.1. The derivation begins with a discussion of two reference frames which we may use to observe fluid motion: the inertial frame and the rotating frame. The BVE itself is derived by assuming that a thin incompressible fluid of uniform depth lies over a sphere and applying Kelvin's circulation theorem.

We solve the BVE with a stream function in its vorticity-stream formulation [120] using Green's function. When differentiated, Green's function provides the Biot-Savart law for the fluid velocity [19, 96]. Coupling the Biot-Savart law for velocity to the flow map and its discretization from chapter II provides the basis of our numerical method.

At the end of this chapter we describe an Eulerian BVE solver based on the National Center for Atmospheric Research's (NCAR) Community Atmosphere Model Finite Volume (CAM-FV) dynamical core. Results for several test cases from this Eulerian scheme and our algorithm are presented in chapter IV for comparison.

3.1 Background

The point vortex equations for a system of N point vortices on the rotating sphere were originally derived by Bogomolov in 1977 [19]. More recently, Dritschel and Polvani [46] studied these equations using contour dynamics to model roll up of vorticity strips on the sphere. Sakajo [158] uses these equations to model interacting vortex sheets on the sphere. Wang [184] studied the BVE's point vortex approximation in a very similar context to this thesis, but was limited by the increasing effects of mesh distortion at large t. Newton [136, chapter 4], and Newton and Shokraneh [137] use point vortex equations to study the integrability of systems of N point vortices on the sphere from the dynamical systems perspective for the case where N is small, $N \sim \mathcal{O}(10)$, and study the effects of a constant background rotation given by the vector $\mathbf{\Omega}$ on arrangements of N point vortices. They refer to the constant background rotation's effects on the N vortices as one-way coupling; $\mathbf{\Omega}$ affects the vortices, but is unaffected by them. Newton and Sakajo [138] couple a small number, $N \sim \mathcal{O}(10)$, of point vortices to a background vorticity field represented by contours using Dritschel's contour dynamics [47, 46] as a model two-way coupling. In a two-way coupled model, the background rotation of the sphere changes due to the point vortices on the spherical surface, and the sphere's rotation vector is no longer constant.

In this thesis, we assume only one-way coupling between the sphere's rotation vector and the fluid; the sphere's rotation rate remains constant, which is a common simplification in dynamical core applications [80, 183, 68, 108, 123]. We use large $N \sim \mathcal{O}\left(10^4 - 10^6\right)$, typically (larger numbers are possible), to represent the full 2D vorticity distribution as a collection of panels, rather than decomposing it into a collection of 1D contours as in [46, 138]. The panel discretization provides a direct means of assigning vortex strengths (section 2.2) that correspond to geophysical dynamics problems of interest. Specific, predetermined arrangements of point vortices [136, 137] or vortex sheets [158] on the sphere are more difficult to correlate with observed global fluid flows. Wang [184] showed that vortex methods using a particle-panel discretization are capable of solving the BVE on the sphere; this thesis extends her work to dynamical core applications using an improved adaptive refinement strategy (section 2.3) and adding the remeshing procedure from section 2.4.

3.2 Inertial vs. rotating reference frame

The general concept that an Eulerian method corresponds to fixed observer and a Lagrangian method corresponds to an observer moving with the fluid velocity was introduced in section 1.3. In this section we introduce another set of reference frames more specific to geophysical fluid dynamics applications: the inertial reference frame and the rotating reference frame. Like the previous choice of Lagrangian vs. Eulerian formulations, the choice of an inertial frame or a rotating frame will have an effect on the form of the fluid equations, despite the fact that they represent the exact same physics.

Mathematically it is perhaps more natural to use the inertial frame, which refers to a frame of reference centered on an observer at a stationary point in space, away from the Earth's surface. This observation point is fixed relative to the center of the Earth, so the planetary orbit around the sun is not seen, but the Earth's spin is. An inertial observer can see the Earth rotate and fluid flow over the planet surface. In this reference frame, Newton's second law applies directly, and the equations of motion are derived in a straightforward manner [30, 12, 183].

From a practical point of view however, it is often more convenient to use a rotating frame, which is a frame of reference centered at a specific point on the Earth's equator, rotating in unison with the planet itself. Geostationary weather satellites and television meteorologists display data in a rotating frame; they show clouds and weather patterns traveling over fixed geography. In the rotating frame we do not directly observe the fact that the planet is spinning, but the effects of that spin on fluid motion are of course still present. These effects show up as source terms in the rotating frame equations of motion. The derivations below follow the presentation of Vallis [183, section 2.1.2]; for additional discussion, see [123, 80, 12, 68].

Suppose we view $\mathbb{R}^3$ from an inertial reference frame, and that the vectors $\boldsymbol{b} \in \mathbb{R}^3$ we observe are subject to a background rotation of constant angular velocity given by the vector $\boldsymbol{\Omega}$. Let us choose a coordinate system whose z-axis is aligned with the axis of rotation. Then $\boldsymbol{\Omega} = [0, 0, \Omega]^{\mathrm{T}}$, where Ω is the angular velocity of the background rotation. We can decompose the rate of change of $\boldsymbol{b}$ into two components,

$$\left(\frac{d\boldsymbol{b}}{dt}\right)_I = \left(\frac{d\boldsymbol{b}}{dt}\right)_R + \boldsymbol{\Omega} \times \boldsymbol{b}, \tag{3.1}$$

where the subscript I denotes the rate of change of $\boldsymbol{b}$ in the inertial frame, and the subscript R denotes the rate of change in the rotating frame; the cross product $\boldsymbol{\Omega} \times \boldsymbol{b}$ is the change in $\boldsymbol{b}$ due to the background rotation. Let the vector $\boldsymbol{r}$ represent position; its derivative is velocity. Let $\boldsymbol{v}_I$ represent velocity measured in the inertial frame and $\boldsymbol{v}_R$ represent velocity measured in the rotating frame. Then (3.1) becomes

$$\left(\frac{d\boldsymbol{r}}{dt}\right)_I = \left(\frac{d\boldsymbol{r}}{dt}\right)_R + \boldsymbol{\Omega} \times \boldsymbol{r} \tag{3.2a}$$

$$\boldsymbol{v}_I = \boldsymbol{v}_R + \boldsymbol{\Omega} \times \boldsymbol{r}. \tag{3.2b}$$

To discuss vorticity, we find the curl of (3.2), which leads to two definitions.

Definition III.1. *Absolute vorticity*, $\boldsymbol{\omega}$, is the curl of the inertial velocity.

$$\boldsymbol{\omega} = \nabla \times \boldsymbol{v}_I \tag{3.3}$$

Definition III.2. *Relative vorticity*, $\boldsymbol{\zeta}$, is the curl of the velocity in the rotating frame.

$$\boldsymbol{\zeta} = \nabla \times \boldsymbol{v}_R \tag{3.4}$$

The absolute and relative vorticities are related by the following equation,

$$\boldsymbol{\omega} = \nabla \times [\boldsymbol{v}_R + \boldsymbol{\Omega} \times \boldsymbol{r}] \tag{3.5a}$$

$$\boldsymbol{\omega} = \boldsymbol{\zeta} + 2\boldsymbol{\Omega}, \tag{3.5b}$$

where we have used the fact that $\nabla \times (\boldsymbol{\Omega} \times \boldsymbol{r}) = 2\boldsymbol{\Omega}$. The vector $2\boldsymbol{\Omega}$ is often referred to as *planetary vorticity*, as it is the vorticity imparted to a particle due to the Earth's rotation. Equation (3.5b) states that absolute vorticity $\boldsymbol{\omega}$ is the sum of relative vorticity $\boldsymbol{\zeta}$ and planetary vorticity $2\boldsymbol{\Omega}$.

In 2D models of a thin layer of fluid, the vorticity vector is orthogonal to the velocity vector, and the vectors given by equations (3.3) and (3.4) have nonzero components only in the normal direction. For notational convenience, let the scalars $\omega = \boldsymbol{\omega} \cdot \hat{\boldsymbol{n}}$ and $\zeta = \boldsymbol{\zeta} \cdot \hat{\boldsymbol{n}}$ denote the radial (normal) components of the absolute and relative vorticity vectors, respectively. These scalars vorticity variables will feature prominently throughout the rest of this thesis, and this scalar definition of relative vorticity is common in much of the meteorological and oceanographic literature.

Equations (3.2) and (3.5) relate the inertial and rotating frames mathematically. This relation is fundamental to the equations of motion that follow, for, even as we choose the rotating frame of reference where the Earth's rotation is not directly apparent, we must still account for the effects of that rotation.

3.3 Barotropic fluids

We consider a thin layer of fluid of constant density and uniform depth characterized by incompressible, non-divergent flow, and derive the equations of motion following from these assumptions [183, section 4.3.4]. In such a fluid, Kelvin's circulation theorem tells us that the circulation about any material element is constant, [12, 183, 30, 120]

$$\frac{d}{dt}\oint_C \boldsymbol{v}_I \cdot d\boldsymbol{r} = \frac{d}{dt}\oint_C (\boldsymbol{v}_R + \boldsymbol{\Omega} \times \boldsymbol{r}) \cdot d\boldsymbol{r} = 0, \tag{3.6}$$

where C is a closed curve following the flow so that no particles cross C as the flow evolves. Next we apply Stokes' theorem to the region D enclosed by C,

$$\frac{d}{dt}\int_D \boldsymbol{\omega} \cdot \hat{\boldsymbol{n}} dA = \frac{d}{dt}\int_D (\boldsymbol{\zeta} + 2\boldsymbol{\Omega}) \cdot \hat{\boldsymbol{n}} dA = 0. \tag{3.7}$$

Since this must hold for all $D \subseteq \mathcal{S}$, upon bringing the derivative into the integral using Reynolds's transport theorem [30, chapter 1], we find that

$$\frac{d}{dt}(\boldsymbol{\omega} \cdot \hat{\boldsymbol{n}}) = \frac{d}{dt}((\boldsymbol{\zeta} + 2\boldsymbol{\Omega}) \cdot \hat{\boldsymbol{n}}) = 0. \tag{3.8}$$

The unit normal $\hat{\boldsymbol{n}}$ is equivalent to the position vector on $\mathcal{S}$, and in spherical coordinates is

$$\hat{\boldsymbol{n}}(\lambda, \theta) = [\cos\theta\cos\lambda, \cos\theta\sin\lambda, \sin\theta]^{\mathrm{T}},$$

where θ and λ are latitude and longitude. For a coordinate system rotating about its z-axis, $2\boldsymbol{\Omega} \cdot \hat{\boldsymbol{n}} = 2\Omega\sin\theta$, since $\boldsymbol{\Omega} = [0, 0, \Omega]^{\mathrm{T}}$, which allows us to write (3.8) as

$$\frac{d}{dt}(\omega \cdot \hat{\boldsymbol{n}}) = \frac{d}{dt}(\boldsymbol{\zeta} \cdot \hat{\boldsymbol{n}} + 2\Omega\sin\theta) = 0. \tag{3.9}$$

Equation (3.9) states that the normal component of absolute vorticity is conserved materially. The quantity $f = 2\Omega\sin\theta$ is known as the Coriolis parameter. The significance of (3.9) is that the absolute vorticity $\omega = \zeta + f$ is constant with respect to time in the Lagrangian frame.

In fact, equation (3.9) is the barotropic vorticity equation; it simply states that a thin layer of incompressible fluid with constant density conserves vorticity materially:

$$\frac{d\omega}{dt} = \frac{d}{dt}(\zeta + f) = 0. \tag{3.10}$$

Properties of barotropic fluids and their relation to global atmospheric circulation are summarized in [52, section 5.2]. As a model of the atmosphere, a barotropic fluid captures many characteristics of realistic large-scale flow and performs well as an introductory model. As discussed in section 1.1, it was also the equation used for the first successful computational weather forecasts [29]. In a Lagrangian setting, equation (3.10) by itself is not enough to allow us solve for the motion of fluid particles in a computational environment. It has no mention of particle positions $\boldsymbol{x}$, which are the solutions of a PDE in Lagrangian form. To find the Lagrangian form of the BVE, we couple (3.10) to the flow map (2.1).

3.4 Lagrangian form

Beginning with the flow map (2.1), which is the foundation of any Lagrangian method, we assume that a particle in the fluid at $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\alpha}, t)$ has velocity given by the fluid's velocity vector at that position, $d\boldsymbol{x}/dt = \boldsymbol{u}(\boldsymbol{x}(\boldsymbol{\alpha}, t), t)$. To find the velocity $\boldsymbol{u}$ for an incompressible fluid, we use the stream function for the flow.

Definition III.3. A *stream function* $\psi = \psi(\boldsymbol{x}, t) = \psi(\lambda, \theta, t)$ is a scalar function defined on $\mathcal{S}$ such that the velocity $\boldsymbol{u}$ satisfies

$$\boldsymbol{u}(\boldsymbol{x}, t) = \nabla\psi(\boldsymbol{x}, t) \times \boldsymbol{x}, \tag{3.11a}$$

where ∇ is the gradient operator on $\mathcal{S}$ given by (1.14). In spherical coordinates, the velocity vector is

$$\boldsymbol{u} = \frac{\partial\psi}{\partial\theta}\hat{\boldsymbol{e}}_\lambda - \frac{1}{\cos\theta}\frac{\partial\psi}{\partial\lambda}\hat{\boldsymbol{e}}_\theta. \tag{3.11b}$$

Applying the curl operator to (3.11b),

$$\nabla \times \boldsymbol{u} = \left(\frac{1}{\cos^2\theta}\frac{\partial^2\psi}{\partial\lambda^2} + \frac{\partial^2\psi}{\partial\theta^2} - \tan\theta\frac{\partial\psi}{\partial\theta}\right)\hat{\boldsymbol{e}}_r, \tag{3.12}$$

we find an equation that relates the stream function to vorticity,

$$(\nabla \times \boldsymbol{u}) \cdot \hat{\boldsymbol{e}}_r = -\nabla^2 \psi. \tag{3.13}$$

Note that the quantity $(\nabla \times \boldsymbol{u}) \cdot \hat{\boldsymbol{e}}_{\boldsymbol{r}}$ is the normal component of either absolute vorticity, ω, or the normal component of relative vorticity, ζ, depending on whether $\boldsymbol{u}$ represents velocity in the inertial or rotating frame. Let ψ_I and ψ_R denote the stream function for the inertial and rotating frames. Then (3.13) becomes either one of the following equations

$$-\nabla^2 \psi_I = \omega \tag{3.14a}$$

$$-\nabla^2 \psi_R = \zeta \tag{3.14b}$$

Unfortunately there is some ambiguity about the precise definition of the stream function. The above definition corresponds to the typical presentation in mathematical texts, even those that relate to geophysical applications *i.e.*, [30, 136, 123], but most textbooks that primarily focus on meteorology or oceanography *i.e.,* [68, 80, 183], define the stream function with the equation

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{x} \times \nabla \psi(\boldsymbol{x}, t) \tag{3.15}$$

instead of (3.11a). This alternative definition has the effect of placing the negative sign on the longitudinal $\hat{\boldsymbol{e}}_\lambda$ component of (3.11b) instead the latitudinal $\hat{\boldsymbol{e}}_\theta$ component and removes the negative signs from (3.13) and (3.14) altogether. We have elected to use the mathematical definition (3.11) in this thesis.

For incompressible fluids, such as those represented by the barotropic vorticity equation, the stream function completely describes the flow through the above equations [30]. Given a vorticity distribution ω or ζ, we can determine the stream function ψ_I or ψ_R by solving the Poisson problem (3.14) and subsequently recover the velocity from the stream function using (3.11a).

We solve the Poisson problem using Green's function for the Laplace-Beltrami operator on the unit sphere $\mathcal{S}$. The Green's function for $\mathcal{S}$ was derived by Bogomolov in 1977 [19] from the Green's function for the 3D Laplacian on the Euclidean space $\mathbb{R}^3$ and the method of images [165]. Bogomolov considered a barotropic fluid contained between two concentric spherical boundaries, one with radius a and the other with radius $a + \epsilon$ in the limit $\epsilon \to 0$. Ten years later, Kimura & Okamoto [96] derived the Green's function for the sphere by considering the 2D free space Green's function on the periodic latitude-longitude plane, and showed that their result is equivalent to Bogomolov's. We state the results of Bogomolov and Kimura & Okamoto as a definition.

Definition III.4. The Poisson problem on the unit sphere $\mathcal{S}$

$$\nabla^2 u(\boldsymbol{x}) = -f(\boldsymbol{x}) \tag{3.16}$$

with $\int_{\mathcal{S}} f(\boldsymbol{x}) = 0$ has the *Green's function*, or fundamental solution,

$$G(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = -\frac{1}{4\pi} \log(1 - \boldsymbol{x} \cdot \tilde{\boldsymbol{x}}) \tag{3.17a}$$

$$G(\lambda, \theta, \tilde{\lambda}, \tilde{\theta}) = -\frac{1}{4\pi} \log\left(1 - \cos\gamma(\lambda, \theta, \tilde{\lambda}, \tilde{\theta})\right) \tag{3.17b}$$

where $\cos\gamma$ is the cosine of the central angle between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ given by (1.16). Green's function represents the response at $\boldsymbol{x} \mapsto (\lambda, \theta)$ due to a unit source located at $\tilde{\boldsymbol{x}} \mapsto (\tilde{\lambda}, \tilde{\theta})$; it satisfies the following equation on $\mathcal{S}$ in the sense of distributions [96, 184].

$$-\nabla^2 G(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \delta(\boldsymbol{x} - \tilde{\boldsymbol{x}}) - \frac{1}{4\pi} \tag{3.18}$$

The constant in (3.18) is due to the periodicity of the sphere, and ensures that

$$\int_{\mathcal{S}} G(\boldsymbol{x}, \tilde{\boldsymbol{x}})\, dA = 0, \tag{3.19}$$

which is necessary for a solution of Poisson problem (3.18) to exist.

The stream function, which is the solution of (3.14), is

$$\psi_I(\boldsymbol{x}) = \int_S G(\boldsymbol{x},\tilde{\boldsymbol{x}})\omega(\tilde{\boldsymbol{x}})\,dA(\tilde{\boldsymbol{x}}) \tag{3.20a}$$

$$\psi_R(\boldsymbol{x}) = \int_S G(\boldsymbol{x},\tilde{\boldsymbol{x}})\zeta(\tilde{\boldsymbol{x}})\,dA(\tilde{\boldsymbol{x}}). \tag{3.20b}$$

We find the velocity field given by a Biot-Savart law,

$$\boldsymbol{u}_I(\boldsymbol{x}) = \int_S \boldsymbol{K}(\boldsymbol{x},\tilde{\boldsymbol{x}})\omega(\tilde{\boldsymbol{x}})\,dA(\tilde{\boldsymbol{x}}) \tag{3.21a}$$

$$\boldsymbol{u}_R(\boldsymbol{x}) = \int_S \boldsymbol{K}(\boldsymbol{x},\tilde{\boldsymbol{x}})\zeta(\tilde{\boldsymbol{x}})\,dA(\tilde{\boldsymbol{x}}), \tag{3.21b}$$

where the integral kernel

$$\boldsymbol{K}(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \frac{-\boldsymbol{x}\times\tilde{\boldsymbol{x}}}{4\pi(1-\boldsymbol{x}\cdot\tilde{\boldsymbol{x}})} \tag{3.22}$$

is the result of applying the $\nabla(\,\cdot\,)\times\boldsymbol{x}$ operator (3.11a) to (3.20). The derivatives act with respect to $\boldsymbol{x}$ on the kernel $G(\boldsymbol{x},\tilde{\boldsymbol{x}})$ to give $\boldsymbol{K}(\boldsymbol{x},\tilde{\boldsymbol{x}})$.

3.4.1 Inertial frame

In the inertial frame we use (3.21a) to solve for the velocity in terms of the absolute vorticity ω to solve the barotropic vorticity equation,

$$\nabla^2\psi_I = -\omega \tag{3.23a}$$

$$\psi_I(\boldsymbol{x},t) = -\frac{1}{4\pi}\int_S \log(1-\boldsymbol{x}\cdot\tilde{\boldsymbol{x}})\omega(\tilde{\boldsymbol{x}})\,dA(\tilde{\boldsymbol{x}}) \tag{3.23b}$$

$$\boldsymbol{u}_I(\boldsymbol{x},t) = -\frac{1}{4\pi}\int_{S^2} \frac{\boldsymbol{x}\times\tilde{\boldsymbol{x}}}{1-\boldsymbol{x}\cdot\tilde{\boldsymbol{x}}}\omega(\tilde{\boldsymbol{x}})\,dA(\tilde{\boldsymbol{x}}) \tag{3.23c}$$

$$\frac{d\omega}{dt} = 0. \tag{3.23d}$$

A Lagrangian method for (3.23) would couple the flow map (2.1) to (3.23c). Equation (3.23d) is satisfied implicitly in the Lagrangian formulation. The inertial frame and the above equations are similar to the equations studied by [137]. Our method uses the rotating reference frame, however, in order to more closely align with atmospheric modeling applications and existing dynamical cores.

3.4.2 Rotating frame

In the rotating frame, the equations are represented in terms of the relative vorticity, ζ, and we use (3.21b) for the velocity,

$$\nabla^2 \psi_R = -\zeta \tag{3.24a}$$

$$\psi_R(\boldsymbol{x}, t) = -\frac{1}{4\pi} \int_S \log(1 - \boldsymbol{x} \cdot \tilde{\boldsymbol{x}}) \zeta(\tilde{\boldsymbol{x}}) \, dA(\tilde{\boldsymbol{x}}) \tag{3.24b}$$

$$\boldsymbol{u}_R(\boldsymbol{x}, t) = -\frac{1}{4\pi} \int_S \frac{\boldsymbol{x} \times \tilde{\boldsymbol{x}}}{1 - \boldsymbol{x} \cdot \tilde{\boldsymbol{x}}} \zeta(\tilde{\boldsymbol{x}}) \, dA(\tilde{\boldsymbol{x}}) \tag{3.24c}$$

$$\frac{d\zeta(\boldsymbol{x})}{dt} = -2\Omega \frac{dz}{dt}. \tag{3.24d}$$

Equation (3.24d) is simply the equation $d/dt(\zeta + f) = 0$; the right-hand side of (3.24d) is the material derivative of the Coriolis parameter f. Since the coordinate system's z-axis is aligned with the axis of rotation, only the z component of particle motion contributes to the Coriolis parameter's derivative. Equation (3.24d) shows the effect of the background rotation: relative vorticity is modified by the advection of planetary vorticity.

To derive a Lagrangian method for the barotropic vorticity equation, we couple the flow map (2.1) to (3.24). Except for the special case of the solid body rotation test, all of our work is done in the rotating frame; for convenience in the remainder of this thesis we drop the subscript R. The equations we solve numerically with (2.13) are

$$\frac{d}{dt}\boldsymbol{x}(\boldsymbol{\alpha}, t) = -\frac{1}{4\pi} \int_S \frac{\boldsymbol{x}(\boldsymbol{\alpha}, t) \times \boldsymbol{x}(\tilde{\boldsymbol{\alpha}}, t)}{1 - \boldsymbol{x}(\boldsymbol{\alpha}, t) \cdot \boldsymbol{x}(\tilde{\boldsymbol{\alpha}}, t)} \zeta(\boldsymbol{x}(\tilde{\boldsymbol{\alpha}}, t)) \, dA(\boldsymbol{x}(\tilde{\boldsymbol{\alpha}}, t)) \tag{3.25a}$$

$$\boldsymbol{x}(\boldsymbol{\alpha}, 0) = \boldsymbol{\alpha} \tag{3.25b}$$

$$\frac{d}{dt}\zeta(\boldsymbol{x}(\boldsymbol{\alpha}, t)) = -2\Omega \frac{d}{dt} z(\boldsymbol{\alpha}, t). \tag{3.25c}$$

Equations (3.25) are the Lagrangian form of the BVE on a rotating sphere in the rotating reference frame.

3.5 Numerical method

We use a collection of particles and panels to discretize $\mathcal{S}$, as discussed in chapter II. We use Cartesian coordinates to avoid problems with the pole singularities in spherical coordinates; as noted by [136], this choice of coordinates also makes the Green's function simpler to compute.

At $t = 0$, we initialize a grid of N panels and M vertices using either the cubed sphere or icosahedral triangles. To discretize the flow map $\boldsymbol{x}(\boldsymbol{\alpha}, t)$, as in section 2.2, we let $\boldsymbol{x}_j(t) = \boldsymbol{x}(\boldsymbol{\alpha}_j, t)$ represent the active particles at panel centers, and $\boldsymbol{y}_i(t) = \boldsymbol{x}(\boldsymbol{\alpha}_i, t)$ represent the passive particles at panel vertices. Each particle is assigned a value of relative vorticity, $\zeta(\boldsymbol{x}) = \omega(\boldsymbol{x}) - 2\Omega z$, where $\omega : \mathcal{S} \mapsto \mathbb{R}$ is an absolute vorticity distribution associated with a specific test problem. In more advanced work, vorticity distributions could be derived from measured data. Active particles are also assigned an area A_j that represents the area of their associated panels using the procedures described in section 2.2.

For $t > 0$, the equations we solve are

$$\frac{d\boldsymbol{x}_j}{dt} = -\frac{1}{4\pi} \sum_{\substack{k=1 \\ k \neq j}}^{N} \frac{\boldsymbol{x}_j \times \boldsymbol{x}_k}{1 - \boldsymbol{x}_j \cdot \boldsymbol{x}_k} \zeta_k A_k, \quad j = 1, \ldots, N \tag{3.26a}$$

$$\frac{d\boldsymbol{y}_i}{dt} = -\frac{1}{4\pi} \sum_{k=1}^{N} \frac{\boldsymbol{y}_i \times \boldsymbol{x}_k}{1 - \boldsymbol{y}_i \cdot \boldsymbol{x}_k} \zeta_k A_k, \quad i = 1, \ldots, M \tag{3.26b}$$

$$\frac{d\zeta_{i,j}}{dt} = -2\Omega \frac{dz_{i,j}}{dt} \tag{3.26c}$$

These sums represent midpoint rule quadrature applied to the Biot-Savart integral in (3.25). For active particles we skip the singular panel which gives the $k \neq j$ term in the active particles' index. Passive particles stay well-separated from active particles due to the fluid's incompressibility, so there is no need to skip any panels in the passive particles' sum.

Equations (3.26) are integrated with fourth order Runge-Kutta. A high order time integration method ensures that time discretization error will be much smaller than the other sources of error discussed in section 2.6.

3.5.1 Interpretation

The discrete equations for the barotropic vorticity equation (3.26) are also the equations for a system of N point vortices on a rotating sphere, and identical to the equations considered by Bogomolov [19, section 3]. Finite collections of point vortices are the subject of vortex methods, and the interpretation and theoretical results from that field apply to this thesis. Point vortices derive their name from an interpretation of equation (3.18). Without regularization the Green's function's logarithmic singularity becomes a delta function after the Laplace-Beltrami operator is applied, and we may view the vorticity due to a source at $\tilde{\boldsymbol{x}}$ as a singular, concentrated source of vorticity, or *point vortex*. The process of assigning vorticity strengths by sampling vorticity distributions amounts to an approximation of that distribution by a superposition of N point vortices [145, 136, 38, 97].

$$\zeta(\boldsymbol{x}) \approx \sum_{k=1}^{N} \Gamma_k \delta(\boldsymbol{x} - \boldsymbol{x}_k) \tag{3.27}$$

where the weights $\Gamma_k = \zeta_k A_k$ are defined by (2.5b).

While this is a useful interpretation, we cannot use equation (3.27) numerically; $\delta(0)$ is infinite and skipping the singular panel at $\boldsymbol{x} = \boldsymbol{x}_k$ would completely remove the kth point vortex from the calculation. It is possible to use mollifiers, or smooth approximations of delta functions, in order to calculate an approximation of equation (3.27); this idea leads to vortex blob methods [38, 97, 120]. We investigate regularized delta kernels as an interpolation scheme in the appendix, section A.1, but smoothed integral kernels are not necessary to solve the barotropic vorticity equation since we do not need to compute the vorticity field directly to solve the BVE. We only need the fluid velocity. The singularity in the Biot-Savart integral (3.25) is not as strong as a delta function, and we can compute velocity from an approximation of it by simply skipping singular panels in (3.26). Thus, our method can be interpreted as a point vortex method as it does not regularize the Green's function or the Biot-Savart integral.

Aside from the interpretation of representing a vorticity field with singular fluid particles, vortex methods also offer insight into the conservation properties of the numerical method. In particular, we note that finite collections of N point vortices like (3.26) are Hamiltonian systems. Newton [136] explores the consequences of this fact in detail for the planar geometries. Bogomolov [19] finds the Hamiltonian and other integral invariants for the spherical case, and relates them to their planar analogues.

The existence of integral invariants in the discrete equations are a primary feature of vortex methods, and is responsible for their conservation properties as a numerical method. Majda and Bertozzi [120, corollary 1.4] list several integral invariants of a barotropic fluid's continuous equations. We focus on three.

$$V = \int_{\mathcal{S}} \zeta(\boldsymbol{x})\, dA \tag{3.28a}$$

$$KE = \frac{1}{2}\int_{\mathcal{S}} |\boldsymbol{u}(\boldsymbol{x})|^2\, dA \tag{3.28b}$$

$$Z = \frac{1}{2}\int_{\mathcal{S}} \zeta(\boldsymbol{x})^2\, dA \tag{3.28c}$$

V in equation (3.28a) defines the total vorticity. For the periodic domain $\mathcal{S}$, it should be identically zero. Equation (3.28b) defines KE as the total kinetic energy of the flow, and Z in equation (3.28c) is the total enstrophy. Each of these equations define conserved quantities of the continuous barotropic vorticity equation; these quantities are derived from the physics of a barotropic fluid and do not depend on the method used to solve the equation. In addition to the error norms introduced in section 1.5.3, we also evaluate the success of our method by measuring how well the quantities defined by (3.28) are maintained during a computation. The Hamiltonian properties of the discrete system (3.26) suggest that our numerical method should conserve integral invariants (3.28) well; we investigate this hypothesis in the test cases in chapter IV.

3.6 For comparison: An Eulerian model

We present an Eulerian method for solving the BVE in this section, based on techniques used by the NCAR CAM-FV dynamical core. The Eulerian form of the BVE is

$$\nabla^2\psi(\lambda,\theta,t) = -\zeta(\lambda,\theta,t) \tag{3.29a}$$

$$\boldsymbol{u}(\lambda,\theta,t) = \frac{\partial\psi}{\partial\theta}\hat{\boldsymbol{e}}_\lambda - \frac{1}{\cos\theta}\frac{\partial\psi}{\partial\lambda}\hat{\boldsymbol{e}}_\theta \tag{3.29b}$$

$$\frac{\partial\omega}{\partial t} + \boldsymbol{u}\cdot\nabla\omega = 0 \tag{3.29c}$$

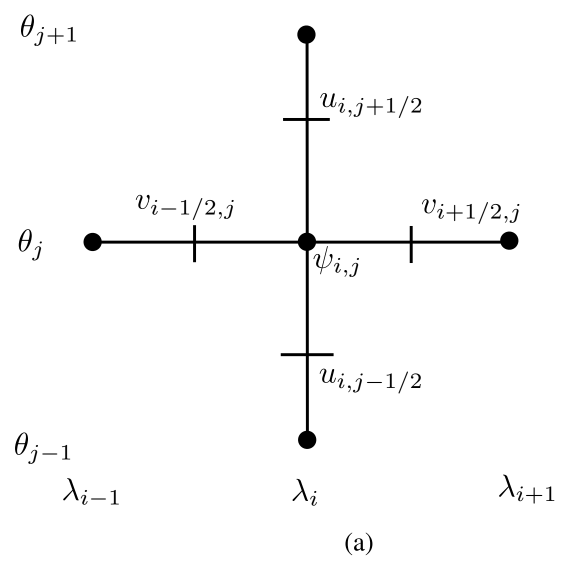
An Eulerian BVE solver has two components: an advection scheme and a Poisson solver. The Poisson solver finds the stream function from the current vorticity field. The stream function is differentiated to give velocity $\boldsymbol{u}$, and the advection scheme uses that velocity to advect the vorticity to the next time step, for input to the next Poisson problem. This cycle repeats until the computation is finished.

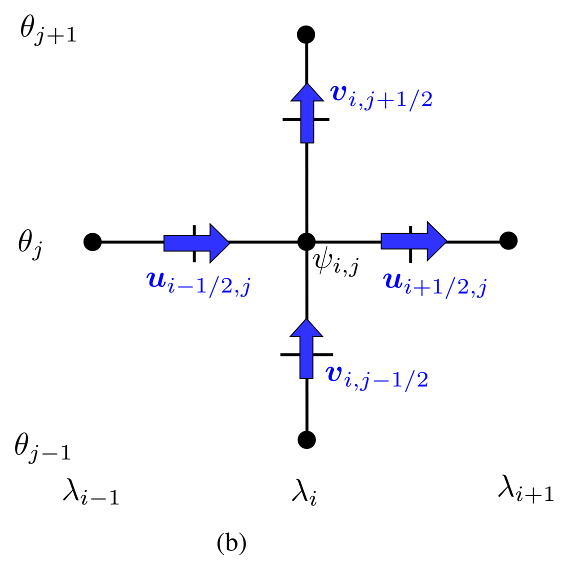
For the Poisson solver, the stream function is stored on a uniform latitude-longitude grid in a 2D array so that $\psi_{i,j}$ corresponds to the value at longitude λ_i and latitude θ_j, where

$$\lambda_i = (i-1)\Delta\lambda, \quad i = 1,\dots,2N \qquad \theta_j = -\frac{\pi}{2} + (j-1)\Delta\theta, \quad j = 1,\dots,N+1 \tag{3.30}$$

and $\Delta\lambda = \Delta\theta = \pi/N$. We use a simple 5-point stencil for the Laplacian operator (figure 3.1 (a)), and solve the system of equations given by that discretization for the stream function ψ,

$$\frac{1}{\Delta\theta^2}\left(\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}\right) - \frac{\tan\theta_j}{2\Delta\theta}\left(\psi_{i,j+1} - \psi_{i,j-1}\right) + \frac{1}{\cos^2\theta_j\Delta\lambda^2}\left(\psi_{i-1,j} - 2\psi_{i,j} + \psi_{i+1,j}\right) = -\zeta_{i,j}. \tag{3.31}$$





θ_{j+1}
$\boldsymbol{v}_{i,j+1/2}$
θ_j
$\psi_{i,j}$
$\boldsymbol{u}_{i-1/2,j}$
$\boldsymbol{u}_{i+1/2,j}$
$\boldsymbol{v}_{i,j-1/2}$
θ_{j-1}
λ_{i-1}
λ_i
λ_{i+1}
(b)

Figure 3.1: (a) 5 point stencil for the Laplacian on a latitude-longitude grid. (b) Flux velocities (blue) are interpolated from the Poisson solver to a C-grid for the advection scheme.

Both values of absolute vorticity ω and the stream function ψ are maintained at the cell centers (black circles). Relative vorticity is the difference between the absolute vorticity and the Coriolis parameter at each grid point,

$$\zeta_{i,j} = \omega_{i,j} - f_{i,j}, \qquad f_{i,j} = 2\Omega \sin\theta_j. \tag{3.32}$$

Equation (3.31) is solved for ψ using the Gauss-Seidel iterative method [112, chapter 4]. The $k+1$ iterate $\psi^{(k+1)}$ is found from the previous iterate $\psi^{(k)}$ using

$$\begin{aligned}\psi_{i,j}^{(k+1)} = \frac{\Delta\lambda^2 \zeta_{i,j}}{2(2+\tan^2\theta_j)} - \frac{\tan\theta_j \Delta\lambda}{4(2+\tan^2\theta_j)}\left(\psi_{i,j+1}^{(k)} - \psi_{i,j-1}^{(k+1)}\right) \\ + (2+\tan^2\theta_j)\left(\psi_{i,j+1}^{(k)} + \psi_{i,j-1}^{(k+1)}\right) + \frac{1}{1+\cos^2\theta_j}\left(\psi_{i+1,j}^{(k)} + \psi_{i-1,j}^{(k+1)}\right).\end{aligned} \tag{3.33}$$

Once (3.33) has converged to a user specified accuracy tolerance, the computed stream function values $\psi_{i,j}$ are differentiated with centered finite differences applied to equation (3.11b) to give the zonal and meridional components of fluid velocity, u and v, respectively.

Velocity values are stored at cell interfaces, so that

$$u_{i,j+1/2} = \frac{1}{\Delta\theta}\left(\psi_{i,j+1} - \psi_{i,j}\right) \tag{3.34a}$$

$$v_{i+1/1,j} = \frac{-1}{\cos\theta_j \Delta\lambda}\left(\psi_{i+1,j} - \psi_{i,j}\right). \tag{3.34b}$$

For a finite volume flux-form advection scheme on a C-grid like Lin-Rood, zonal velocity values are needed along meridional cell edges, and meridional velocity values are needed along zonal cell edges. These velocity values are computed by averaging the nearest four data values. For example, in figure 3.1 (b), the zonal flux velocity $u_{i+1/2,j}$ will be used for flux calculations; it is defined as

$$u_{i+1/2,j} = \frac{1}{4}\left(u_{i,j+1/2} + u_{i,j-1/2} + u_{i+1,j-1/2} + u_{i+1,j+1/2}\right). \tag{3.35}$$

Meridional flux velocities averaged to zonal cell edges are computed similarly.

We use the Lin-Rood advection scheme [116] with these flux velocities to advect vorticity to the next time step [117, equations 13 and 15]. This scheme uses a dimension splitting technique to solve the advection equation as a combination of two 1D advection problems, one for zonal flux and one for meridional flux. Numerical fluxes from [116, equation 2.24] are evaluated using the piece-wise parabolic method of Colella & Woodward [35].

The Lin-Rood scheme applies a semi-Lagrangian time step in the zonal direction to address the pole problem of the latitude-longitude grid [116, section 3]. Time stepping for zonal advection is divided into an integer part and a fractional part, where integer refers to the integer part of the CFL number. Integer CFL numbers imply exact advection, as tracers or vorticity are advected over an integer number of grid cells. This allows a longer time step to be used, with a correction applied later for the fractional part of the CFL number. Applying this semi-Lagrangian technique in the zonal direction prevents the polar grid cells from restricting the global time step. Semi-implicit time stepping for the advection scheme uses an average of two intermediate time steps so that error terms due to the long semi-Lagrangian time step cancel [117, equation 20]. This method is slightly different than the implementation of Lin-Rood advection used by CAM-FV, since here we advect vorticity directly, rather than computing it from velocity.

To summarize the Eulerian model, the Poisson solver (3.33) gives velocity derivatives (3.34) which are averaged to cell interfaces by (3.35). The averaged velocities become the input to the advection scheme, which uses a dimension-split flux form solver and a semi-Lagrangian time step. Further details can be found in [116, 117].

We also note that this advection scheme was used in a shallow water equation solver [117], which was then applied to a vertically Lagrangian discretization of the 3D primitive equations of a hydrostatic atmosphere in [115]. This became the CAM-FV dynamical core, which was the default NCAR atmospheric model through November, 2012. CAM-FV was then replaced by a spectral element method, CAM-SE, which uses a cubed sphere grid and has much better parallel computing scalability [43].

CHAPTER IV

Barotropic Vorticity Equation: Test Cases and Results

In this chapter we solve several test cases and model problems with the Lagrangian Particle-Panel Method (LPPM) that we developed in the previous chapters. The test cases begin with problems that have analytic solutions for either particle positions or vorticity as a way to measure error and verify our numerical methods. Moving on, we find solutions to problems that may not have analytic solutions but are applicable to the geophysical fluid dynamics modeling community. Finally, we apply LPPM to a model of breaking Rossby waves in the stratosphere in a computation designed to simulate the vorticity dynamics associated with sudden stratospheric warming events. For brevity, we assign each test case an abbreviation; these are presented in table 4.1.

In all of the cases except for the first (solid body rotation) the sphere has a background rotation and the equations are solved in the rotating frame. The dimensionless Rossby number *Ro* is a ratio that expresses the significance of relative accelerations to accelera-

Abbreviation	Test problem
SB	Solid body rotation
RH4	Rossby-Haurwitz wave
GV	Single Gaussian vortex
SJ	Steady jet
PJ	Perturbed jet
StratModel	Stratosphere model

Table 4.1: List of the test cases and models whose solutions are presented in this chapter.

tions due to the background rotation of the sphere, or Coriolis accelerations [183, section 2.8.1]. The Rossby number can be derived from scale analysis of the horizontal momentum equation and is defined as

$$R_o = \frac{U}{fL}, \tag{4.1}$$

where U is a typical velocity magnitude, L is a typical length scale, and f is the magnitude of the Coriolis parameter. Typical values for the planetary scale motions modeled well by the barotropic vorticity equation are $U \sim 10$ m/s, $L \sim 10^6$ km, and $f \sim 10^{-4}$ s^{-1} give a Rossby number of $R_o = 0.1$ [80]. It is more convenient, for this work, to express the Rossby number in terms of vorticity; noting that relative vorticity has units of U/L, we express the Rossby number as

$$Ro = \frac{\zeta}{f} = \frac{\max_S |\zeta|}{2\Omega}. \tag{4.2}$$

Problems with larger values of Ro will be less affected by the background rotation of the sphere given by the vector $\mathbf{\Omega}$. In this chapter we solve problems with $0.1 \leq Ro \leq 1$, to stay close to values near that of large scale fluid motion on the Earth.

4.1 Solid body rotation

Rigid rotation is a standard test case for developing global circulation models, and is frequently combined with the addition of a passive tracer to examine a method's ability to conserve mass as well as its ability to resolve a simple fluid flow [187]. Solid body rotation tests are usually conducted in the context of solving the advection equation, but Lagrangian methods satisfy the advection equation exactly for such a simple flow, so we adapt the test case to the BVE instead. Unlike most test cases, this problem has exact solutions for particle positions which makes it a particularly good test problem for a Lagrangian method.

The test case corresponds to a fluid rotating in unison with the underlying sphere, which is summarized succinctly by stating that there is zero relative vorticity,

$$\omega = f. \tag{4.3}$$

The stream function is $\psi_{\mathrm{I}} = \Omega \sin\theta$ in the inertial frame and $\psi_{\mathrm{R}} = 0$ in the rotating frame. We choose $\Omega = 2\pi$ to normalize the period of rotation.

Setting $\boldsymbol{v}_R = 0$ in (3.2b), and applying the flow map equation $d\boldsymbol{x}/dt = \boldsymbol{v}_I$, we find the exact solution for the particle positions is the solution of the following ODE [137],

$$\frac{d}{dt}\boldsymbol{x}(t) = \boldsymbol{\Omega} \times \boldsymbol{x}. \tag{4.4}$$

In the rotating frame the exact solution is the initial position, and is constant,

$$\boldsymbol{x}_R^E(\boldsymbol{\alpha}, t) = \boldsymbol{\alpha}. \tag{4.5}$$

The fluid simply rotates with angular velocity Ω, and has zero velocity in the rotating frame. This solution is a steady state of the Lagrangian equations of motion and our method maintains it exactly.

In order to test our software and numerical method then, for this test case only, we use the inertial frame based on (3.23). In the inertial frame with $\boldsymbol{\Omega} = [0, 0, \Omega]$, the exact solution in spherical coordinates only involves longitude: $\lambda(t) = \lambda_0 + \Omega t$. Latitude remains constant. If $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \alpha_3]^T$, the Cartesian coordinate solution is

$$\boldsymbol{x}_I^E(\boldsymbol{\alpha}, t) = \begin{bmatrix} \alpha_1 \cos\Omega t - \alpha_2 \sin\Omega t \\ \alpha_2 \cos\Omega t + \alpha_1 \sin\Omega t \\ \alpha_3 \end{bmatrix}. \tag{4.6}$$

The superscript E denotes that these are the exact solutions. Error measures (1.23) for this test case are calculated with respect to particle position. For example, the l_2 error is

$$E_{\boldsymbol{x}}(t) = \frac{\left(\sum_{j=1}^{N} \left|\boldsymbol{x}_j(t) - \boldsymbol{x}^E(\boldsymbol{x}_{0j}, t)\right|^2 A_j\right)^{1/2}}{\left(\sum_{j=1}^{N} \left|\boldsymbol{x}^E(\boldsymbol{x}_{0j}, t)\right|^2 A_j\right)^{1/2}}. \tag{4.7}$$

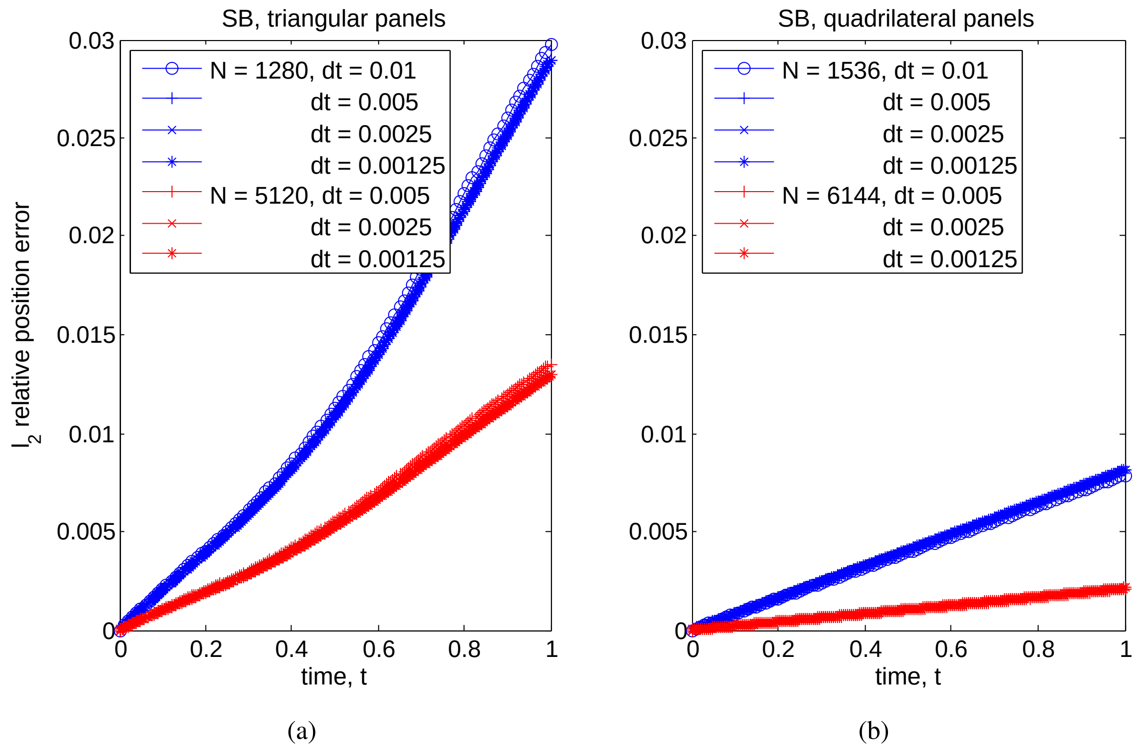


Figure 4.1: SB, no remeshing: Position error vs. time using (a) triangular panels and (b) quadrilateral panels for various values of Δt. From $t = 0$ to $t = 1$ the sphere completes one full revolution.

We examine particle position errors for different values of N and several values of Δt in figure 4.1. We do not use remeshing in these calculations in order to verify the validity of the Biot-Savart law for velocity and that the time integration scheme is behaving as we expect from theoretical predictions. Remeshing will be investigated with Rossby-Haurwitz wave test problems in section 4.2. The plots in figure 4.1 verify that higher resolution panel sets have both lower errors and lower rates of error growth. Improvement in error from decreasing Δt is not evident on this plot; it will be addressed next.

Figure 4.1 suggests that the dominant sources of error are related to the spatial discretization, rather than the time step. When Δt is small, it is difficult to identify time discretization error when comparing computed solutions to exact solutions. Instead, we compute a reference solution for a given N using a very small time step Δt^*, and measure

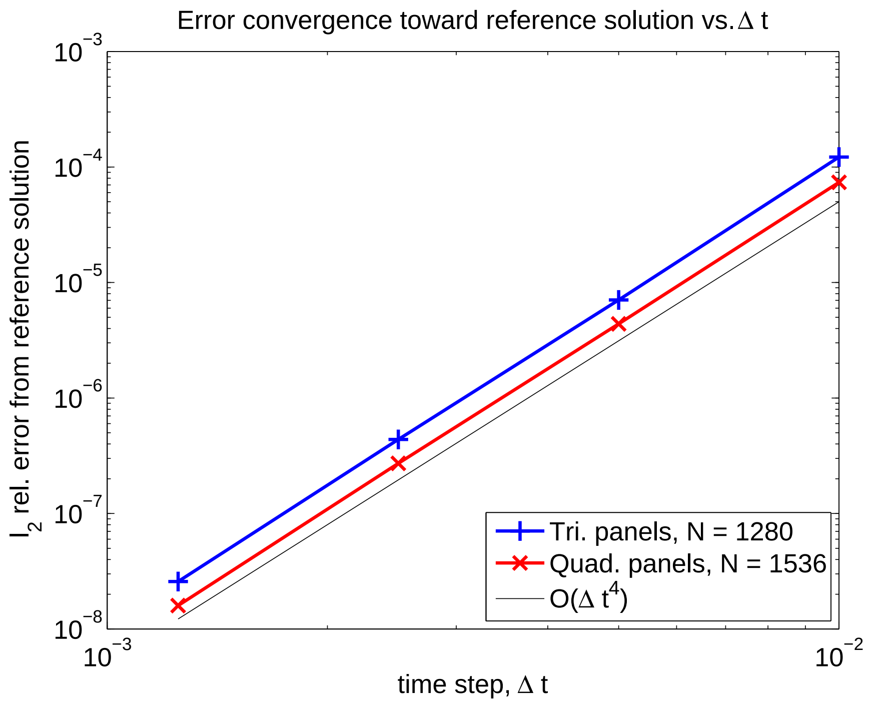


Figure 4.2: Solid body rotation, one revolution: Convergence to reference solutions as $\Delta t \to 0$ verifies fourth order Runge-Kutta accuracy.

convergence to that reference solution as Δt decreases to Δt^*. Particle position differences from the computed reference solutions are plotted against Δt in figure 4.2 for two panel sets. The blue curve shows the relative l_2 position error as $\Delta t \to 0$ for a set of $N = 1280$ triangular panels, and the red curve shows the same results for a set of $N = 1536$ quadrilateral panels. Both curves verify fourth order convergence as $\Delta t \to 0$, which we expect from the Runge-Kutta time integration scheme.

We also use this test case to investigate whether or not particles depart from the sphere, as they do with lower order time stepping schemes, or when the time step is too large. Figure 4.3 shows the maximum particle departure from the sphere after one full revolution for one set of $N = 1280$ triangular panels and one set of $N = 1536$ quadrilateral panels. Both plots show that particles stay within $O\left(\Delta t^4\right)$ of the sphere which allows us to avoid re-projecting particles back to $|\boldsymbol{x}| = 1$.

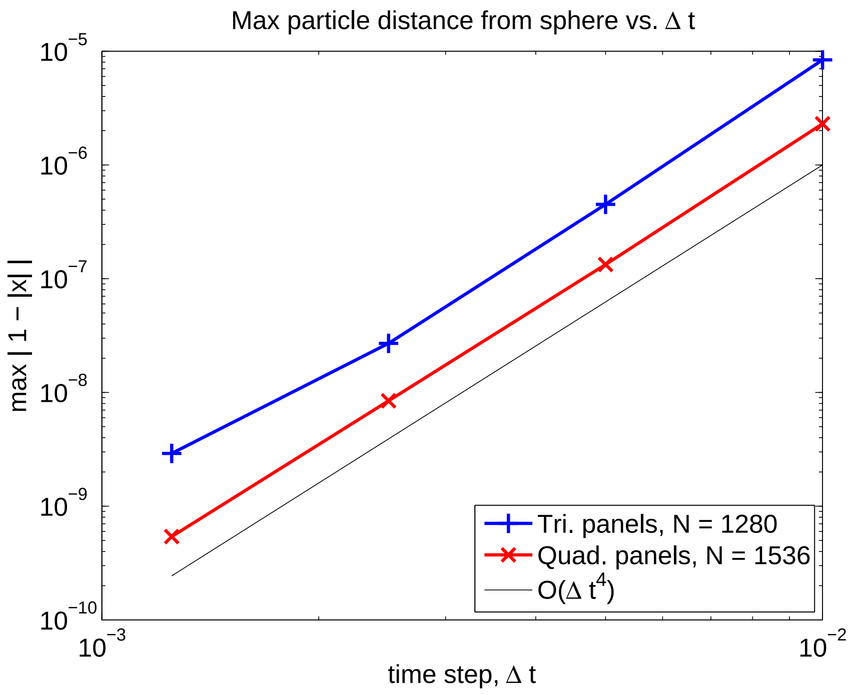


Figure 4.3: Solidy body rotation: Max distance from sphere after one revolution vs. time step for two panel sets. Triangular panels, $N = 1280$ (blue); quadrilateral panels, $N = 1536$ (red).

Finally, we use the solid body rotation test case to time the parallel implementation discussed in section 2.5. Noting that our current method (3.26) for solving the BVE requires $O\left(N^2\right)$ operations per time step, we expect that communication time will be asymptotically smaller than computation time as $N \to \infty$, and that parallel speedup and efficiency should scale well, with improving results for larger N. These expectations are verified in table 4.2, which presents timing data from computations on the University of Michigan's Linux-based parallel computing cluster, *Flux*. Flux uses Intel processors and our code was compiled with the Intel compiler suite using the OpenMPI distributed memory library with `-O2` optimization. Without parallelizing anything except the velocity computation we find nearly linear speedup up to 48 processors, and parallel efficiency greater than 70% for even a coarse $N = 24576$ panel set.

Parallel Speedup						
processors	2	4	8	16	24	48
$N = 24576$	2.00	3.97	7.70	13.69	20.27	35.47
$N = 98304$	1.98	3.98	7.73	14.66	21.84	42.04

Parallel Efficiency						
processors	2	4	8	16	24	48
$N = 24576$	1.00	0.99	0.96	0.86	0.84	0.74
$N = 98304$	0.99	0.99	0.97	0.92	0.91	0.88

Table 4.2: Parallel speedup and parallel efficiency of algorithm II.1 for the barotropic vorticity equation. Timing results from code running on UM's Flux cluster, Intel compilers with OpenMPI, `-O2` optimization.

4.2 Rossby-Haurwitz waves

Rossby-Haurwitz waves are an important test case for several reasons. A Rossby-Haurwitz wave is a circulation pattern whose stream function is given by a spherical harmonic [52, section 5.4]. Haurwitz [77] demonstrated the similarity of several spherical harmonic modes to observations of global circulation. They are also analytic solutions of the barotropic vorticity equation, which provide easy measures of error. The stream function of a generic Rossby-Haurwitz wave has the form

$$\psi(\lambda, \theta) = \cos(m\lambda) P_n^m(\sin\theta), \tag{4.8}$$

where P_n^m is the associated Legendre polynomial from spherical harmonic $Y_n^m(\lambda, \theta) = e^{im\lambda} P_n^m(\sin\theta)$. We choose the real part of Y_n^m for the stream function which results in the cosine term in (4.8). Many texts provide introductions to spherical harmonic functions; we have used [185, section 43] and [22, section 18.11]. Spherical harmonics are eigenfunctions of the Laplace-Beltrami operator (1.14) on $\mathcal{S}$, with eigenvalues $n(n+1)$,

$$\nabla^2 Y_n^m(\lambda, \theta) = n(n+1) Y_n^m(\lambda, \theta) \tag{4.9}$$

Thus, the vorticity associated with a Rossby-Haurwitz wave is simply a multiple of the stream function ψ,

$$\zeta(\lambda, \theta) = -\nabla^2 Y_n^m(\lambda, \theta) = -n(n+1) Y_n^m(\lambda, \theta). \tag{4.10}$$

We may also derive the phase speed of a Rossby-Haurwitz wave that follows from (4.9). To begin, we expand the material derivative in the BVE into its components

$$\frac{d\omega}{dt} = \frac{\partial \zeta}{\partial t} + \boldsymbol{u} \cdot \nabla(\zeta + f) = 0, \tag{4.11}$$

where we have used the fact that $\partial f/\partial t = 0$. Next we write velocity and vorticity in terms of the stream function, using (3.11b) and (3.14b), and replace the gradient with (1.14),

$$-\frac{\partial \nabla^2 \psi}{\partial t} - \frac{\partial \psi}{\partial \theta}\frac{1}{\cos\theta}\frac{\partial}{\partial \lambda}(-\nabla^2\psi + f) + \frac{1}{\cos\theta}\frac{\partial \psi}{\partial \lambda}\frac{\partial}{\partial \theta}(-\nabla^2\psi + f) = 0. \tag{4.12}$$

Using the eigenfunction relation specific to this test case, (4.9), we find that

$$-n(n+1)\frac{\partial \psi}{\partial t} + 2\Omega\frac{\partial \psi}{\partial \lambda} = 0 \tag{4.13a}$$

$$\implies \frac{\partial \psi}{\partial t} - \frac{2\Omega}{n(n+1)}\frac{\partial \psi}{\partial \lambda} = 0 \tag{4.13b}$$

since $\partial f/\partial \lambda = 0$. Note that (4.13b) is the linear advection equation in the longitudinal direction [52, chapter 5]. Hence, the stream function and vorticity pattern of a Rossby-Haurwitz wave propagate toward the west without changing shape with angular phase speed $c = 2\Omega/(n(n+1))$.

In the tests that follow, we add a zonal wind to the vorticity profile, so that the stream functions and relative vorticity for this test case have the form

$$\psi(\lambda, \theta) = \alpha \sin\theta + C\cos(m\lambda)P_n^m(\sin\theta) \tag{4.14a}$$

$$\zeta(\lambda, \theta) = 2\alpha \sin\theta + n(n+1)C\cos(m\lambda)P_n^m(\sin\theta) \tag{4.14b}$$

The parameter α defines the strength of the background wind, and C is a parameter that defines the maximum amplitude of the Rossby-Haurwitz wave [77].

The waves described by stream function (4.14) have phase speeds with angular velocity

$$c_{RH} = \frac{2\Omega}{n(n+1)} - \frac{\alpha}{\Omega} + \frac{2\alpha}{\Omega n(n+1)} \tag{4.15}$$

in the rotating frame [77, equation 18].

Using (4.15), we may choose α so that the phase speed of the Rossby-Haurwitz wave is zero. We denote this choice of α as α_S to indicate its association with a stationary wave,

$$\alpha_S = \frac{2\Omega}{n(n+1)-2}. \tag{4.16}$$

A stationary Rossby-Haurwitz wave is a steady solution of the barotropic vorticity equation's Eulerian form; the vorticity pattern and the stream function do not move. However, in the Lagrangian formulation, particle velocities are nonzero and this test case is not a steady state [184]. The static wave pattern does provide an additional means for measuring error, even in the Lagrangian frame. With a stationary phase speed $\alpha = \alpha_S$ in (4.14), the exact value of relative vorticity $\zeta_E(\boldsymbol{x})$ at any point in space is given by equation (4.14b) for all time. Using equation (1.23b) the l_2 relative vorticity error is calculated by

$$E_\zeta(t) = \frac{\left(\sum_{j=1}^{N}\left(\zeta_j - \zeta_E(\boldsymbol{x}_j(t))\right)^2 A_j\right)^{1/2}}{\left(\sum_{j=1}^{N}\zeta_E(\boldsymbol{x}_j(t))^2 A_j\right)^{1/2}}, \tag{4.17}$$

where ζ_j is the calculated value of relative vorticity at the jth active particle given by equation (3.26).

Due to its inclusion in the dynamical core test cases for the shallow water equations from Williamson *et al.* [187], the $m = 4$, $n = 5$ Rossby-Haurwitz wave is now a standard problem for atmospheric models. We change the phase speed of the wave from [187] to the stationary speed given by Haurwitz [77]. This allows for a simple way to measure error in the computed vorticity. Our test case is defined by the following functions.

$$\psi(\lambda,\theta) = \alpha\sin\theta + C\cos^4\theta\sin\theta\cos(4\lambda) \tag{4.18a}$$

$$\zeta(\lambda,\theta) = 2\alpha\sin\theta + 30C\cos^4\theta\sin\theta\cos(4\lambda) \tag{4.18b}$$

$$\omega(\lambda,\theta) = 2(\alpha+\Omega)\sin\theta + 30C\cos^4\theta\sin\theta\cos(4\lambda) \tag{4.18c}$$

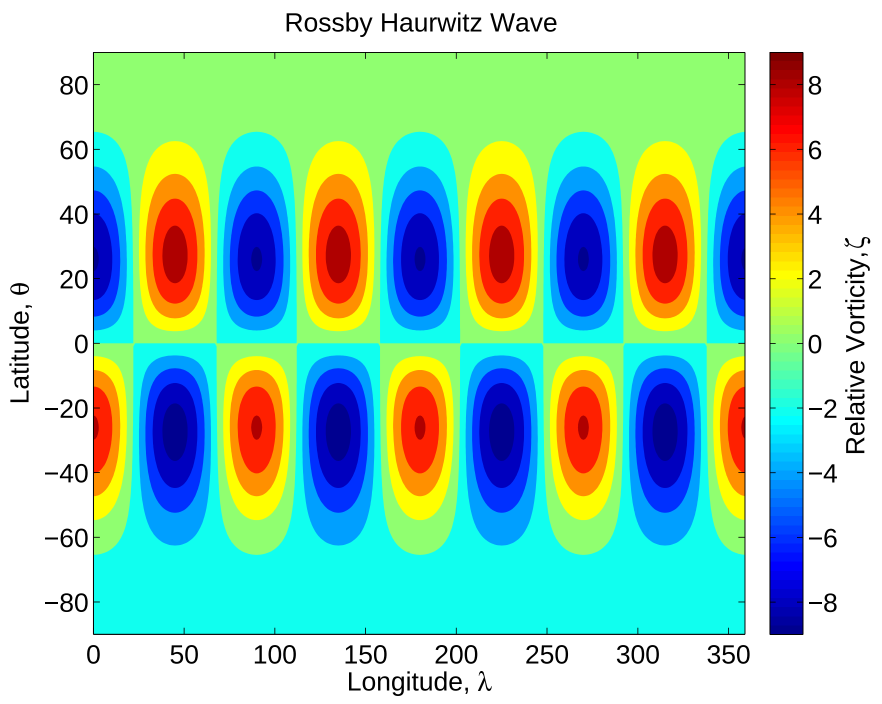


Figure 4.4: Rossby-Haurwitz wave (RH4 test case) relative vorticity from equation (4.18b).

with $\Omega = 2\pi$, $\alpha = \pi/7$ and $C = 1$. $\Omega = 2\pi$ normalizes the rotational period of the sphere. Setting $C = 1$ ensures that the Rossby number for this problem is similar to observed values at $Ro = 0.71$. We refer to this test case as RH4 for brevity; its relative vorticity pattern is shown in figure 4.4.

Similar to how we began the solid body rotation tests, we plot error $E_\zeta(t)$ as a function of time in figure 4.5 for calculations without remeshing to gain an understanding of how the method defined by equations (3.26) behaves on its own. Results for the three different varieties of panel sets introduced in section 2.2 are shown; cubed sphere quadrilaterals are plotted in red, icosahedral triangles in blue, and Voronoi panels in black. Looking at each color individually, we see error decreasing as the panel sets are refined. Error appears to grow at a slow linear rate at early times, then jumps in a manner that suggests instability. In fact the jump is caused by mesh distortion, which we can see by looking at plots of particles, panels, and edges at the corresponding times (figures 2.14 and 2.15).

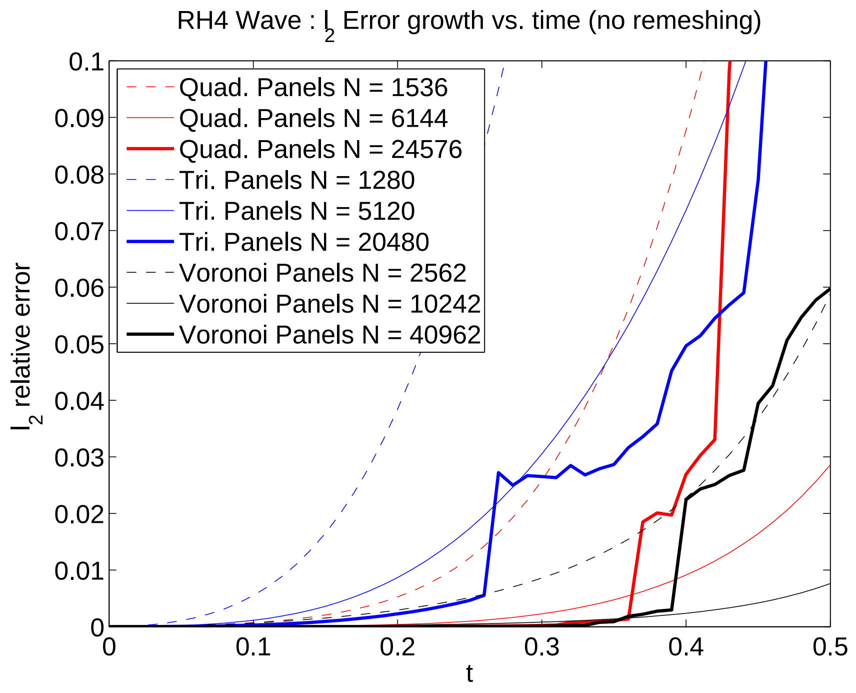


Figure 4.5: RH4: l_2 relative vorticity error percentages vs. time for Rossby-Haurwitz wave Y_5^4 without remeshing. Number of panels are indicated by N in the legend. Quadrilateral panel sets are shown in red; triangular panel sets, blue; Voronoi panel sets, black. All calculations used $\Delta t = 0.01$.

In figure 4.6 we plot total kinetic energy and total enstrophy data from calculations without remeshing. Total kinetic energy and total enstrophy are calculated by applying the particle-panel discretization to (3.28),

(4.19a) $$KE(t) = \frac{1}{2}\sum_{j=1}^{N} \left|\boldsymbol{u}(\boldsymbol{x}_j(t))\right|^2 A_j$$

(4.19b) $$Z(t) = \frac{1}{2}\sum_{j=1}^{N} \zeta_j(t)^2 A_j$$

where $\boldsymbol{u}(\boldsymbol{x}_j(t))$ is given by (3.26a) and $\zeta_j(t)$ is the relative vorticity carried by active particle $\boldsymbol{x}_j(t)$ computed with (3.26c). Figure 4.6 shows that our method works very well prior to the point where distortion becomes too great. Panel sets with higher resolution (greater N) delay the onset of distortion error, as shown in the top plot. At early times, the method performs very well; vorticity errors are less than 0.1%. From the middle plot we see that total kinetic energy depends on N, and for each calculation individually, is conserved

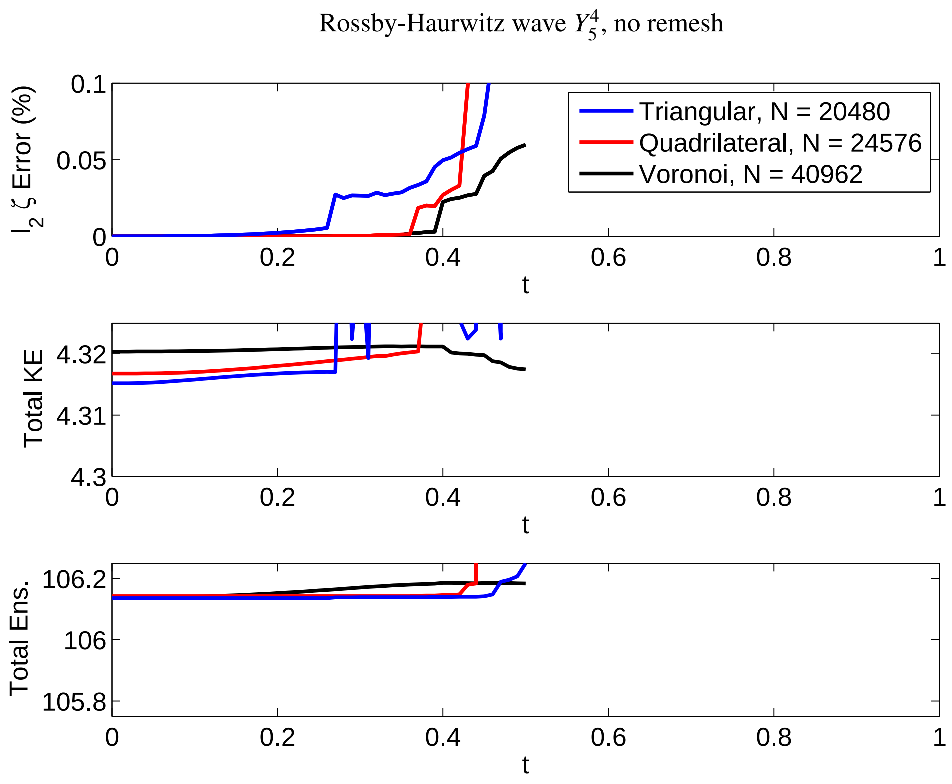


Figure 4.6: RH4 wave, no remesh: vorticity error (top), kinetic energy (middle), and enstrophy (bottom). Triangular panels, $N = 20480$ (blue). Quadrilateral panels, $N = 24576$ (red). Voronoi panels, $N = 40962$ (black).

well while the calculation is stable. Voronoi panel sets do not exactly preserve enstrophy since their passive particles (and hence their panel areas) are determined at each time step. We attribute the differences in vorticity error between each calculation in figure 4.6 as being primarily due to the different values of N, rather than the different polygonal types represented by each color.

To allow our calculations to proceed to longer times, beyond the point where the non-remeshed calculations have failed due to distortion, we apply remeshing. For illustrative purposes, we take the example of a triangular panel set with $N = 20480$ panels and $\Delta t = 0.01$. Figure 4.7 shows the test case results for a panel set without remeshing (black), with direct remeshing (red), and Lagrangian remeshing (blue). Remeshing time steps are $t_{rm} = \{0.1, 0.2, 0.3, \dots\}$. Both methods use the cubic interpolation scheme provided by

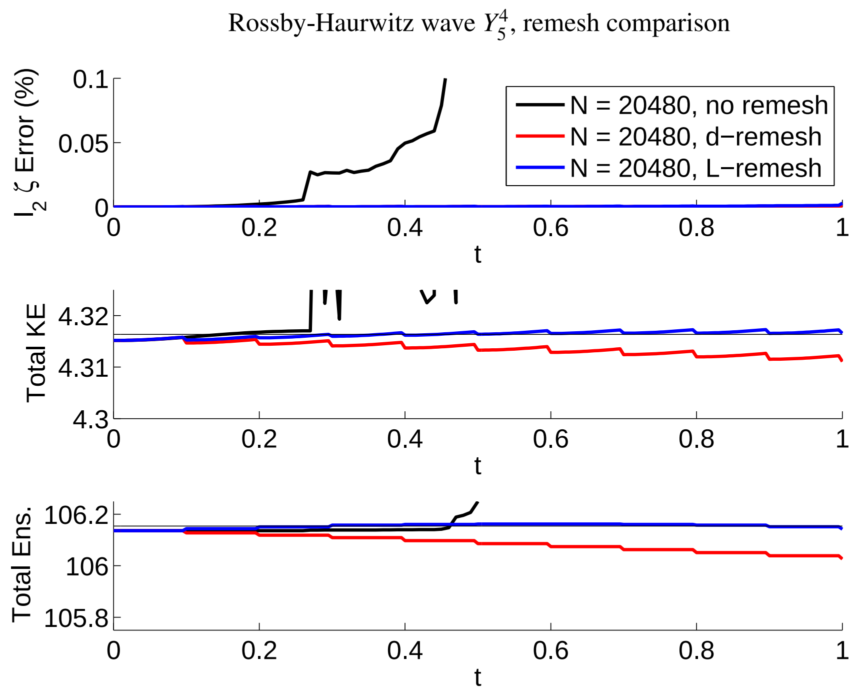


Figure 4.7: RH4 wave, direct vs. Lagrangian remeshing: Vorticity error (top), kinetic energy (middle), and enstrophy (bottom). All calculations used triangular panels with $N = 20480$. Calculation without remeshing shown in black, direct remeshing, red;, Lagrangian remeshing, blue.

SSRFPACK [147]. In the top plot, we see no visually distinguishable difference between direct remeshing and Lagrangian remeshing; both succeed at continuing the calculation to the final time $T = 1$. Differences between the two methods are visible in the plots of kinetic energy (middle) and enstrophy (bottom). The direct remeshing procedure introduces dissipation into the computation and neither kinetic energy nor enstrophy are consserved as well as they are by the Lagrangian remeshing procedure.

Figure 4.8 plots vorticity error at $T = 1$ for the Rossby-Haurwitz wave number 4 test case using different values of N. The convergence rates for both methods are higher than we would expect of the midpoint rule, which indicates that these calculations are dominated by remeshing error. Differences in vorticity error between the two remeshing procedures are visually indistinguishable in figure 4.7, but on the logarithmic scale of figure 4.8 we see that although the Lagrangian remeshing procedure has a faster convergence rate,

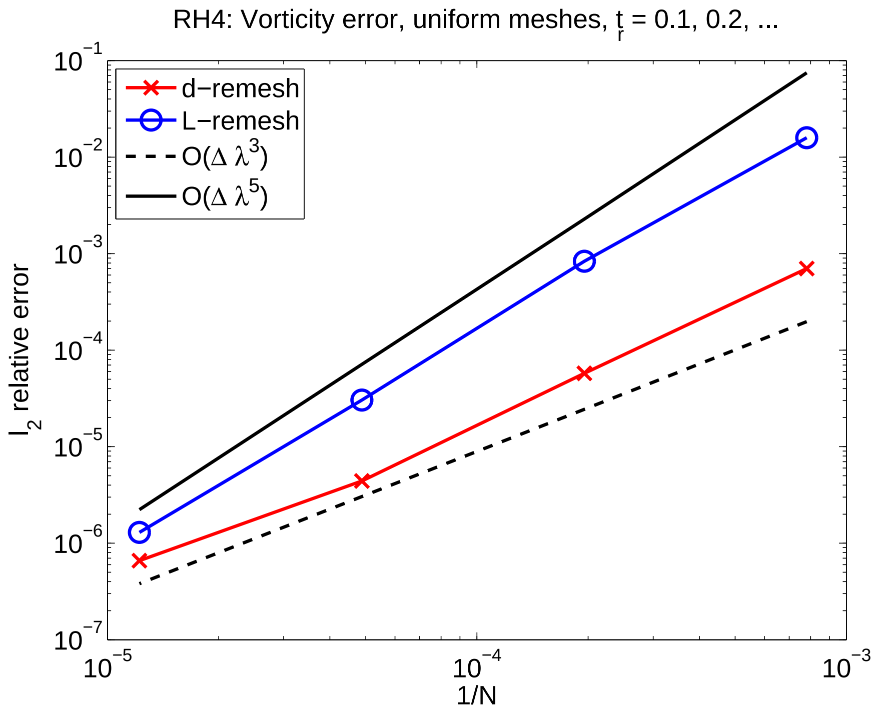


Figure 4.8: RH4: Vorticity error at $t = 1$ is plotted for calculations that used direct remeshing (red) and Lagrangian remeshing (blue) vs. N.

it also has higher values of l_2 vorticity error on coarse panel sets (small N). The difference in vorticity error between remeshing procedures is reduced as panel sets are refined (as $N \to \infty$). The slopes in figure 4.8 suggest that for sets with $N > 100000$, the Lagrangian remeshing procedure may overtake the direct remeshing procedure and become more accurate.

We conclude from figures 4.7 and 4.8 that Lagrangian remeshing achieves better conservation properties than a direct remeshing procedure, which introduces smoothing error into the interpolated quantities. This smoothing is likely responsible for the better accuracy of the direct remeshing procedure for coarse meshes. As computations proceed, a fluid flow may develop features that are too small for a coarse panel to resolve, causing discretization error that a direct remeshing procedure may mitigate by smoothing.

To illustrate this idea, suppose a panel straddles a steep gradient. The combination of the Lagrangian remeshing procedure and midpoint rule quadrature assigns one value of vorticity to the entire panel, regardless of the gradient whose spatial scale is the same or smaller than the panel size. Direct remeshing smooths the gradient somewhat, possibly increasing its spatial scale to a point where the panel set is able to resolve it. We will confront this situation in other test cases, and examine the effectiveness of adaptive refinement for handling small scale features. Although the vorticity pattern is fixed, steep gradients develop in the Lagrangian parameter α as particles are transported by the Rossby-Haurwitz wave. Eventually, these gradients will develop scales smaller than a fixed panel size. Our adaptive refinement strategies address this problem; we also suggest other strategies for future study in chapter VII.

As a final computation for this test case, table 4.3 compares the results from our Lagrangian Particle-Panel Method (LPPM) to the Lin-Rood BVE solver introduced in section 3.6. The LPPM calculations used $\Delta t = 0.01$ for all resolutions and were remeshed with cubic interpolation every 20th time step $t_{rm} = \{0.2, 0.4, \dots\}$. At coarse resolutions, direct remeshing is more accurate than Lagrange remeshing, as explained above. This difference between remeshing procedures disappears as N increases (figure 4.8). Rather than comparing remeshing algorithms, we present table 4.3 to show a side-by-side comparison of LPPM to methods used by an operational dynamical core. The Lin-Rood calculations for 3° and 2° mesh sizes also used $\Delta t = 0.01$, while the 1° computation used $\Delta t = 0.005$. LPPM has less vorticity error by 2-3 orders of magnitude, but Lin-Rood has slightly better l_∞ errors.

Kinetic energy and enstrophy for LPPM using direct remesh (d-Remesh) and Lagrangian remesh (L-remesh) are shown for $N = 20480$ triangular panels (which have panel edges lengths of $\approx 2^\circ$) alongside the Lin-Rood $2^\circ \times 2^\circ$ results in figure 4.9. The Lin-Rood scheme has nearly perfect conservation of both total kinetic energy and total enstrophy,

l_2 vorticity error at t = 2			
	$\Delta\lambda = 3°, \Delta t = 0.01$	**$\Delta\lambda = 2°, \Delta t = 0.01$**	**$\Delta\lambda = 1°, \Delta t = 0.005$**
Lin-Rood	4.53E-02	1.23E-02	5.00E-03
	N = 6144, Δt = 0.01	**N = 20480, Δt = 0.01**	**N = 81920, Δt = 0.01**
LPPM, L-remesh	3.39E-02	3.03E-05	1.29E-06
LPPM, d-remesh	1.17E-03	4.41E-06	6.60E-07

l_∞ vorticity error at t = 2			
	$\Delta\lambda = 3°, \Delta t = 0.01$	**$\Delta\lambda = 2°, \Delta t = 0.01$**	**$\Delta\lambda = 1°, \Delta t = 0.005$**
Lin-Rood	4.85E-02	2.24E-02	5.40E-03
	N = 6144, Δt = 0.01	**N = 20480, Δt = 0.01**	**N = 81920, Δt = 0.01**
LPPM, L-remesh	6.73E-01	5.43E-02	7.59E-03
LPPM, d-remesh	1.59E-02	5.53E-03	2.72E-03

Table 4.3: Comparision of l_2 and l_∞ vorticity error measures for the Lin-Rood BVE solver from section 3.6 and our Lagrangian Particle-Panel Method (LPPM) with Lagrangian remeshing (L-remesh) and direct remeshing (d-remesh). Uniform panel sets (constant N) are chosen to approximately match the resolution of the Lin-Rood lat-long grid spacing.

for this test case, despite the implicit diffusion of its advection scheme [88]. This is due to the fact that this test case is a steady state test problem for an Eulerian method; we would expect the Lin-Rood solver to perform admirably in such a test case. Further comparison of LPPM and Lin-Rood using Rossby-Haurwitz waves with non-stationary phase speeds is an ongoing project.

4.3 Gaussian vortices

Gaussian vortices on the sphere were studied by Wang [184] as an example of vortex dynamics on the sphere. Boyd and Zhou also present analytical solutions for the special case of a Gaussian vortex centered at the north pole of a spherical coordinate system [23]. They note that since the north pole of a spherical coordinate system can be placed at an arbitrary point on $\mathcal{S}$ by applying a sequence of rotational transformations, their results apply to arbitrarily positioned vortices as well.

A general Gaussian distribution on the sphere is given by

$$\zeta^*(\boldsymbol{x}) = Ae^{-2B^2(1-\boldsymbol{x}\cdot\boldsymbol{x}_0)}. \tag{4.20}$$

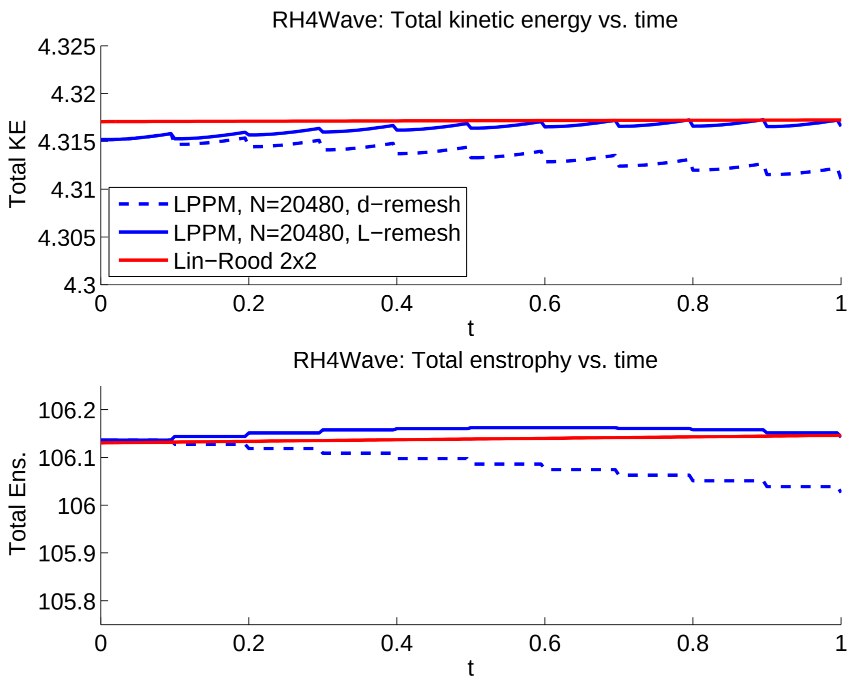


Figure 4.9: RH4 Wave: Kinetic energy and enstrophy comparison of LPPM with each remeshing procedure and the Lin-Rood BVE solver. All calculations used $\Delta t = 0.01$. Since the RH4 test case is a steady state Eulerian solution, Lin-Rood does very well.

where B is the shape parameter of the Gaussian, A is its maximum value, and $\boldsymbol{x}_0 \in \mathcal{S}$ is the location of its center. We recognize the square of the Euclidean distance between an arbitrary point $\boldsymbol{x} \in \mathcal{S}$ and the center of the Gaussian $\boldsymbol{x}_0$ in the exponent as (1.15h). In order to satisfy the necessary condition $\int_{\mathcal{S}} \zeta(\boldsymbol{x})\, dA = 0$, we subtract from ζ^* its average value $\bar{\zeta}$, which is computed as

$$\bar{\zeta} = \frac{1}{4\pi} \sum_{j=1}^{N} \zeta^*(\boldsymbol{x}_j(0)) A_j. \tag{4.21}$$

at the beginning of the computation. $\bar{\zeta}$ is also known as the Gauss constant [184, 23]. This gives initial relative and absolute vorticity distributions of

$$\zeta_0(\boldsymbol{x}) = A e^{-2B^2(1-\boldsymbol{x}\cdot\boldsymbol{x}_0)} - \bar{\zeta} \tag{4.22a}$$

$$\omega_0(\boldsymbol{x}) = 2\Omega z + A e^{-2B^2(1-\boldsymbol{x}\cdot\boldsymbol{x}_0)} - \bar{\zeta}. \tag{4.22b}$$

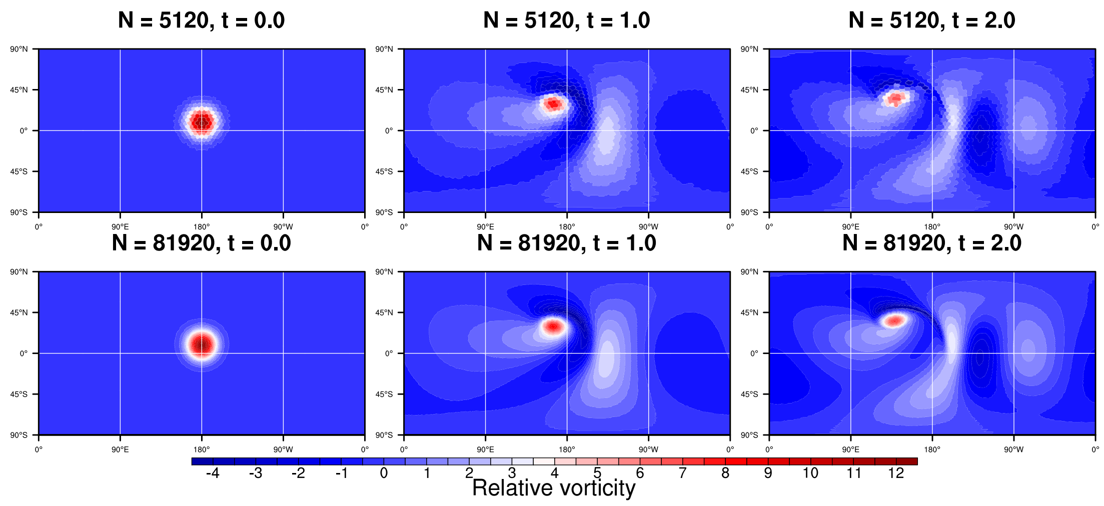


Figure 4.10: Relative vorticity of a Gaussian vortex for two sets of triangular panels. Top row, $N = 5120$. Bottom row, $N = 81920$.

In spherical coordinates, using (1.20), the vorticity distributions are

$$\zeta_0(\lambda,\theta) = Ae^{-2B^2\left(1-\cos\gamma(\lambda,\theta,\lambda_0,\theta_0)\right)} - \bar{\zeta} \tag{4.23a}$$

$$\omega_0(\lambda,\theta) = 2\Omega\sin\theta + Ae^{-2B^2\left(1-\cos\gamma(\lambda,\theta,\lambda_0,\theta_0)\right)} - \bar{\zeta}. \tag{4.23b}$$

To define our specific test case, we choose $B = 4$, $A = 4\pi$, and $\boldsymbol{x}_0 \mapsto (\lambda_0, \theta_0)$ with $\lambda_0 = 0$, $\theta_0 = \pi/20$. The amplitude $A = 4\pi$ of the Gaussian gives a relative vorticity maximum that is equal to the planetary vorticity maximum; the Rossby number for this test case is $Ro = 1$.

The starting center of the Gaussian is 9° north of the equator, and we expect the vortex to travel to the northwest due to β-drift induced by the background rotation of the sphere [80]. Additionally, the vorticity over the majority of the sphere away from the vortex is nearly zero, which means that this problem is ideal for testing the efficiency of our adaptive refinement strategies from section 2.3. We establish a baseline with some uniform mesh computations. Example results are plotted in figure 4.10 and a comparison with results from the Lin-Rood solver (section 3.6) is shown in figure 4.11.

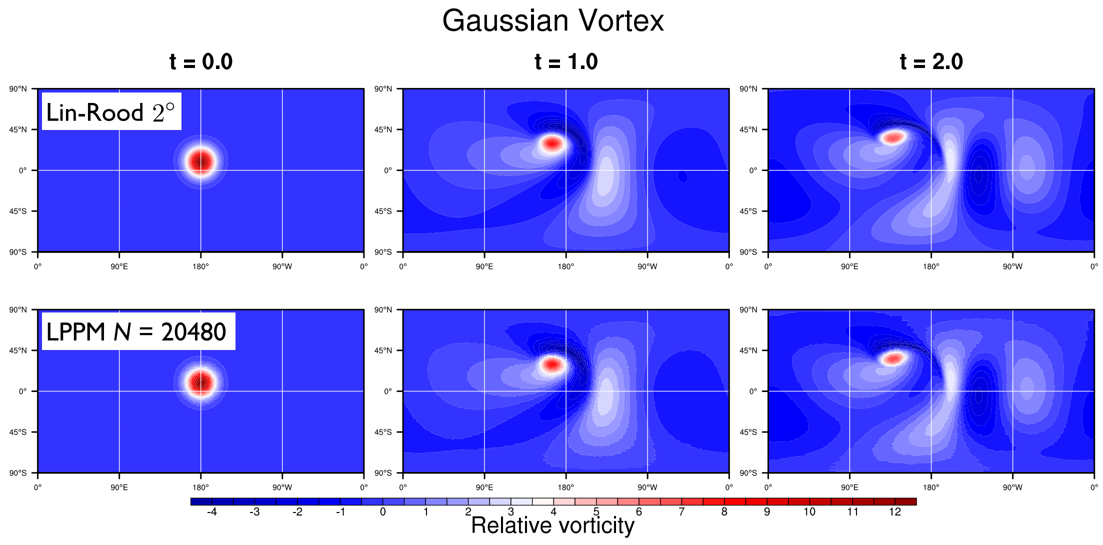


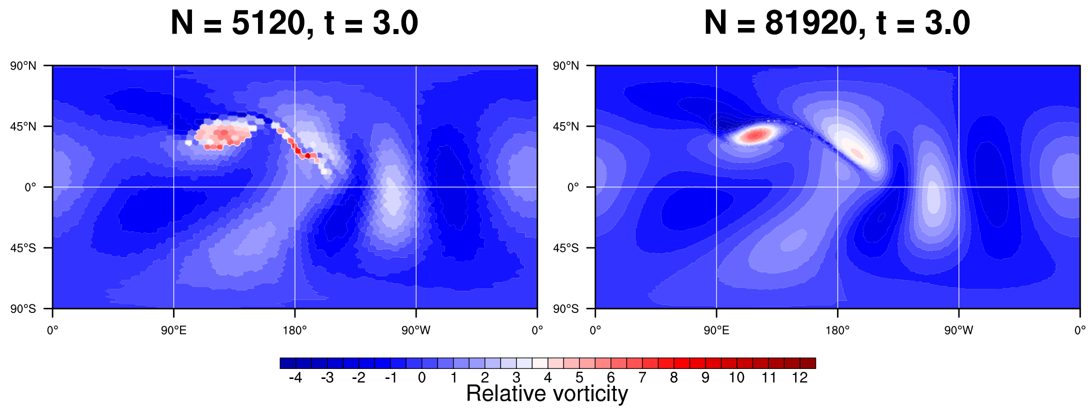
Figure 4.11: GV: Comparison of Lin-Rood BVE solver to LPPM for Gaussian vortex test case. At 2°, there are 16200 grid cells on the Lin-Rood lat-long grid (top row); 20480 triangular panels are a comparable resolution (bottom row).

Figure 4.10 shows the results from two computations using triangular panels. The top row shows a relatively coarse resolution with $N = 5120$, which gives an average initial panel edge length of approximately 4°. The bottom row used $N = 81920$, which corresponds to an approximate resolution of 1°. The time step for the $N = 5120$ computation was $\Delta t = 0.01$ and the time step for $N = 81920$ was $\Delta t = 0.005$; both computations used cubic Lagrangian remeshing at intervals of twenty time steps.

Three times are plotted in figure 4.10; the initial time $t = 0$ is shown in the left column, $t = 1$ in the middle, and $t = 2$ on the right. At $t = 1$, both computations appear well-resolved with similar solutions. β-drift is evident as t increases; the vortex moves northwest and becomes asymmetric due to the meridional planetary vorticity gradient. As the vortex rotates and travels away from its initial position, fluid that began at the equator gets entrained into the vortex and transported poleward. Absolute vorticity is conserved following fluid motion, so as equatorial fluid particles move northward into regions with larger Coriolis force, their relative vorticity decreases. This results in the dark blue patch of negative vorticity along the north and east periphery of the vortex. We also see an eastward propagating equatorial wave in all calculations; as suggested by [21], such waves are unique to spherical geometry and not present in equatorial β-plane models.

Additionally, at $t = 3$ in figure 4.12 we see the development of a secondary vortex along the filament left behind by the original vortex, especially in the coarse resolution computations. The $N = 5120$ computation is clearly unresolved at $t = 3$, and a checker-board effect has appeared in the relative vorticity field. The loss of resolution as t increases is not due to distortion; our remeshing procedure has addressed that issue. Rather, resolution is lost as the developing filament width becomes smaller than the edge lengths of the uniform panels. By $t \approx 1.5$ in the $N = 5120$ calculation, the filament shed by the vortex has decreased in width to the point that it is smaller than the roughly 4° panels used to compute the flow. Some panels straddle the filament and are unable to resolve its associated sub-grid scale vorticity gradient. The checker-boarding is due to the combination of our Lagrangian remeshing scheme, which does not smooth flow features, and midpoint rule quadrature which assigns one constant value of vorticity to the entire panel regardless of the gradient. This error leads to an erroneously strong secondary vortex at $t = 3$ in the $N = 5120$ computation. The higher resolution $N = 81920$ computation is more capable of resolving the filament that develops, and its secondary vortex is much weaker.

The filament represents fluid that began at the equator and has since moved poleward; this transport is the means of generating the vorticity gradient along the north and east periphery of the vortex. The Lagrangian parameter α provides a way of visualizing filament development and fluid transport. Figures 4.13 and 4.14 plot the development of the vortex in two ways: left column plots show relative vorticity, while the plots in the right column show the latitudinal component of the Lagrangian parameter α. Since we have set α equal to particles' initial positions, the latitudinal component of α is equivalent to particles' ini-



(a) (b)

Figure 4.12: GV: A checkerboard effect shows up in relative vorticity plots when resolution is lost (a); comparing with figure 4.15, we see that this corresponds to a loss of conservation of both kinetic energy and total enstrophy. Increasing resolution (b) indicates that the strength of the secondary vortex that develops along the filament is sensitive to this type of error.

tial latitudes. The color scale for the initial latitude plots is chosen so that particles that begin near the equator are colored black, particles that begin in the northern hemisphere are colored red, orange, and yellow, and particles in the southern hemisphere at $t = 0$ are colored shades of blue. The vortex transports equatorial fluid toward the northwest in a filament that gets stretched to finer spatial scales as t increases. This filament is the feature that the $N = 5120$ panel set fails to resolve for $t > 1.5$. The $N = 81920$ panel set is capable of resolving the filament until $t \approx 2.5$, at which point it has stretched to scales finer than $1°$.

Figure 4.15 shows the computed total kinetic energy (top) and total enstrophy (bottom) for four different N. Dashed vertical lines are twenty time steps apart and indicate remeshing times. The green lines in figure 4.15 correspond to the $N = 5120$ computation in the top row of figure 4.10; the red line shows the $N = 20480$ computation from the bottom row of figure 4.11, and the royal blue lines in figure 4.15 correspond to the $N = 81920$ computation shown in the bottom row of figure 4.10 and figures 4.13 and 4.14. It is easy to pick out on energy and enstrophy plots the times at which each N can no longer resolve

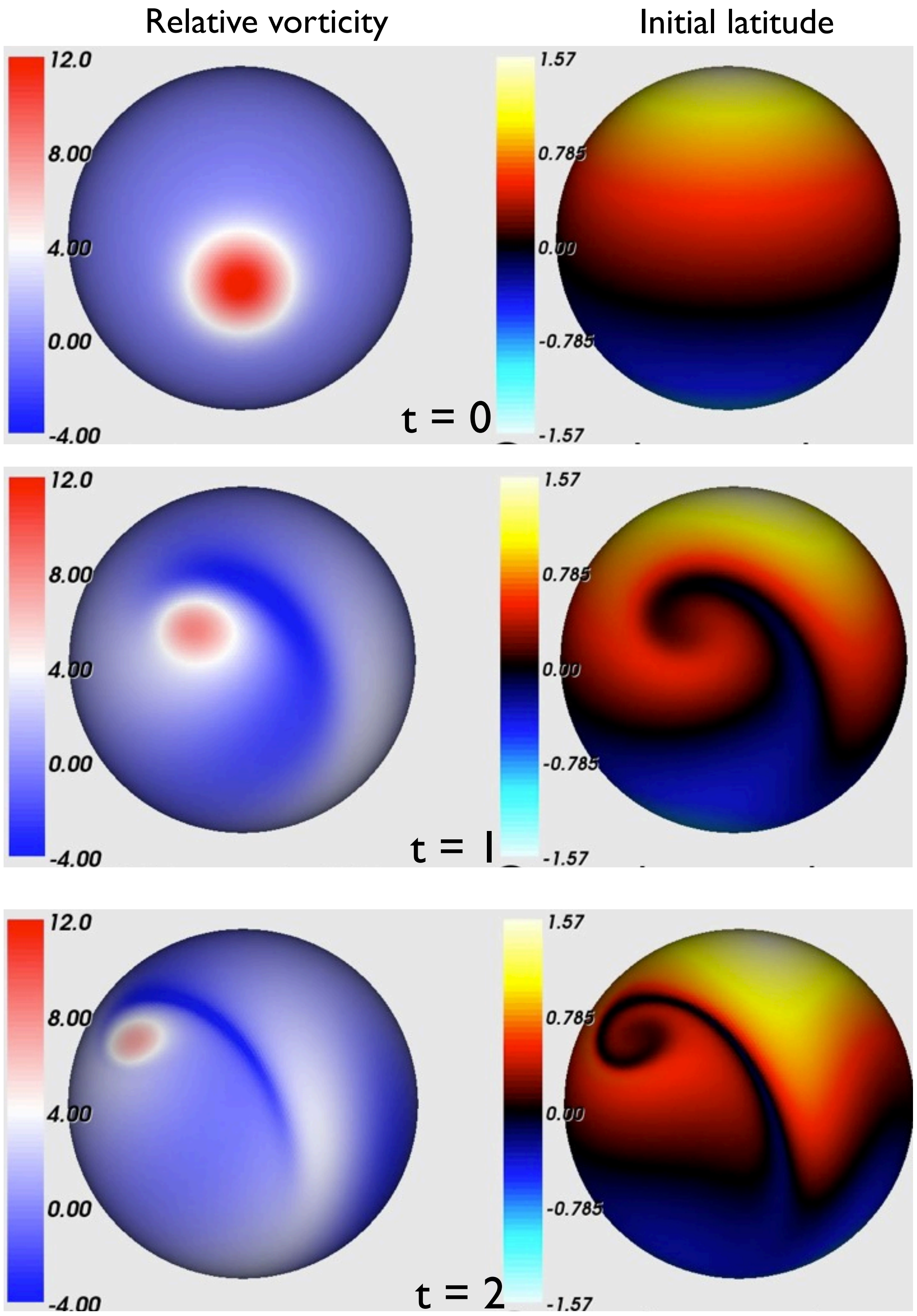


Figure 4.13: Time evolution of relative vorticity and initial latitude due to a Gaussian vortex; $N = 81920$ triangular panels and $\Delta t = 0.005$, Lagrangian remeshing every 20th time step.

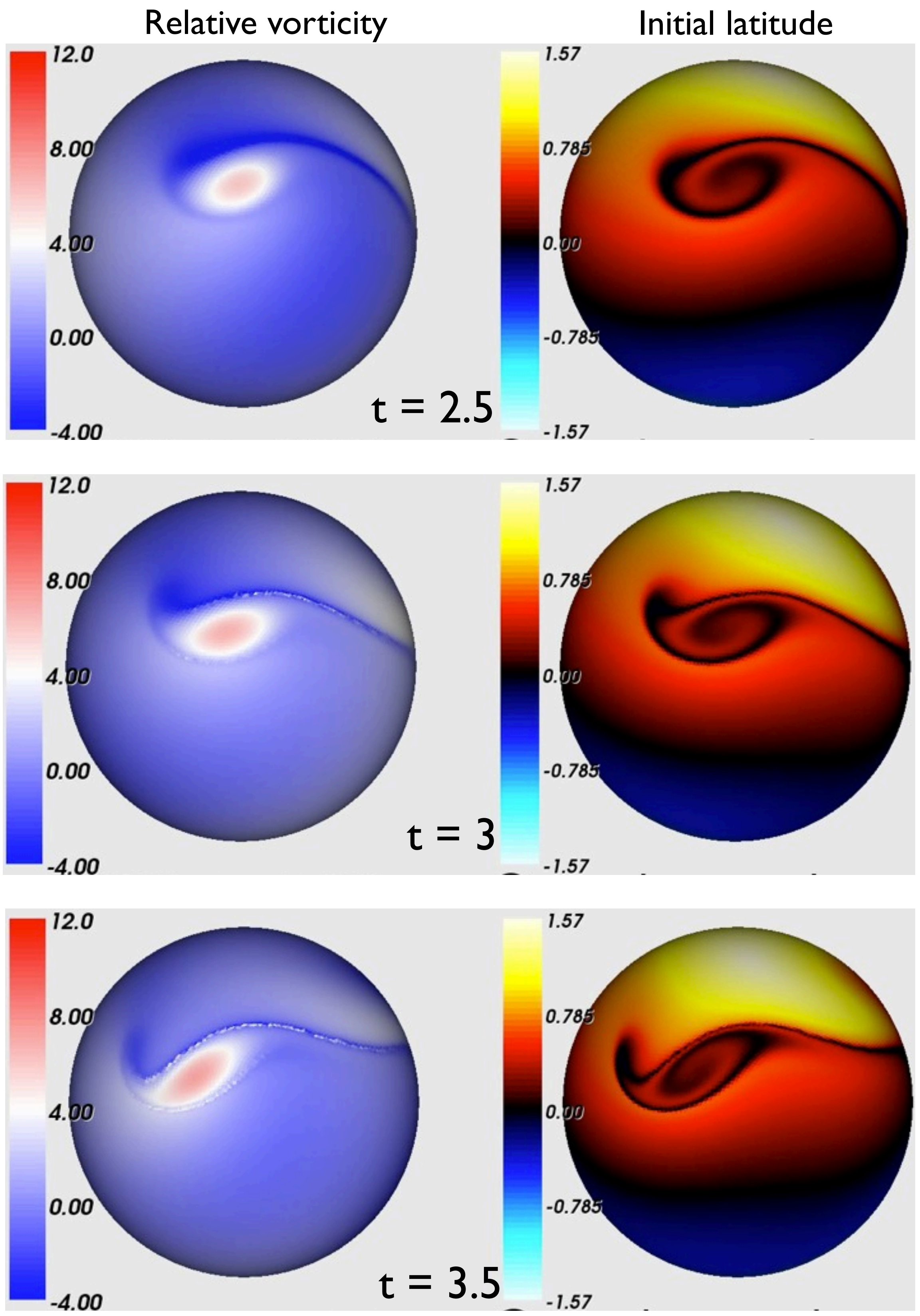


Figure 4.14: Continued from figure 4.13, with the view re-centered on the vortex.

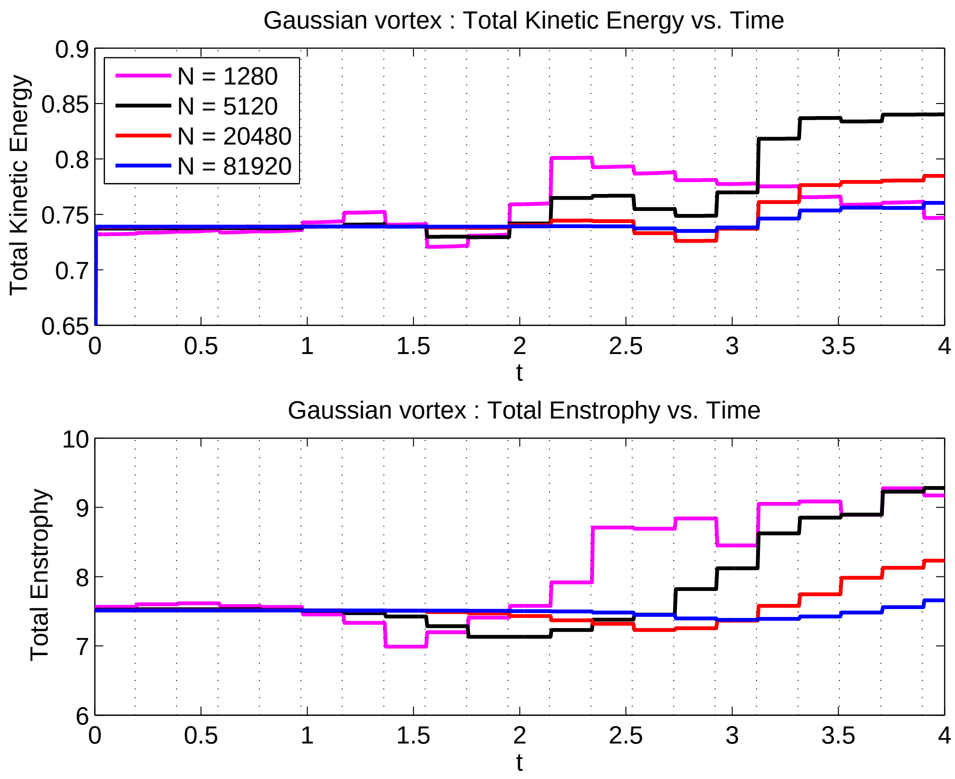


Figure 4.15: GV: Total kinetic energy and enstrophy vs. time. Triangular panels. Vertical dashed lines are 20 time steps apart and indicate remeshing steps.

the filament, as these times coincide with a loss of conservation. Shortly after $t = 2.5$, even the $N = 81920$ computation is no longer able to resolve the filament; after this time the blue lines depart from their initial values of both total kinetic energy and enstrophy. Checker-boarding effect begins to appear in the relative vorticity plots for $t > 2.5$ as well (figure 4.14).

Aside from providing a visualization of fluid transport, the plots of the Lagrangian parameter's latitudinal component provide a means of viewing the flow map. The Lagrangian remeshing procedure interpolates the Lagrangian parameter; the plots of initial latitude provide a glimpse of that interpolation problem. Considering the $N = 81290$ computation of figures 4.13 and 4.14 along with the blue line from figure 4.15, we see that while the flow is resolved, our method conserves both energy and enstrophy very well.

Conservation is lost when the flow becomes unresolved. Comparing figures 4.14 and 4.15 we associate the appearance of checker-board paneling in relative vorticity plots with a loss of conservation of both energy and enstrophy. Indications that the flow has become unresolved are thus provided by the appearance of checker-boarding along filaments in vorticity plots, or equivalently, a loss of conservation of energy or enstrophy.

The onset of error caused by a loss of resolution may be deferred using adaptive refinement. We set up a computation with the adaptive refinement criteria of section 2.3, starting with a background mesh of approximately 8°, and let the adaptive refinement criteria put higher resolution where it is needed.

Figures 4.16 and 4.17 show an adaptive computation that used $\Delta t = 0.005$ and refinement tolerance values of $\epsilon_1 = 0.005$, $\epsilon_2 = 0.001$ applied to an initial background mesh of $N = 1280$ triangular panels. Adaptive Lagrangian remeshing was applied at intervals of 20 time steps. Plot formats are the same as figures 4.13 and 4.14, with relative vorticity plotted in the left column and the initial latitude in the right column. Also plotted in the right column are the panel edges, shown in green, to illustrate the effects of the adaptive refinement criteria. At $t = 0$, we only applied the circulation criterion (2.8) which yielded a mesh with $N = 2966$ panels, with the highest resolutions focused at the vortex. By $t = 1$ the combination of both the circulation criterion and the total variation of the Lagrangian parameter criterion (2.10) generate a mesh with $N = 11012$ panels. The vortex core has remained highly resolved, and the filament is beginning to develop its own regions of higher resolution. Through $t = 2$ and $t = 2.5$ the vortex core and its filament are still well resolved, showing no evidence of the checker boarding that would indicate conservation has been lost.

Side-by-side plots of relative vorticity at $t = 2.5$ for the two computations are shown in figure 4.18. The adaptive criteria resulted in a mesh of $N = 12377$ panels at $t = 2.5$, and

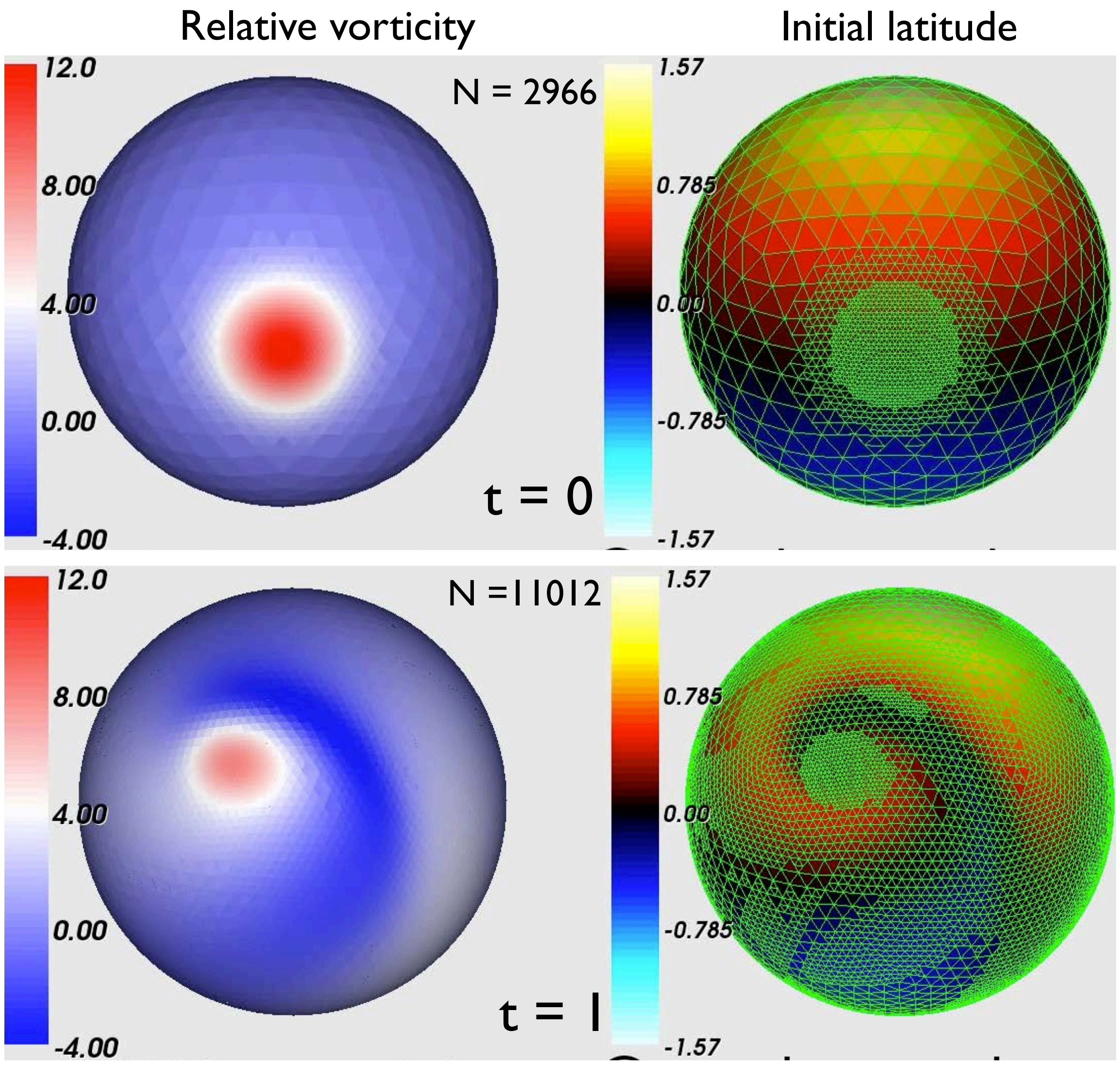


Figure 4.16: Time evolution of a Gaussian vortex with triangular panels and $\Delta t = 0.005$, adaptive Lagrangian remeshing every 20th time step. Coloring of left column shows relative vorticity, right column coloring indicates initial latitude. Panel edges are shown in green.

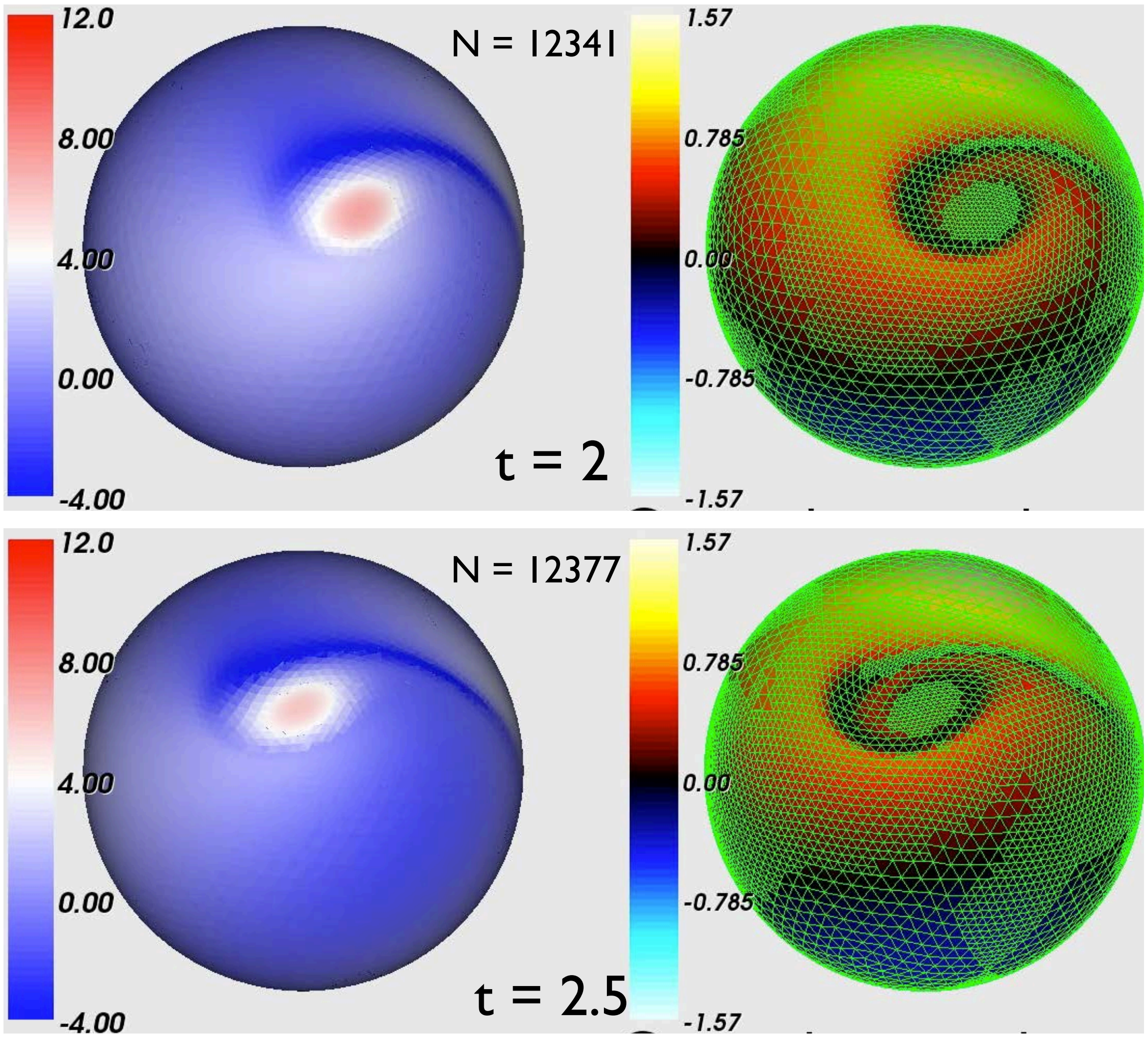


Figure 4.17: Continued from figure 4.16.

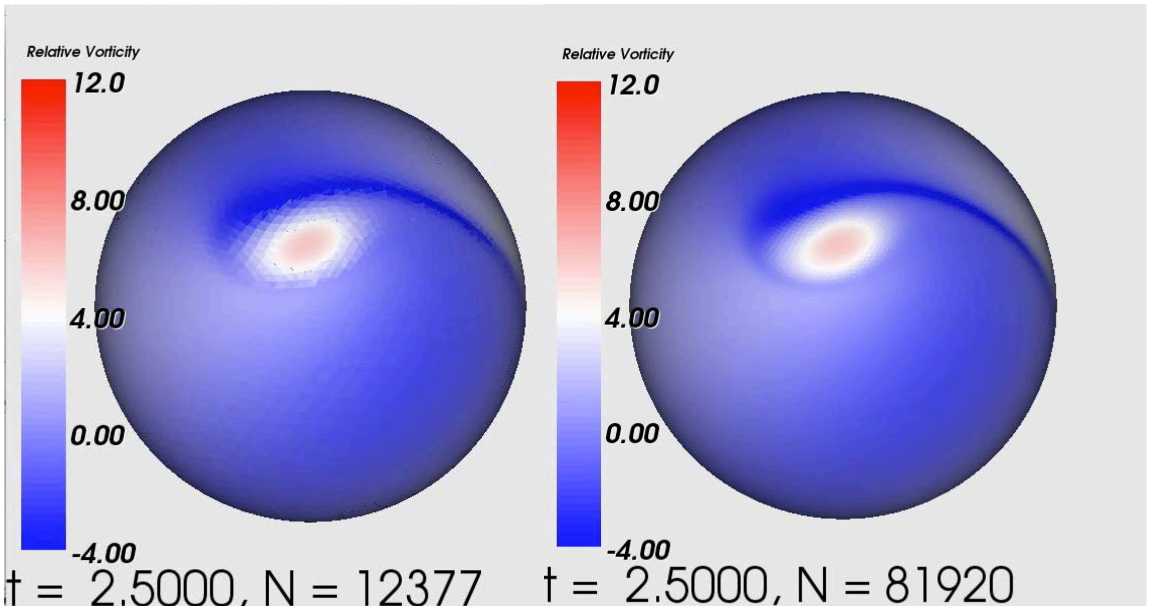


Figure 4.18: Comparison of adaptive (left) vs. uniform (right) mesh for Gaussian vortex test case at $t = 2.5$.

comparable results to the much higher $N = 81920$ uniform mesh computation, with less than 1/6 the number of panels.

4.4 Jet dynamics

Since the barotropic vorticity equation does not include effects of divergence, which are associated with gravity waves, and does not consider thermodynamic effects either, it is not suitable as a model for the atmospheric flow near the planetary boundary layer. It is, however, a fair approximation of higher levels of the atmosphere, where gravity waves are less prevalent and the thermal gradient is low. At pressures of 300 hPa or less, atmospheric flow is dominated by planetary-scale features [52] that can be represented well by the barotropic model. We have already considered the special case of a Rossby-Haurwitz wave whose vorticity distribution is given by a spherical harmonic in section 4.2. Meteorologists use the more general term *Rossby wave* to refer to planetary-scale waves that arise due to the planetary vorticity gradient – the variation of the Coriolis parameter with latitude.

Rossby waves are characterized by large-scale motion that conserves absolute vorticity in a barotropic model or potential vorticity in a baroclinic model [80, section 7.7].

The 300 hPa pressure level is approximately the level of the polar front jet stream, which is a primary forcing mechanism for weather patterns at the surface [15, 52, 80]. In addition to the jet stream's practical importance, it also presents a mathematically interesting problem. A zonal jet along a latitude circle is an unstable steady solution of the barotropic vorticity equation. This instability is known as *barotropic instability* [80]. Left unaltered, a steady zonal jet will persist indefinitely, but if small perturbations develop due to numerical error, they will be amplified by the instability and the initially steady jet will develop into a Rossby wave pattern [80, section 7.7], [52, section 6.5.1]. We develop two test cases based on this idea: a steady zonal jet which we use to examine how well our model maintains a steady state, and a perturbed zonal jet which we use as a model of developing Rossby waves. We model the jet's zonal velocity as a Gaussian function of latitude, and define the test case as

$$\theta'(\lambda) = \theta_0 + \epsilon_P \cos(m\lambda) \tag{4.24a}$$

$$u(\lambda, \theta) = \frac{\pi}{2} \exp\left[-2\beta^2 \left(1 - \cos\theta \cos\left(\theta'(\lambda)\right) - \sin\theta \sin(\theta'(\lambda))\right)\right] \tag{4.24b}$$

$$v(\lambda, \theta) = 0 \tag{4.24c}$$

$$\zeta(\lambda, \theta) = \Big(\cos\theta \Big(2\beta^2(\cos(\theta'(\lambda)) \sin\theta - \sin(\theta'(\lambda)) \cos\theta)\Big) + \sin\theta\Big)\, u(\lambda, \theta). \tag{4.24d}$$

These equations represent a zonal jet whose center lies at latitude $\theta'(\lambda)$; θ_0 is the jet's mean latitude and we introduce a perturbation in latitude with amplitude ϵ_P and longitudinal wave number m. Setting the parameter $\epsilon_P = 0$ defines the steady state test case. β in (4.24b) is a shape parameter; larger values lead to a narrower jet. We set the jet's meridional velocity $v = 0$. Applying the curl operator to equations (4.24b) and (4.24c) gives relative vorticity (4.24d). Latitude profiles of u and ζ for the steady state are shown in figure 4.19.

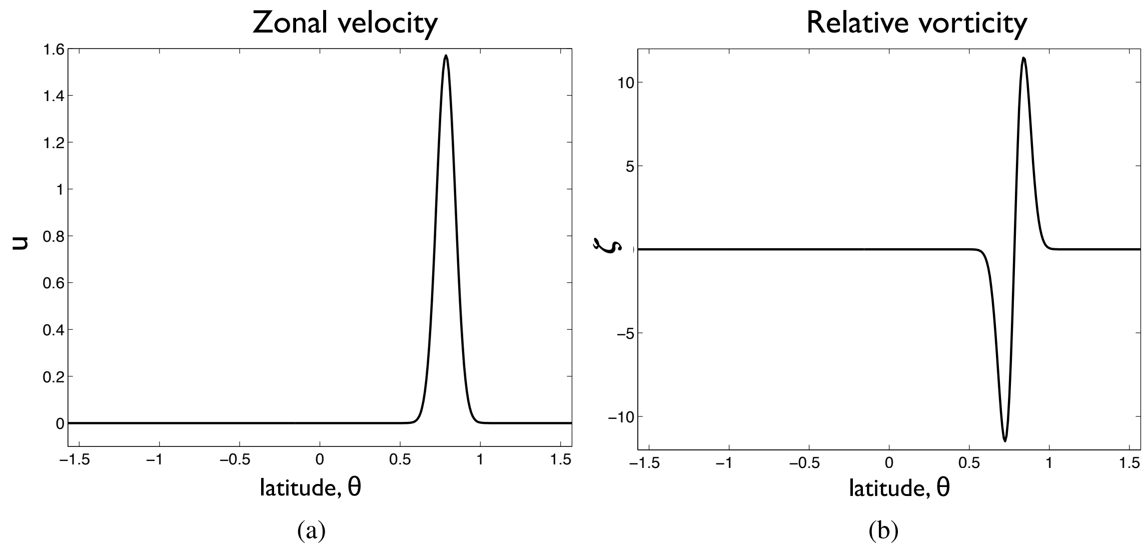


Figure 4.19: Latitude profiles of (a) zonal velocity and (b) relative vorticity of a jet from equations (4.24) with $\theta_0 = \pi/4, \beta = 12$, and $\epsilon_P = 0$.

Velocity and vorticity for a wave number 4 perturbation are shown in figure 4.20.

The velocity and vorticity functions in (4.24) are designed to model the midlatitude polar front jet, which has a typical latitude of 45N. Particles in a jet with maximum velocity of $\pi/2$ from (4.24b) traverse a latitude circle in 4 units of time on the unit sphere. For orientation, we note that the circumference of the 45N latitude circle on the Earth is approximately 28000 km. A four-day circuit at that latitude on the Earth would correspond to a speed of approximately 290 kilometers per hour (kph). Observed jet stream speeds are typically around 100-160 kph, so this test case has the correct order of magnitude for a model on the unit sphere.

4.4.1 Steady zonal jet

Plots of the steady zonal jet on a set of quadrilateral panels with $N = 24576$ are shown in figure 4.21 for $t = 0, 1, 2$. This calculation used $\Delta t = 0.005$ and Lagrangian remeshing every 10 time steps $t_{rm} = \{0.05, 0.01, 0.015, \dots\}$. The jet core shows up as the thin white line separating the positive vorticity (red) from the negative vorticity (blue), and is aligned

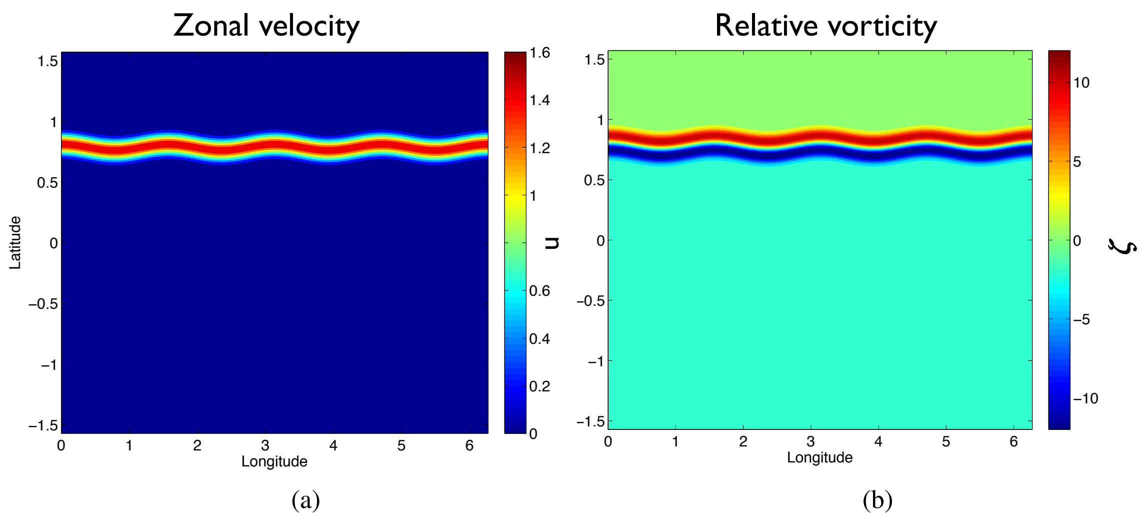


Figure 4.20: (a) and (b) as in figure 4.19 with $\epsilon_P = 0.025$ and $m = 4$.

with the 45N parallel through $t = 2$, which was the final time for this computation. Very small perturbations are beginning to appear at $t = 2$, and appear to have a wave number 4 signal, which is indicative of grid imprinting due to the cubed sphere.

The relative changes in total kinetic energy and total enstrophy plotted in figure 4.22 are defined as

$$R_{KE}(t) = \frac{|KE(t) - KE(0)|}{KE(0)} \tag{4.25a}$$

$$R_Z(t) = \frac{|Z(t) - Z(0)|}{Z(0)} \tag{4.25b}$$

where $KE(t)$ and $Z(t)$ are given by (4.19a) and (4.19b). Total kinetic energy is maintained within ±2% of its initial value and total enstrophy is maintained within ±4% of its initial value over the duration of the computation. The slight growth in kinetic energy and enstrophy at the early times are similar to the RH4 results (figure 4.9). A new pattern of error develops shortly after $t = 1.5$, when both kinetic energy and enstrophy begin to change more rapidly; this is due to stretching of the Lagrangian parameter $\boldsymbol{\alpha}$, which is being interpolated by the Lagrangian remeshing procedure.

Figure 4.23 shows plots of the longitudinal component of the Lagrangian parameter

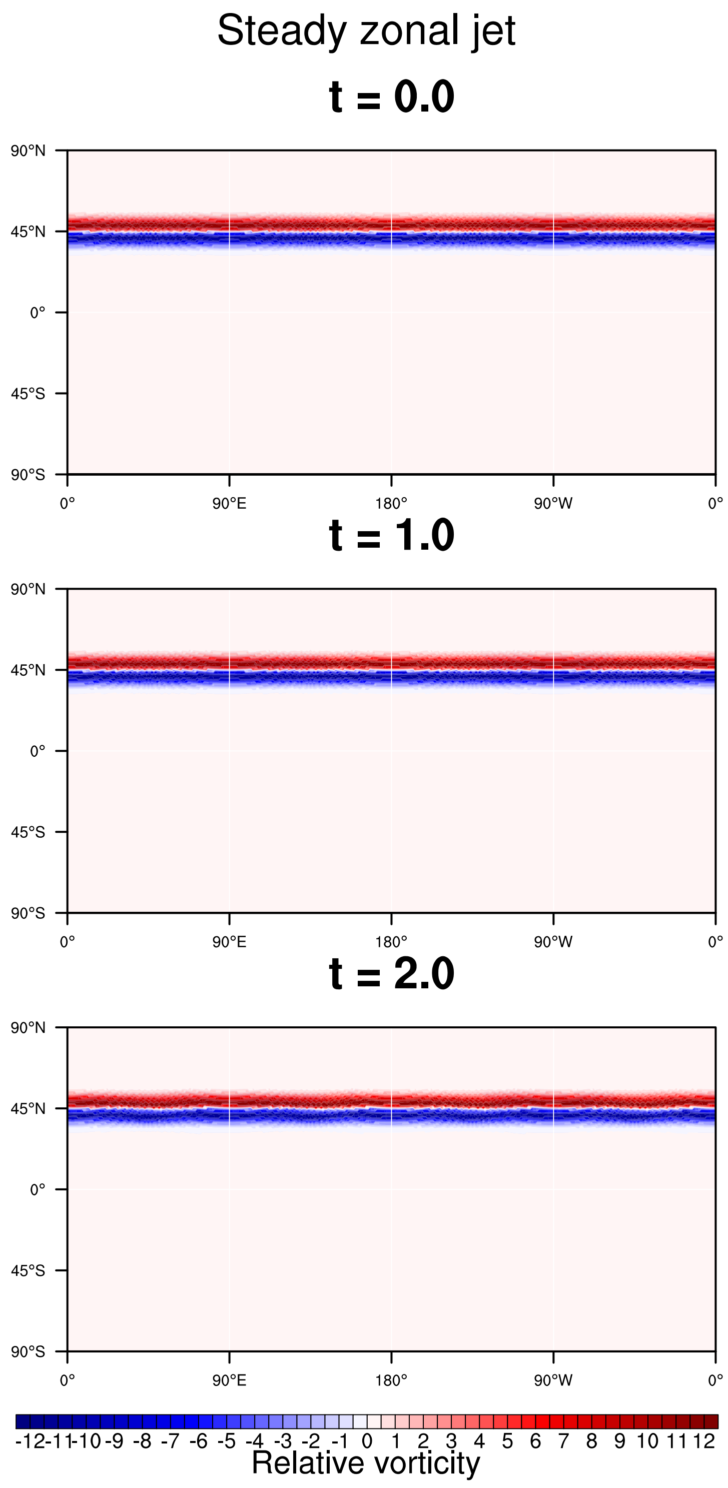


Figure 4.21: Steady zonal jet at $t = 0, 1, 2$ on a uniform set of $N = 24576$ quadrilateral panels, Lagrangian remeshing at intervals of 10 time steps.

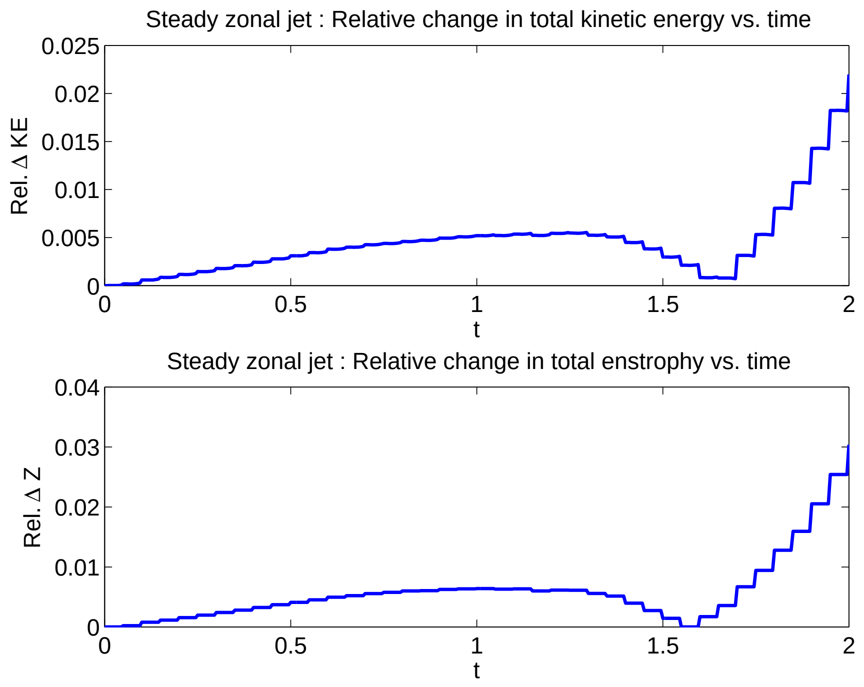


Figure 4.22: Relative changes in total kinetic energy (top) and total enstrophy (bottom) vs. time for a steady zonal jet with $N = 24576$ quadrilateral panels, Lagrangian remeshing at intervals of 10 time steps.

α at $t = 0, 1, 2$. Like the plots of initial latitude shown in the Gaussian vortex test case, the plots of initial longitude provide insight into the interpolation part of the Lagrangian remeshing procedure. At $t = 0$, each interval of 15° degrees longitude has its own color. As t increases, stretching in the Lagrangian parameter due to transport by the jet is apparent along 45N latitude. The jagged, high frequency features near the poles are plotting artifacts not related to our numerical solution. The gradient of α becomes too steep to be interpolated accurately after a certain amount of time has passed, approximately $t = 1.5$ for this computation, based on figure 4.22. At this point in the computation, for this resolution, it is no longer feasible to interpolate the flow map all the way back to $t = 0$ during each remeshing. Changing the remeshing procedure to interpolate to a new Lagrangian parameter defined at intermediate times is a feature that will be implemented in the next version of the LPPM software.

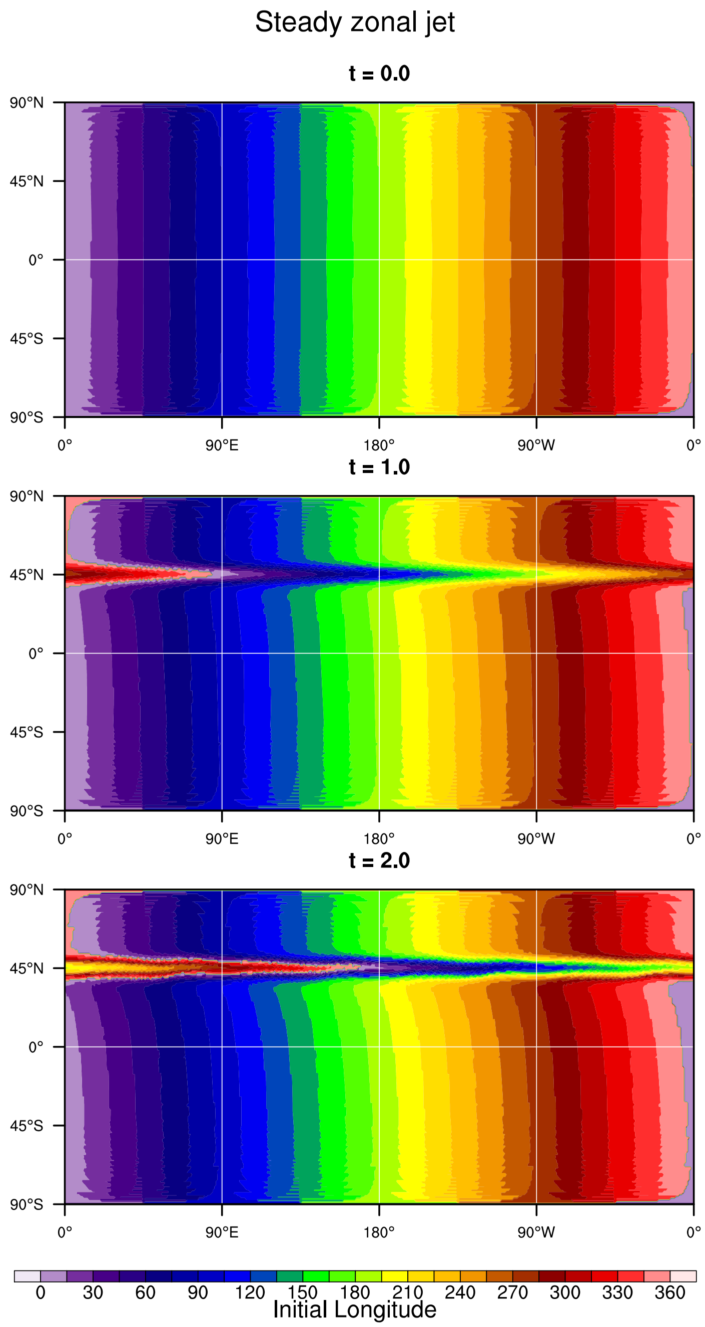


Figure 4.23: Stretching of Lagrangian parameter by steady zonal jet. Colors depict the longitudinal component of α and indicate the initial longitude of each fluid particle. By $t = 2$ fluid in the jet core has traveled halfway around the 45N latitude circle.

Despite the difficulty of interpolating highly stretched α during each remeshing, and the increasing changes beginning to show up in kinetic energy and enstrophy after $t = 1.5$, the $t = 2$ calculation is still an accurate solution of the problem. Recalling that the test case is defined so that fluid particles complete a circumnavigation of the globe along the jet's central latitude in four units of time, we expect to see that particles have traveled through an arc length of 180° from $t = 0$ to $t = 2$. Figure 4.23 shows that this is indeed the case for our solution. The light purple color marks particles that originate in $\lambda_0 \in [0°, 15°)$ and pink marks particles that began in $\lambda_0 \in [345°, 360°)$. At $t = 0$ this boundary lies along 0° and at $t = 2$ this boundary lies at 180° longitude in the jet core at 45N.

The steady zonal jet test case can be thought of as parallel shear flow on the sphere. It is therefore a good test case to examine Voronoi grids' ability to handle shearing, as discussed in section 2.3. While the properties of Voronoi diagrams suggest that a Voronoi grid may do well in such a problem by preventing strongly sheared panels, similar to the untangling method described by [182], in actuality this test case provides further justification for remeshing, as shown next.

Figure 4.24 shows the steady jet test case at $t = 0.1, 0.2, 0.5$, and 1 for a calculation on a Voronoi mesh with $N = 10242$ panels, $\Delta t = 0.01$, and no remeshing. Coloring shows relative vorticity, with red indicating positive (counterclockwise) vorticity and blue indicating negative (clockwise) vorticity. Maximum velocity is in the jet core, which lies along white panels with nearly zero vorticity between the red and blue regions. Active particles are plotted as black dots, and the left column shows the whole sphere's relative vorticity. The right column is a zoomed-in view of the jet core. Voronoi panel edges are plotted as black lines; passive particles lie at their vertices. From $t = 0$ to approximately $t = 0.2$ the Voronoi algorithm maintains a nearly uniform mesh but by $t = 0.2$ the flow's instability is beginning to cause spurious roll-up (indicated by the curved arrows). These roll-up features are caused by numerical errors in particle positions that have been amplified by the flow instability. In this steady state problem, meridional velocity should be zero and roll-up should not occur. Yet by $t = 0.5$ some spiral features caused by amplified error are evident, and by $t = 1$ the jet is hardly recognizable.

It is possible that adaptive refinement combined with a Voronoi mesh might perform better on this problem, and that using a centroidal-Voronoi mesh as in [151, 162, 91] instead of a basic Voronoi diagram might offer another improvement. However, we have already had success using our remeshing algorithm (figure 4.21), so we apply the same approach here. The remeshing procedure for a Voronoi panel set is identical to a quadrilateral or triangular panel set; the new particles are simply particles from a Voronoi mesh instead of a cubed sphere or set of icosahedral triangles. Figure 4.25 shows $t = 1$ for a Voronoi mesh with $N = 10242$ panels that was remeshed with Lagrangian remeshing at $t_{rm} = \{0.2, 0.4, 0.6, 0.8, 1.0\}$. The remeshing procedure has maintained the structure of the steady zonal jet to $t = 1$ much better than the Voronoi grid shown in figure 4.24 which did not use remeshing.

The need to remesh a Voronoi grid, as shown by this test case, makes the Voronoi panels merely another variety of our basic algorithm II.1. Development of a recursive procedure for panel refinement similar to those shown in figures 2.5 and 2.6 is still an ongoing project for Voronoi grids, which must accommodate arbitrary varieties of polygons within the same mesh. As a consequence, we have not yet developed adaptive Voronoi algorithms and the remainder of this thesis uses quadrilateral and triangular panels.

4.4.2 Perturbed zonal jet

The perturbed jet test case provides an opportunity to study the effects of barotropic instability. We test two wave numbers, $m = 4$ and $m = 12$ in (4.24) and examine the development of barotropic Rossby waves on the sphere. Holton [80, section 7.7.1] pro-

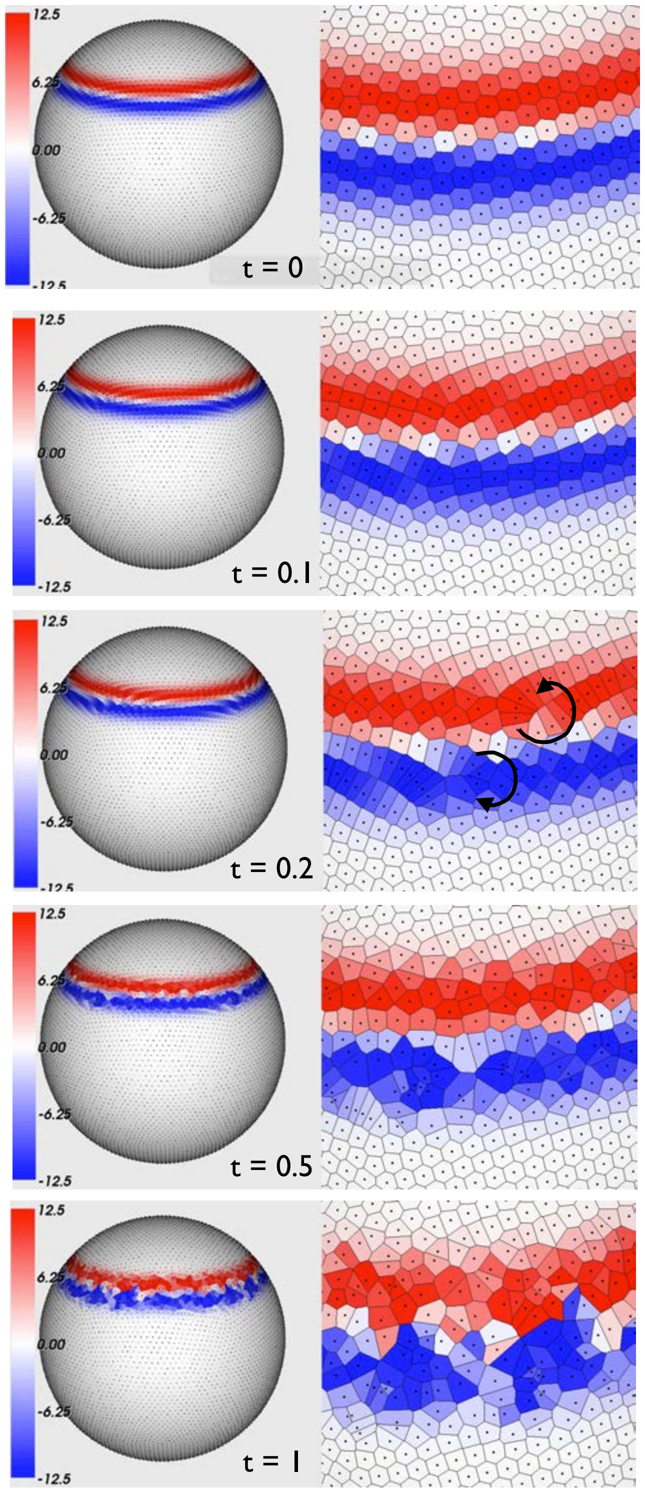


Figure 4.24: Relative vorticity time evolution of a steady zonal jet, $N = 10242$ Voronoi panels, no remeshing. Right column is a close up view centered at latitude $\theta_0 = \pi/4$. Arrows in the $t = 0.2$ frame indicate the direction of spurious roll-up caused by numerical error.

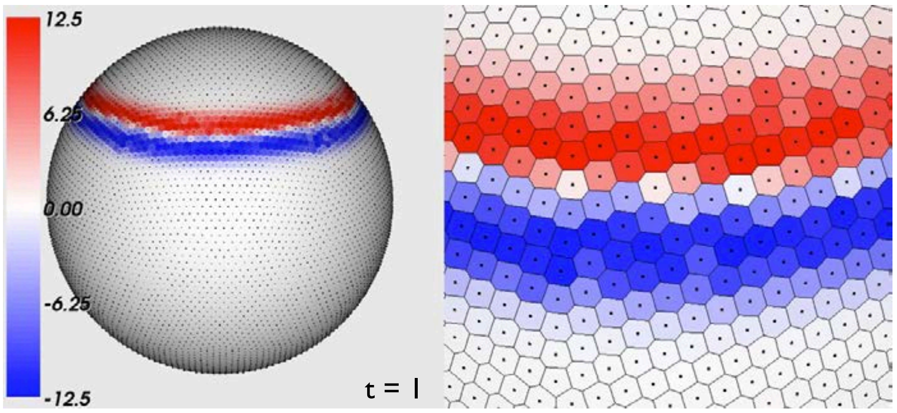


Figure 4.25: Relative vorticity at $t = 1$ for a steady zonal jet on $N = 10242$ Voronoi panels, remeshed at $t_{rm} = \{0.2, 0.4, 0.6, 0.8, 1.0\}$. Compare with $t = 1$ frame from figure 4.24.

vides analysis on the Cartesian β-plane that includes a derivation of the dispersion relation and group velocity for planetary waves. This dispersion relation was also investigated by Haurwitz [77], as it relates to the stationary wave speed (4.16) of a Rossby-Haurwitz wave.

Aside from providing an interesting meteorological example of dispersion, this test case provides another good problem for adaptive refinement. The test case definition (4.24) concentrates most of the vorticity near the jet's central latitude. The majority of the sphere has nearly zero velocity and vorticity, as shown in figure 4.20. Consequently, we are able to represent those areas of the sphere with large panels to save computational resources for the vicinity of the jet where they are most needed.

The first perturbed jet test we perform has initial conditions given by (4.24) with $\epsilon_P = 0.025$ and $m = 4$, figure 4.20. Adaptive refinement was applied to a base uniform mesh with 6144 quadrilateral panels using refinement criteria $\epsilon_1 = 0.002$ and $\epsilon_2 = 0.045$ to give an initial panel set with $N = 15036$ panels. A maximum of four levels of refinement beyond the base mesh was allowed in this calculation. The time step was $\Delta t = 0.00125$, chosen to keep the computation stable for the smallest panels that could possibly be created by the adaptive refinement procedure. Lagrangian remeshing was applied at intervals of 20 time steps.

Figure 4.26 shows four snapshots of this computation, $t = 0, 0.75, 1.5$, and $t = 2$. Coloring in both columns depicts relative vorticity; the left column provides a global view and the right column shows a zoomed-in view with the panel edges plotted as thin black lines. At $t = 0.75$ not much has changed visually from $t = 0$, but the adaptive refinement criteria are beginning to pick up changes, particularly along the equatorial half of the jet. The developing Rossby wave kicked off by the unstable environment and perturbed initial conditions is clearly evident at $t = 1.5$. The adaptive refinement criteria are triggered by a developing anticyclone at the base of the deepening trough. At $t = 2$ the wave has steepened further and developed strong vorticity gradients, particularly in its troughs. We note that between $t = 1.5$ and $t = 2$, the refinement criteria have actually coarsened the mesh in the core of the anticyclone, but retained the highest degree of refinement along its southern extent in a feature that resembles frontogenesis but with negative, rather than positive vorticity.

The smallest panels in figure 4.26 have approximate edge lengths of 0.3°. A uniform mesh at this resolution would require $N = 393216$ panels, while the adaptive refinement criteria (2.8) and (2.10) have enabled a solution with $N \leq 46824$: less than 1/8 the cost of a uniform mesh computation at this resolution.

The next test uses (4.24) with $\epsilon_P = 0.025$ and $m = 12$. Adaptive refinement criteria $\epsilon_1 = 0.008$ and $\epsilon_2 = 0.09$ were applied to a base mesh of 1536 quadrilateral panels at intervals of twenty time steps with $\Delta t = 0.0025$. The short wavelength of this perturbation feels the effects of barotropic instability almost immediately, and the jet begins to roll-up into twelve pairs of vortices as early as $t = 0.25$, shown in figure 4.27. The adaptive refinement criteria resolve the thin filaments that stretch between the developing vortices as t increases. Some finer structures inside the vortex cores are also suggested in the $t = 0.75$ frame. These choices of refinement tolerances and initial resolution have resulted

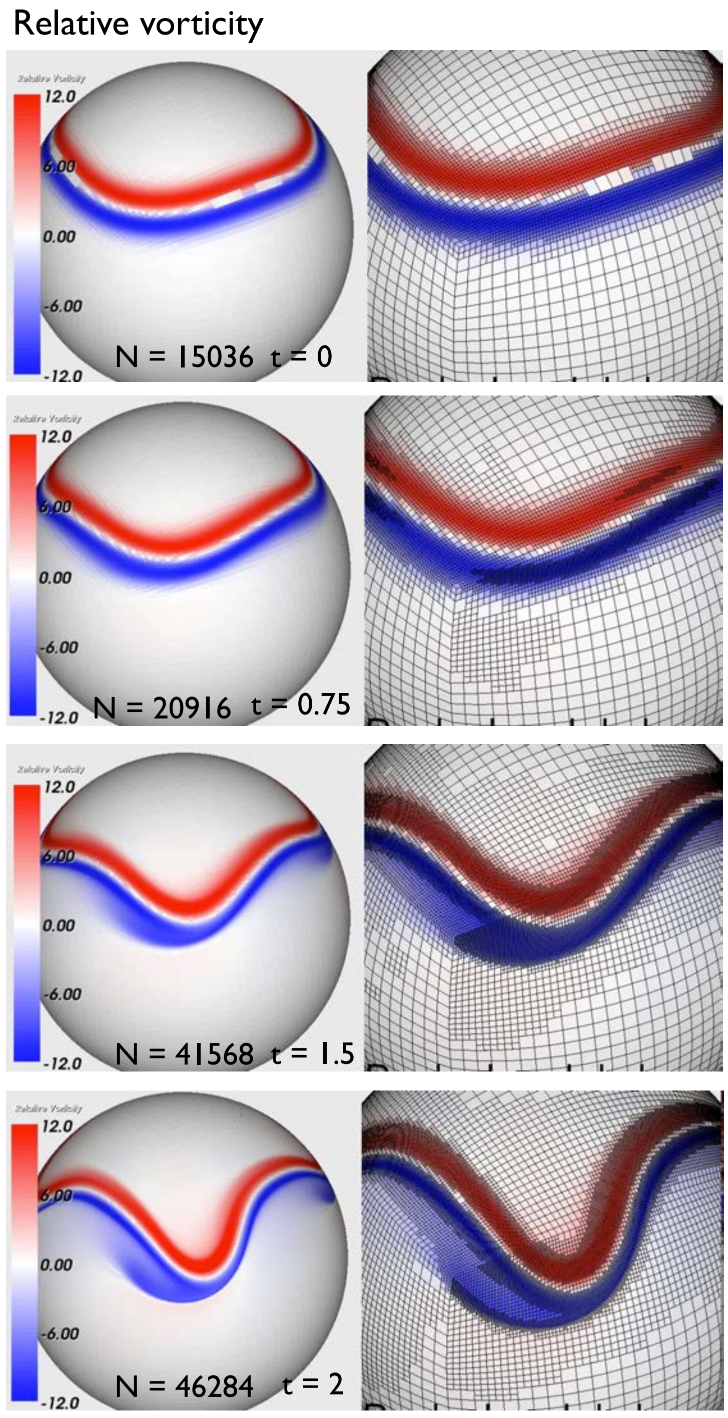


Figure 4.26: Zonal jet with wave number 4 perturbation. Quadrilateral panels with adaptive Lagrangian remeshing at intervals of 20 time steps. Coloring in both columns shows relative vorticity. Right column has a closer viewpoint to show quadrilateral panel edges.

in a very small value of $N = 10596$, given the spatial scales shown in the solution by $t = 0.75$.

4.5 Stratospheric model of breaking Rossby waves

In this section we change the focus to our application: modeling the global circulation of the atmosphere. As noted in the previous section, a barotropic fluid is a suitable model of geophysical motion dominated by vorticity rather than divergence, with negligible thermodynamic effects. Regions of the atmosphere that are typically near these conditions are the upper troposphere and the stratosphere [80, 78]. Stratospheric dynamics are introduced in [80, chapter 12] and a review article is provided by Haynes [78]. In weather forecasting applications, it is typical to consider the troposphere (the near-surface atmosphere, where weather occurs) as decoupled from the stratosphere. The stratosphere tends to vary on a longer time scale than the troposphere and its effects can be incorporated with data input to a weather model instead of having to be simulated directly. But in climate simulations, which by definition seek to simulate long-time weather, stratospheric motion cannot be neglected.

Events known as sudden stratospheric warmings dramatically influence winter weather patterns by disrupting the stratospheric polar vortex that normally persists throughout the winter hemisphere [124, 34]. In its normal state, without a sudden stratospheric warming, flow in the stratosphere over the Earth's winter hemisphere is dominated by a cyclonic polar vortex [78]. This vortex influences the position of the troposphere's polar front jet, which separates cold polar air from more temperate latitudes. During a sudden stratospheric warming event, the stratospheric polar vortex breaks down, and the troposphere's polar front jet develops large amplitude Rossby waves that have wide ranging effects on weather patterns at the surface. Areas within a high amplitude trough of the polar front

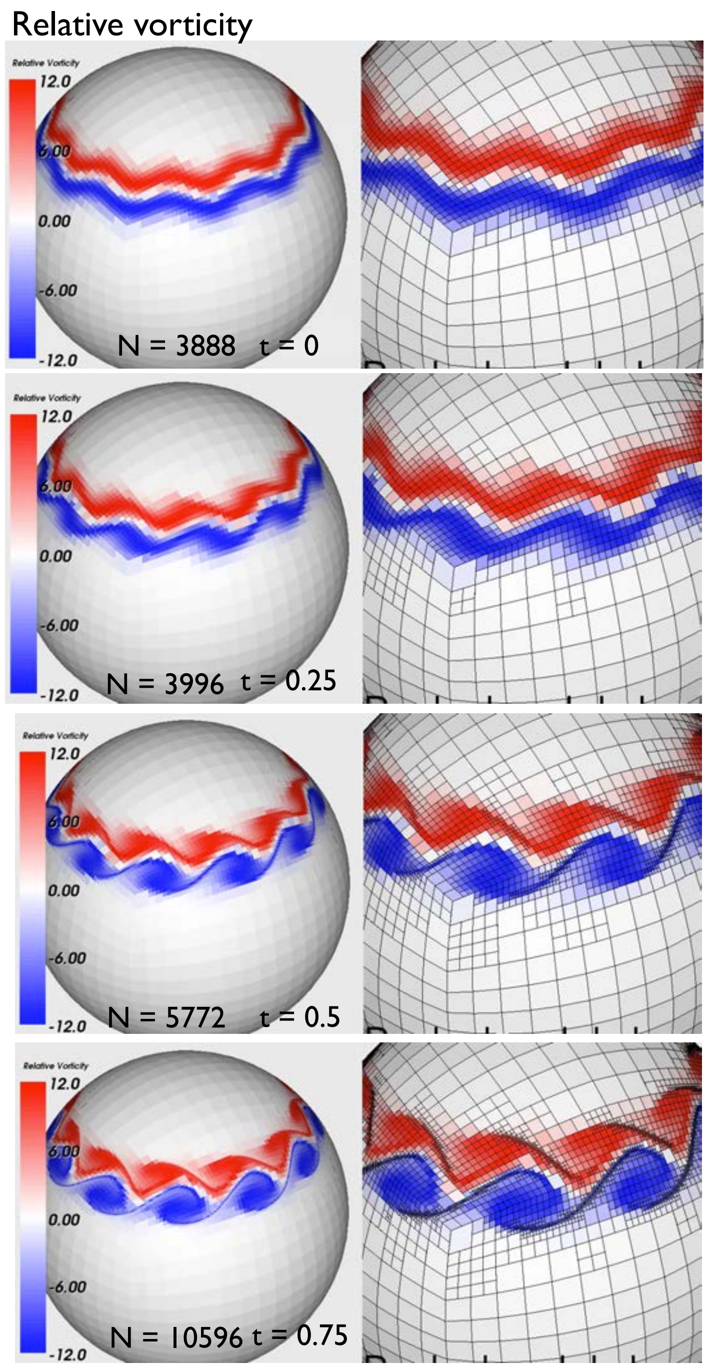


Figure 4.27: Zonal jet a wave number 12 perturbation applied to its central latitude, computed with adaptive Lagrangian remeshing at intervals of 20 time steps. Coloring shows relative vorticity; zoomed-in plots in right column show quadrilateral panel edges.

jet will experience very cold temperatures, while areas within a ridge will experience abnormally warm weather. Historically, sudden stratospheric warmings happened almost periodically, every two years [80]. Within the last decade they have been occurring more frequently, at an annual rate or faster [34]. This increase in frequency is attributed to the effects of climate change, specifically the increase in global average temperature and changes in stratospheric ozone levels [124, 34].

Despite their name, sudden stratospheric warmings can be modeled without thermodynamics using a barotropic model [93, 78]. The stratosphere, unlike the troposphere, has a positive vertical thermal gradient; temperature increases with height in the stratosphere. As a consequence, stratospheric motions are strongly stratified in the vertical. Although sudden stratospheric warmings cause a rapid rise in stratospheric temperatures, due to the strong vertical stratification of this layer of the atmosphere, the resultant motions can still be described well by a barotropic model [78].

Sudden stratospheric warming events are caused by Rossby wave breaking in the troposphere. Breaking planetary waves cause vertical momentum transfer between the troposphere and the stratosphere. From the perspective of the stratosphere, the upper boundary of the troposphere (the tropopause) acts like a lower boundary condition. Rossby wave breaking in the troposphere can be modeled from the stratospheric perspective as a change in the height the tropopause [93, 78]. From a mathematical modeling perspective, the tropopause can be viewed as the stratosphere's bottom topography.

Dynamic models of sudden stratospheric warmings and the breakdown of the winter polar vortex focus on the appearance of a *surf zone* along the periphery of the main vortex [128, 93, 78]. The surf zone is formed by the intrusion of low vorticity fluid into the region originally occupied by the polar vortex, and results in the development of strong vorticity gradients along the periphery of the main vortex and thin filaments of high vorticity similar to the filament that developed in the Gaussian vortex test case (section 4.3). Filaments provide a mechanism for the generation of secondary vortices, depending on how they are perturbed by their surrounding flow. Forcing or shearing may excite the generation of a secondary vortex from a vortex filament. At the same time, the planetary vorticity gradient has a stabilizing effect on the motion and acts to suppress or delay the generation of secondary vortices [78, 80]. Numerical models of stratospheric motion in the presence of a breaking Rossby wave must be able to resolve both the large scale vortices and the small scale filaments without introducing numerical error that could be amplified by the instability inherent to the fluid equations.

The methods developed in this thesis are ideally suited to model this type of flow. A one-layer model based on the barotropic vorticity equation is provided by Juckes & McIntyre [93]. Potential vorticity Q is conserved materially in this model, where

$$Q = \omega + F \tag{4.26}$$

is the sum of absolute vorticity and a forcing function F designed to simulate the effect of a breaking tropospheric Rossby wave incident to the tropopause. We apply LPPM (3.26) to this problem by recovering the relative vorticity from Q

$$\zeta = Q - f - F \tag{4.27}$$

where f, as before, is the Coriolis parameter. The model is defined by the equations

$$\frac{dQ}{dt} = 0 \tag{4.28a}$$

$$f(\boldsymbol{x}, t) = 2\Omega z(t) \tag{4.28b}$$

$$F(\lambda, \theta, t) = F_0 A(t) B(\theta) \cos\lambda \tag{4.28c}$$

$$\frac{d}{dt}\boldsymbol{x}(t) = -\frac{1}{4\pi}\int_S \frac{\boldsymbol{x} \times \tilde{\boldsymbol{x}}}{1 - \boldsymbol{x} \cdot \tilde{\boldsymbol{x}}}\Big(Q(\tilde{\boldsymbol{x}}, 0) - f(\tilde{\boldsymbol{x}}, t) - F(\tilde{\boldsymbol{x}}, t)\Big)\, dA(\tilde{\boldsymbol{x}}), \tag{4.28d}$$

which is simply (3.26) with ζ replaced by (4.27) in the Biot-Savart integral. The forcing of stratospheric motion caused by breaking Rossby waves tend to be geographically stationary, which guides the form of the model forcing function [78]; $F(\lambda, \theta, t)$ is a stationary, quasi-topographic function with longitudinal wave number 1. It is made up of a constant, $F_0 = 0.6\Omega$ which keeps the maximum forcing amplitude at 0.3 times the maximum planetary vorticity, which represents a minor warming event [93]. The functions $A(t)$ and $B(\theta)$ are plotted in figure 4.28 and defined as

$$A(t) = \begin{cases} \frac{1}{2}\left(1 - \cos\frac{\pi t}{T_p}\right) & t \in [0, T_p) \\ 1 & t \in [T_p, T_f - T_p) \\ \frac{1}{2}\left(1 - \cos\left(\frac{\pi(t-(T_f-T_p))}{T_p} + \pi\right)\right) & t \in [T_f - T_p, T_f) \end{cases} \tag{4.29a}$$

$$B(\theta) = \begin{cases} \frac{\tan^2\theta_1}{\tan^2\theta}\exp\left(1 - \frac{\tan^2\theta_1}{\tan^2\theta}\right) & \theta \in (0, \pi/2] \\ 0 & \theta \in [-\pi/2, 0] \end{cases} \tag{4.29b}$$

where the parameter T_p defines the beginning of the Rossby wave breaking event's maximum intensity and T_f is the final time [93, 92]. The function $B(\theta)$ has a parameter θ_1 which represents the latitude where the breaking Rossby wave is centered. This choice of $B(\theta)$ corresponds to the dashed curves in figure 2 (b) of [93], not the function the authors actually used to produce the plots shown in that paper [92]. Therefore we do not expect our results to exactly reproduce the plots shown in [93]; however, we do expect that our results will be qualitatively similar to those presented by Juckes & McIntyre. A plot of the forcing function $F(\lambda, \theta, t)$ at its maximum amplitude (in time) is shown in figure 4.29.

We model the polar vortex with the relative vorticity for the steady zonal jet (4.24) with a slightly different amplitude, which has two parameters θ_0 and β.

(4.29c)

$$\zeta_0(\lambda, \theta) = \pi\left(\cos\theta\left(2\beta^2(\cos\theta_0\sin\theta - \sin\theta_0\cos\theta)\right) + \sin\theta\right)e^{-2\beta^2(1-\cos\theta\cos\theta_0-\sin\theta\sin\theta_0)}$$

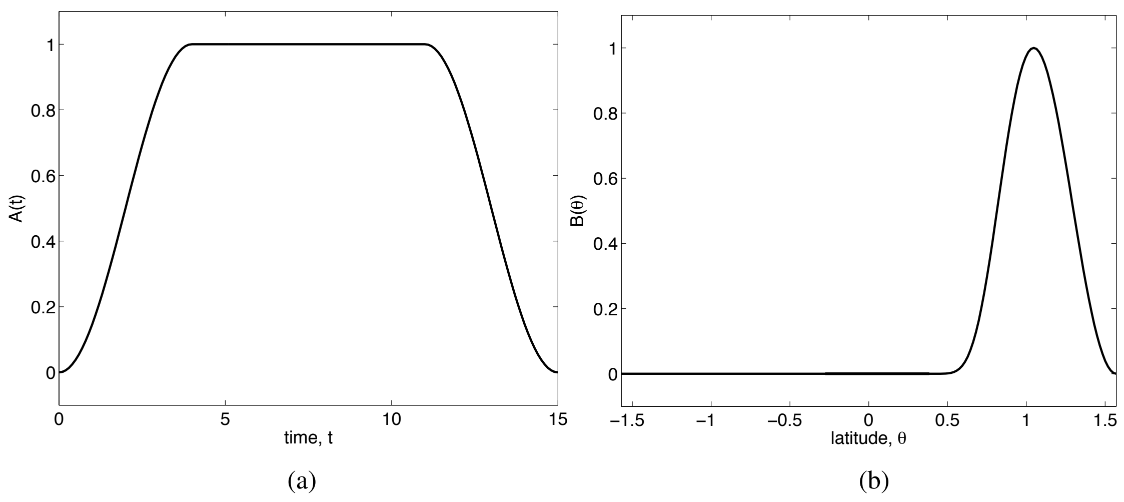


Figure 4.28: Forcing functions for the stratosphere model [93, 92]. (a) Time component of F: $A(t)$. (b) Latitude component of F: $B(\theta)$.

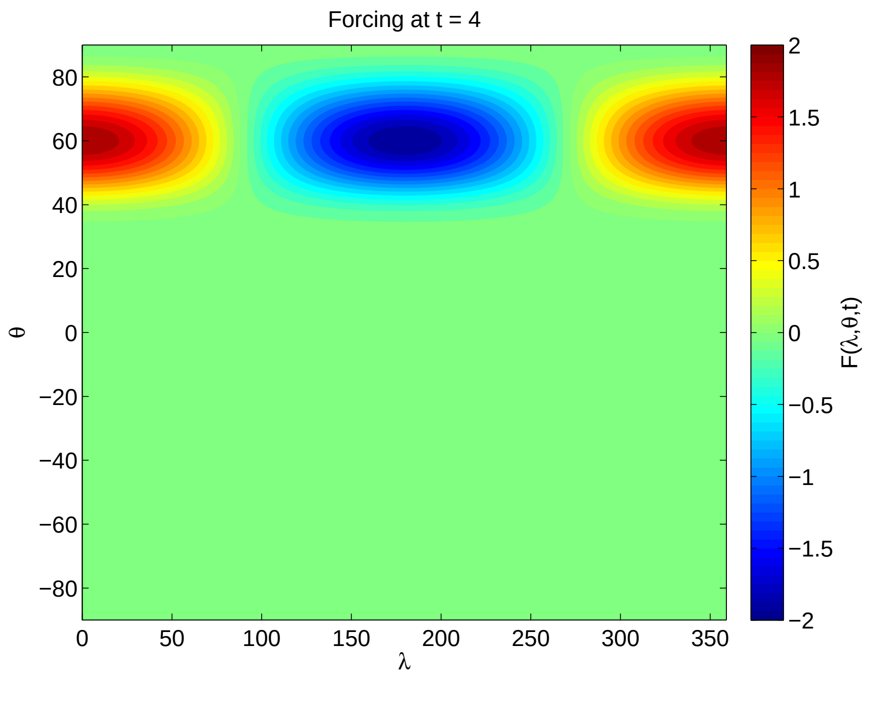


Figure 4.29: The forcing function $F(\lambda, \theta, t)$ at $t = 4$ with $T_p = 4, \theta_1 = \pi/3$. This quasi-topographic function models the vertical transfer of momentum from the troposphere to the stratosphere due to a breaking Rossby wave incident on the tropopause from below [93].

The amplitude of ζ_0 is chosen so that the maximum absolute vorticity is approximately 1.5 times the maximum planetary vorticity [93].

The initial value for the material invariant Q is simply the absolute vorticity, since $F(\lambda, \theta, 0) = 0$,

$$Q(\lambda, \theta, 0) = \zeta_0(\lambda, \theta) + 2\Omega \sin\theta. \tag{4.30}$$

The statement of this problem is completed by defining parameters in a similar manner as [93].

$$\theta_0 = \frac{15\pi}{32} \tag{4.31a}$$

$$\theta_1 = \frac{\pi}{3} \tag{4.31b}$$

$$\beta = \frac{3}{2} \tag{4.31c}$$

$$T_p = 4 \tag{4.31d}$$

$$T_f = 15 \tag{4.31e}$$

These parameters yield an initial vorticity distribution similar to the one used in [93], which was not defined analytically and therefore is not exactly reproducible [92].

The authors of [93] note that although Q is a material invariant in this model, the Eulerian spectral method they employed to solve the equations required the addition of a hyper-diffusion operator to maintain stability, so the equation they actually solve instead of (4.28a) is

$$\frac{d}{dt} Q(\boldsymbol{x}, t) = \nu \nabla^6 (Q(\boldsymbol{x}, t) - Q(\boldsymbol{x}, 0)) \tag{4.32}$$

where the diffusion coefficient ν is chosen to be as small as possible while still producing smooth numerical results. With vortex methods and the algorithms we have developed in this thesis, we can solve (4.28) directly, without having to add a diffusion or hyper-diffusion operator.

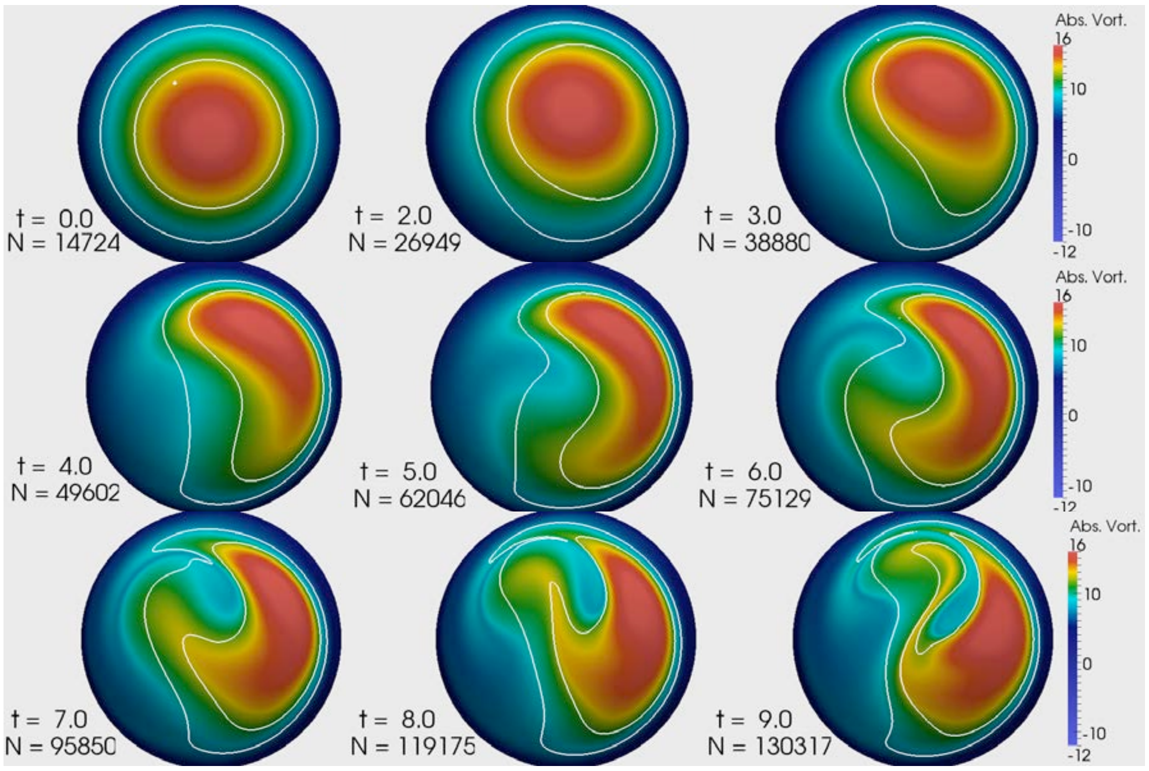
We run the problem with an initial mesh of 6144 quadrilateral panels and adaptive refinement tolerances $\epsilon_1 = 0.00004$ and $\epsilon_2 = 0.023$, with $\Delta t = 0.005$. Absolute vorticity $\omega = Q - F$ is plotted in figure 4.30 along with two material contours of potential vorticity, $Q = 8$ and $Q = 12$ from a point of view looking down at the north pole for times $t \in \{1, 2, \ldots, 9\}$. These Q contours approximately correspond to the material particles advected during the computations by [93]. Relative vorticity is plotted in figure 4.31. Adaptive meshes associated with this calculation are shown in figure 4.32.

The surf zone is evident as a small bend in both Q contours in the $t = 3$ frame of figure 4.30. Vorticity gradients surrounding this feature steepen and develop into stretching filaments particularly evident at $t \geq 6$. The main vortex wraps around the surf zone and sheds a small secondary vortex at $t = 8$ and $t = 9$.

Viewing the relative vorticity plots in figure 4.31, we associate the surf zone with an intensifying anticyclone that covers the pole at $t = 4$ and $t = 5$. What appears as the development of a secondary vortex in plots of absolute or potential vorticity, appears in plots of relative vorticity to be a developing vorticity tripole [8], especially at $t = 5, 6, 7$.

Based on the simplifications inherent to a barotropic model, and the form of the forcing function which was guided more by a desire for convenience in modeling rather than exact realism [93], we should not expect that this model or our results would serve as a forecasting tool. However, we do expect that this model and the LPPM solutions will accurately capture the qualitative behavior of the polar vortex breakdown associated with sudden stratospheric warnings. Haynes emphasizes that as a winter stratospheric polar vortex breaks down, it often forms two smaller cyclonic vortices along the periphery of the original polar vortex [78], which suggests a tripole structure similar to the one that develops in our model.

A study of an actual sudden stratospheric warming event in the southern hemisphere



(a) (b)

Figure 4.30: Stratospheric model of a breaking Rossby wave. Colors show absolute vorticity ω; contours of potential vorticity $Q = 8$ and $Q = 12$ are shown in white.

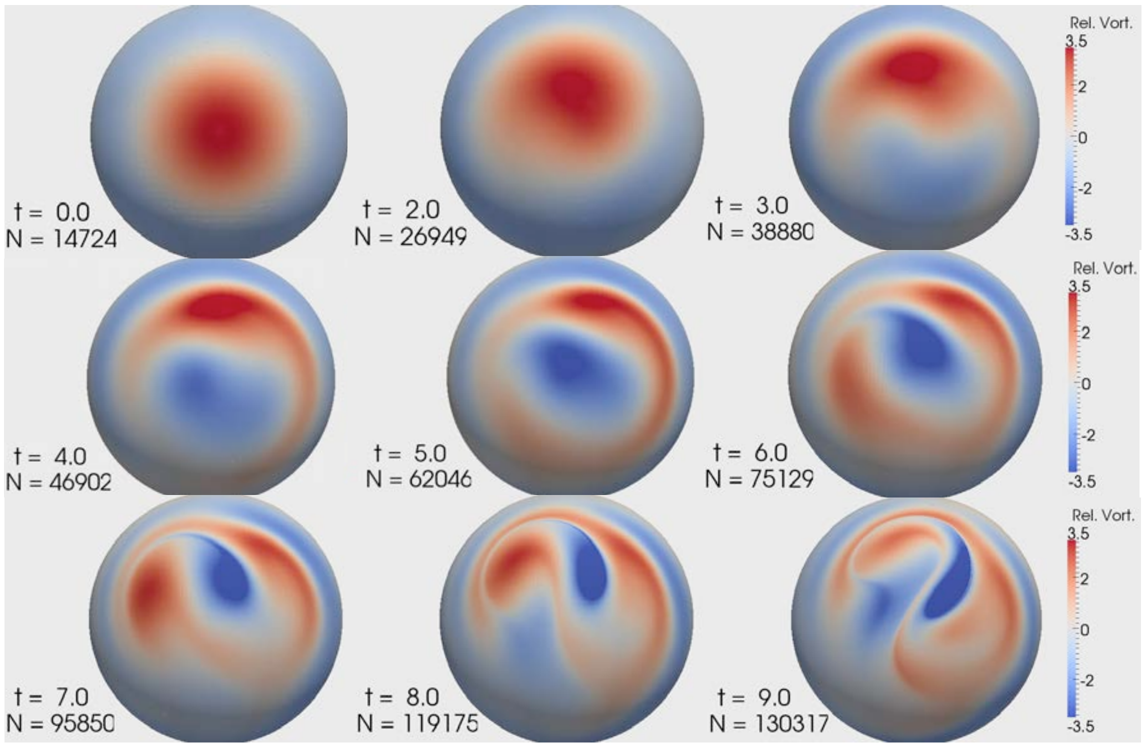


Figure 4.31: Relative vorticity plots from the stratosphere model (4.28). The polar vortex is initially displaced, from $t = 0$ to $t = 3.0$, then breaks down into two smaller cyclonic vortices and an anticylonic vortex to form a tripole structure.

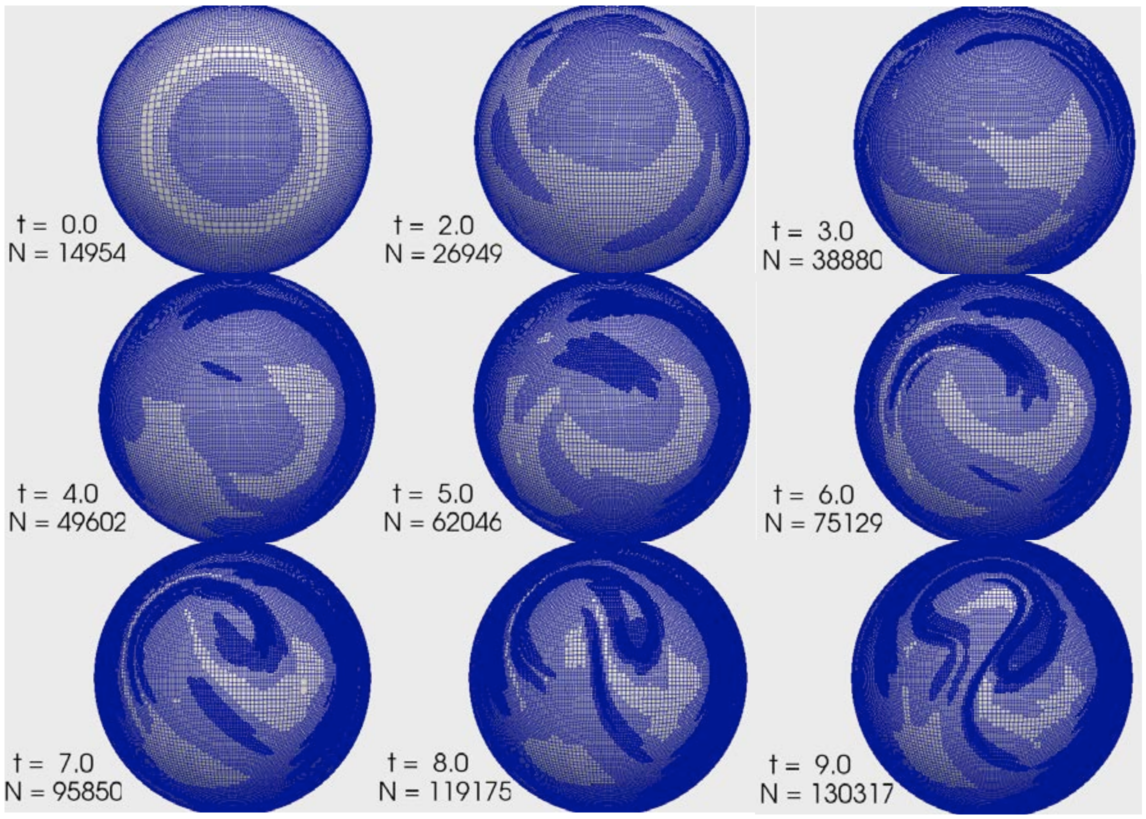


Figure 4.32: Adaptive meshes associated with the computations shown in figure 4.31.

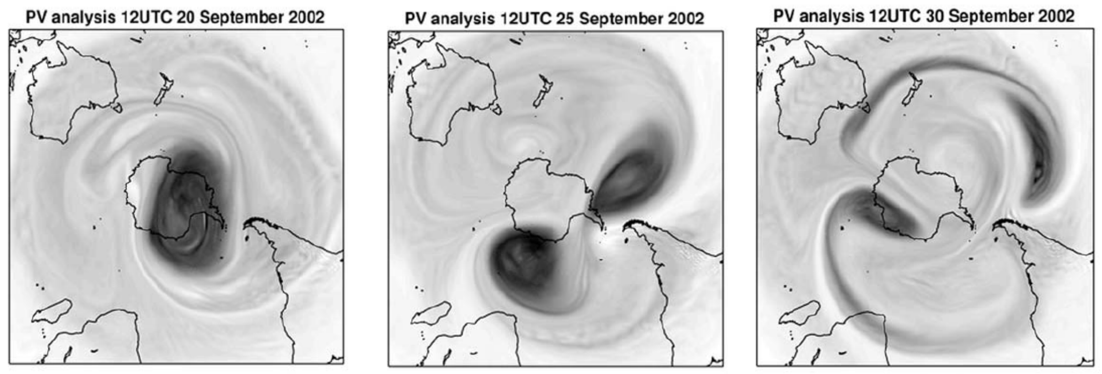

PV analysis 12UTC 20 September 2002
PV analysis 12UTC 25 September 2002
PV analysis 12UTC 30 September 2002

Figure 4.33: Potential vorticity on the 850K isentropic surface over Antarctica shows breakdown of the southern hemisphere's winter stratospheric polar vortex during a sudden stratospheric warming event in September 2002. From [161].

winter 2002 shows the polar vortex break down into a tripole vortex similar to the results of the computation in figure 4.31. Plots of potential vorticity along the 850K isentropic surface from ECMWF reanalysis data are provided by [161] and reproduced in figure 4.33. Data from the 2009 northern hemisphere stratospheric warming are shown in figure 4.34 [135]. The observed data correspond nicely to our results; the potential vorticity depicts the polar vortex breaking down into a tripole structure, with two small cyclonic vortices and one central anticyclone.

A tripole vortex is a particularly interesting structure, and it is somewhat surprising that one should have shown up in this model. Tripole vortices were once thought to be unstable states; it was suggested that a perturbed monopole vortex (such as the polar vortex in the stratosphere model) quickly relaxes back to an axisymmetric state [17, 11]. The hypothesis that a perturbed monopole quickly returns to axisymmetry was based on analysis of a linear model, and other studies of the full nonlinear equations showed that it was not true for large perturbations [48, 155]. For a more detailed summary of the discussion regarding tripole vortices, see the more recent paper by Barba & Leonard [8]. Barba & Leonard cite several numerical studies that have exhibited tripole vortices, as well as rotating tank experiments

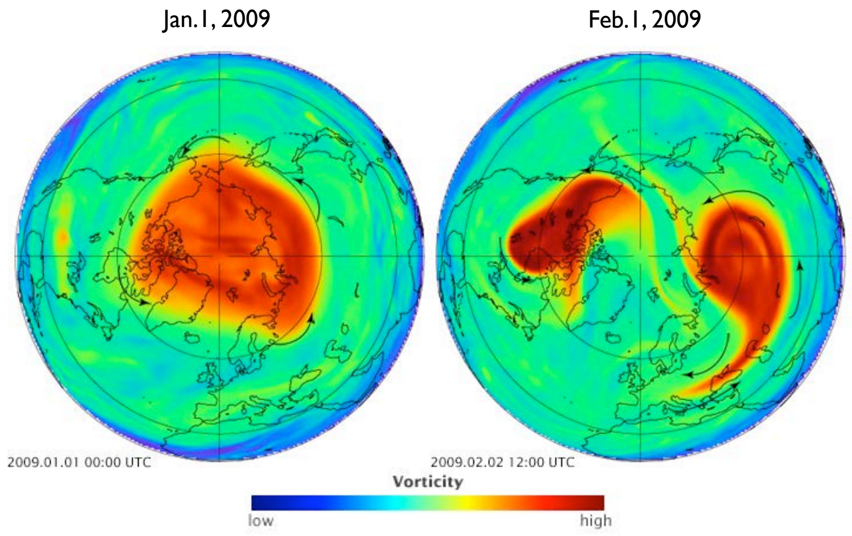


Figure 4.34: NASA presentation of stratospheric vorticity data from January and February 2009 [135]. Abnormally cold weather and large amounts of snow fell across continental Europe due to the perturbed polar vortex that year.

that have observed them. The existence of tripole vortices in rotating tank experiments suggests that tripole vortices should be common features of geophysical fluids, which are also rotating. However, Barba & Leonard cite only one study from 1992 [143] as documented evidence of a naturally occurring tripole vortex; that study found one in the oceanic currents of the Bay of Biscay. The appearance of tripole vortices in our model of a perturbed polar vortex, as well as in observations of sudden stratospheric warmings [161, 135] suggests that tripole vortices may be prominent features of sudden stratospheric warming events. Clearly, more data are needed to verify such a conjecture; the similarities between observations and the simple barotropic model solved here invite further study.

CHAPTER V

Advection Equation

The advection equation is of such fundamental importance to the dynamical core community that despite is seeming simplicity, it still warrants detailed study [52, 109]. Tracers to the dynamical core community are any quantity that is transported by the fluid flow; examples are suspended particles of air pollutants such as radioactive particles or volcanic ash, moisture variables such as water vapor and suspended liquid water, particulate cloud nuclei, and carbon or other chemical species important to atmospheric or oceanic chemical processes. Advected tracers can also be dynamic quantities like absolute vorticity in the barotropic vorticity equation, potential vorticity in the shallow water and primitive equations, or thermodynamic quantities in more complex equation sets. Tracer transport is important to the short time scales of weather forecasts, where moisture variables directly impact cloud physics. Accurate tracer advection is perhaps even more important to climate models which include more coupling between ocean, atmosphere, land, ice, chemistry and vegetative models as they develop. Each component of a climate model has its own collection of tracers that must be advanced in time.

The 2D advection equation may be thought of as a special case of the numerical methods we have already derived. From the perspective of the flow map discretizations developed in chapter II, the only difference lies in where the velocity comes from; for the barotropic vorticity equation, we calculated $\boldsymbol{u}$ using the Biot-Savart integral. For advection equation problems, fluid velocity $\boldsymbol{u}$ is prescribed from an external source.

Advection is often the first test conducted by a developing dynamical core; it is listed as test case 1 in the dynamical core test cases introduced by Williamson *et al.* [187]. Their test case 1 advects a tracer distribution in the shape of a cosine bell once around the sphere using the velocity field that corresponds to solid body rotation. The test is designed to challenge Eulerian models, which must discretize the advective derivative, by altering the axis of rotation to force the tracer field over the most problematic regions of their spatial discretizations. For example, models that use a latitude-longitude grid run the test case so that it transports the cosine bell over both poles and cubed sphere grids must transport the bell over the cube's vertices.

In a Lagrangian model, advection due to such a simple velocity field can be satisfied exactly, hence this is not a fair measure of a numerical method's success. As a consequence, we presented the solid body advection problem as a test case for the barotropic vorticity equation in section 4.1, rather than the advection equation. A better series of advection tests for a Lagrangian method like the one we develop in this thesis are the deformational flow tests presented in [131] and [105]. We use these tests to evaluate our method as an advection scheme, and to understand the effects of remeshing in an environment isolated from the computation of velocity. In particular, we compare the results of Lagrangian remeshing with low-order piecewise linear interpolation to the cubic interpolation provided by SSRFPACK, and we compare the results of direct remeshing to Lagrangian remeshing with the same interpolation method.

5.1 Lagrangian form

The advection equation is very simply stated in Lagrangian form; it is given by the flow map (2.1),

$$\frac{d}{dt}\boldsymbol{x}(\boldsymbol{\alpha},t) = \boldsymbol{u}(\boldsymbol{x}(\boldsymbol{\alpha},t),t) \tag{5.1a}$$

$$\boldsymbol{x}(\boldsymbol{\alpha},0) = \boldsymbol{\alpha} \tag{5.1b}$$

$$\frac{d}{dt}\phi(\boldsymbol{x}(\boldsymbol{\alpha},t)) = 0, \tag{5.1c}$$

where mixing ratio of the ϕ is the tracer being transported.

The particle-panel discretization applied to (5.1) results in our numerical method for the advection equation

$$\frac{d}{dt}\boldsymbol{x}_j(t) = \boldsymbol{u}(\boldsymbol{x}_j(t),t) \quad j = 1,\ldots,N \tag{5.2a}$$

$$\boldsymbol{x}_j(0) = \boldsymbol{\alpha}_j \tag{5.2b}$$

$$\phi_j = \phi(\boldsymbol{\alpha}_j) \tag{5.2c}$$

$$\frac{d}{dt}\boldsymbol{y}_i(t) = \boldsymbol{u}(\boldsymbol{y}_i(t),t) \quad i = 1,\ldots,M \tag{5.2d}$$

$$\boldsymbol{y}_i(0) = \boldsymbol{\alpha}_i \tag{5.2e}$$

$$\phi_i = \phi(\boldsymbol{\alpha}_i) \tag{5.2f}$$

where $\boldsymbol{x}_j$ are active particles, $\boldsymbol{y}_i$ are passive particles, and $\phi_{i,j}$ are the tracer quantities carried by their corresponding particles.

5.2 Test cases and results

The tests defined by [131] and [105] use three tracer fields for ϕ. The first is a C^∞ pair of Gaussian hills used to determine the order of convergence for numerical error as spatial resolution is refined. The second is a C^1 set of cosine bells, used to provide a more realistic

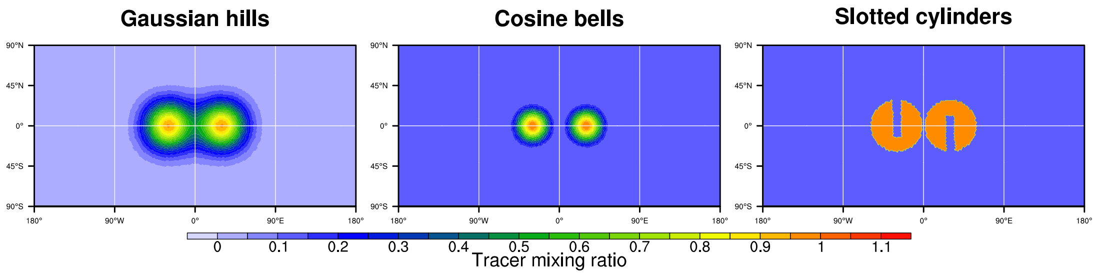


Figure 5.1: Three tracer fields used for deformational flow tests of advection schemes from [131, 105].

test of the advection scheme. A set of C^0 slotted cylinders provide the third tracer field. The slotted cylinders are used to measure advection schemes' abilities to preserve shapes and non-smooth data. These three tracer fields are shown in figure 5.1.

Results from a variety of modeling groups using the tests from [131] were presented at the NCAR Tracer Transport Workshop in March, 2011. Discussions at that workshop led to the updated test cases in [105]. Results from the updated test cases for a diverse collection of advection schemes including LPPM will appear in [106]. In this section we summarize the points most relevant to the development of our Lagrangian Particle-Panel Method (LPPM); the full results from all test cases in [105] are included in the forthcoming results paper [106].

The tests define two deformational velocity fields: one with zero divergence and one with nonzero divergence. The non-divergent wind field has zonal and meridional velocity components given by [105, equations 18,19], which we restate here,

$$u(\lambda,\theta,t) = \frac{10}{T}\sin^2\left(\lambda - \frac{2\pi t}{T}\right)\sin(2\theta)\cos\left(\frac{\pi t}{T}\right) + \frac{2\pi}{T}\cos\theta, \tag{5.3a}$$

$$v(\lambda,\theta,t) = \frac{10}{T}\sin\left(2\left(\lambda - \frac{2\pi t}{T}\right)\right)\cos\theta\cos\left(\frac{\pi t}{T}\right). \tag{5.3b}$$

Both velocity fields are reversible with period T, so that the tracer distribution at $t = 0$ is identical to $t = T$, for straightforward error computations. In each test case, the final tracer distribution's exact solution is identical to its initial condition. Additionally, the form

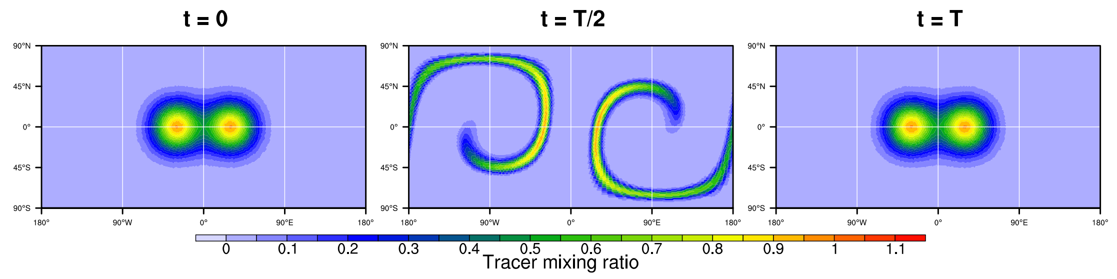


Figure 5.2: Test case 1 from [105]. The Gaussian hills tracer is advected by the non-divergent wind field (5.3). The tracer is deformed and stretched into filaments from $t = 0$ to $t = T/2$, then the flow reverses and the tracer returns to its initial configuration by $t = T$. Plots shown use LPPM, $N = 20480$ triangular panels, $\Delta t = 0.0125$, Lagrangian remeshing with $R_I = 20$.

of each wind field is designed to compress and deform the tracers' initial configurations into smaller scale filaments, with the maximum deformation occurring at $t = T/2$. After $t = T/2$ the flow reverses itself and returns to the initial conditions by $t = T$, as shown in figure 5.2.

As an initial test, we verify that our algorithm is working by turning off remeshing and integrating equation (5.2) with the non-divergent velocity field from $t = 0$ to the final time, $T = 5$. The reversible velocity field implies that particle positions $\boldsymbol{x}(\boldsymbol{\alpha}, T)$ should be equal to their initial position $\boldsymbol{\alpha} = \boldsymbol{x}(\boldsymbol{\alpha}, 0)$. Maximum position error is plotted at the final time against Δt in figure 5.3, and is computed as

$$E_{\boldsymbol{x}} = \max_j \left|\boldsymbol{\alpha}_j - \boldsymbol{x}_j(T)\right|. \tag{5.4}$$

This figure shows the importance of carrying out even simple tests like this one. The red curve shows error saturating near 10^{-6}, which suggested that there may be a single-precision bug in code that was meant to compute in double-precision. The blue curve shows the improvement after identifying and fixing the bug. Both curves verify fourth order convergence as $\Delta t \to 0$, which we expect from the Runge-Kutta time stepping scheme.

All computations in this chapter used uniform (constant N) meshes without adaptive refinement, in order to compare with the other modeling groups participating in the transport

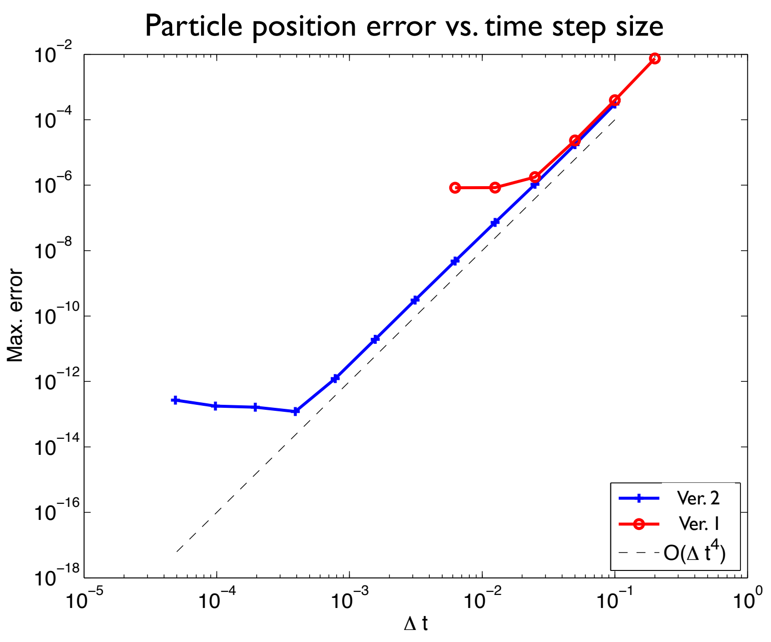


Figure 5.3: Maximum particle position error plotted vs. time step size at $t = T$. First trial (red curve) exposed a single precision bug. Second trial (blue) shows the improvement after fixing the bug.

workshop. At the beginning stages of LPPM's development, piecewise linear interpolation provided the new values of α at Lagrangian remeshing steps. We present results comparing remeshing with piecewise linear interpolation method to the current remeshing procedure which relies on the cubic interpolation provided by SSRFPACK below.

In order to accommodate the divergent velocity field, panel areas are recalculated at each time step. Thus, while particle tracer values $\phi_{i,j}$ are constant between remeshing steps, panel areas A_j are not; this implies that mass will not be exactly conserved. The total mass

$$M(t) = \sum_{j=1}^{N} \phi_j A_j(t) \tag{5.5}$$

varies slightly in time. The magnitude of these changes decreases as the number of panels N is increased. Relative change in total mass $M(t)$ is calculated as

$$R_M(t) = \frac{|M(t) - M(0)|}{M(0)}. \tag{5.6}$$

We plot $\max_{0 \le t \le T} R_M(t)$ in figure 5.4 for various N. The plot shows that changes in total mass converge to zero at a rate proportional to $\Delta\lambda^2$, where $\Delta\lambda$ is the average arc length of panel edges. This is equivalent to a convergence rate proportional to $1/N$, and verifies theoretical expectations of midpoint rule quadrature.

Changes in the total mass integral are due to remeshing and to approximating panel edges as great circles during the calculation of panel areas (figure 2.19). This interpretation suggests that this error could be reduced by adding resolution to panel edges. Adding a passive particle to each edge's midpoint is one method of accomplishing this. Instead of one great circle from origin to endpoint, each edge is computed as two great circles; this doubles the number of sub-triangles in the triangular decomposition of each panel (figure 1.7). The improvement in relative change in total mass due to the addition of passive particles at edge midpoints is shown in figure 5.5. These computations did not use remeshing, in order to isolate the effect of adding resolution to panel edges. While using 3 passive particles per edge is a definite improvement over edges with only endpoints, it would require changing the data structures and point query algorithm presented in chapter II. Adaptive refinement has not yet been applied to this suite of test problems and may produce a similar gain in accuracy, so we have not pursued increased resolution of panel edges further.

Since submitting our results to [106], we upgraded the interpolation scheme in the remeshing procedure from a piecewise linear method to the cubic Hermite interpolation scheme provided by the SSRFPACK software library [147]. Rerunning test case 1 from [105], which measures tracer error for the Gaussian hills distribution and the non-divergent wind field, showed a large improvement due to the more accurate interpolation method. This comparison is presented in figure 5.6, which shows error convergence in both the l_2 and l_∞ norms for the Gaussian hills tracer at $t = T$ as the number of panels N is increased.

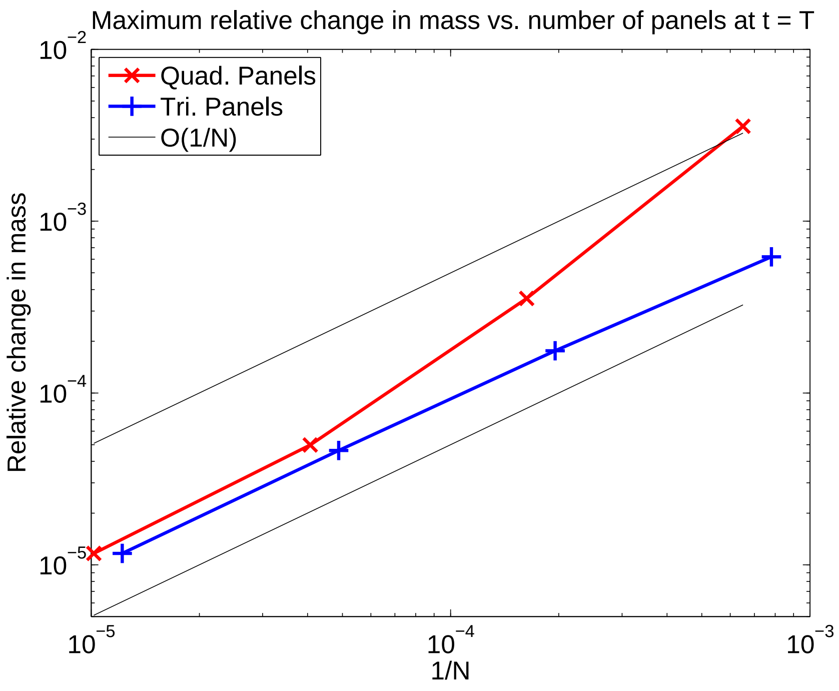


Figure 5.4: Maximum relative change in total mass (5.6) from $t = 0$ to $t = T$ for various N on quadrilateral panels (red) and triangular panels (blue). Black lines have slope $1/N$, where N is the number of panels. All computations used $\Delta t = 0.0125$ and Lagrangian remeshing every 20 time steps.

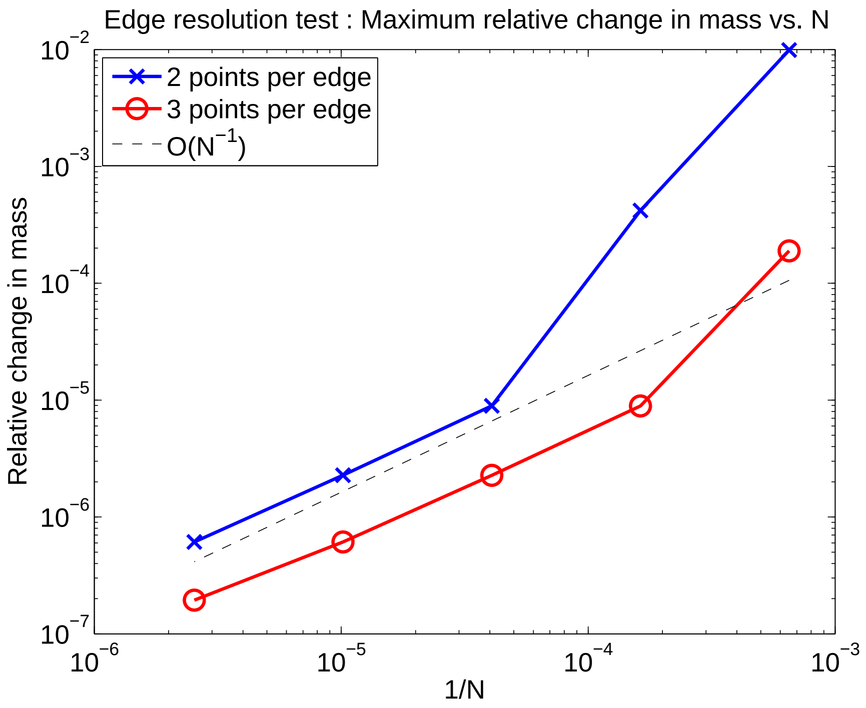


Figure 5.5: Maximum relative change in total mass (5.6) for a standard set of triangular panels (blue) and a set with an additional passive particle placed at edge midpoints (red). No remeshing, $\Delta t = 0.0125$.

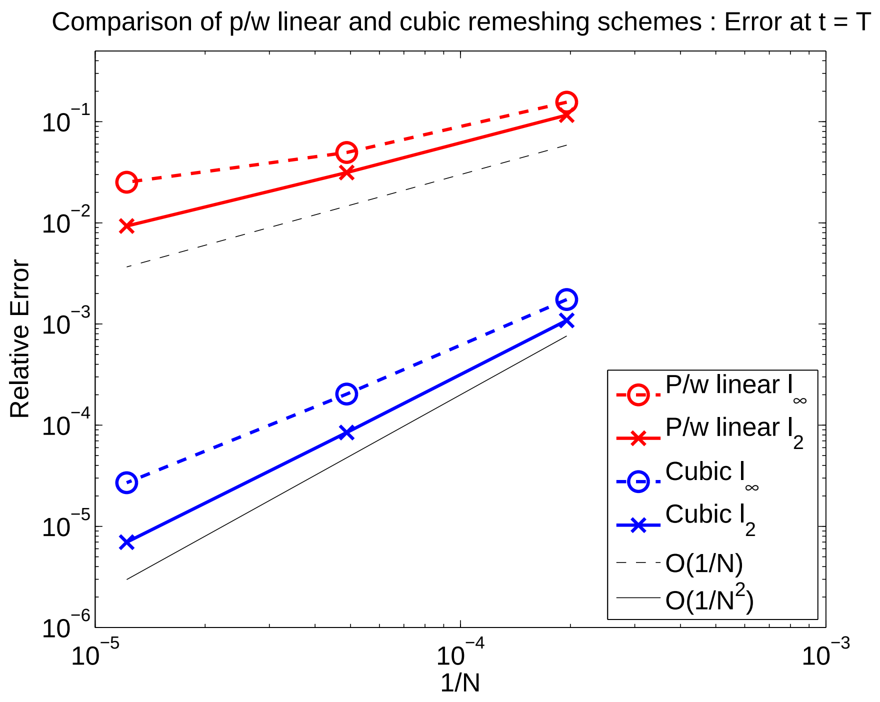


Figure 5.6: Comparison of Lagrangian remeshing with piecewise linear vs. cubic interpolation at $t = T$ for the non-divergent wind field and Gaussian hills tracer.

Lagrangian remeshing with the cubic interpolation method is more accurate and converges at a faster rate than the same remeshing procedure with piecewise linear interpolation.

Relative error measures l_2 and l_∞ for the Gaussian hills tracer field advected by (5.3) as $\Delta\lambda \to 0$ are shown in figure 5.7. These computations used $\Delta t = 0.0125$ and Lagrangian remeshing with cubic interpolation at intervals of 20 time steps for all N. Data from quadrilateral panel computations are shown in blue, triangular panels in red. For reference, two lines with slope $\Delta\lambda^4$ are plotted as thin black lines. Both types of panels show similar rates of convergence in both l_2 and l_∞ norms. The computed rate of convergence $\mathcal{K}_2$ of l_2 error and the rate of convergence $\mathcal{K}_\infty$ of l_∞ error [105, equation 26] are given in table 5.1, along with the same values for the Lin-Rood advection scheme for comparison. Figure 5.6 and table 5.1 document a significant improvement in error convergence rates due to higher-order interpolation during Lagrangian remeshing.

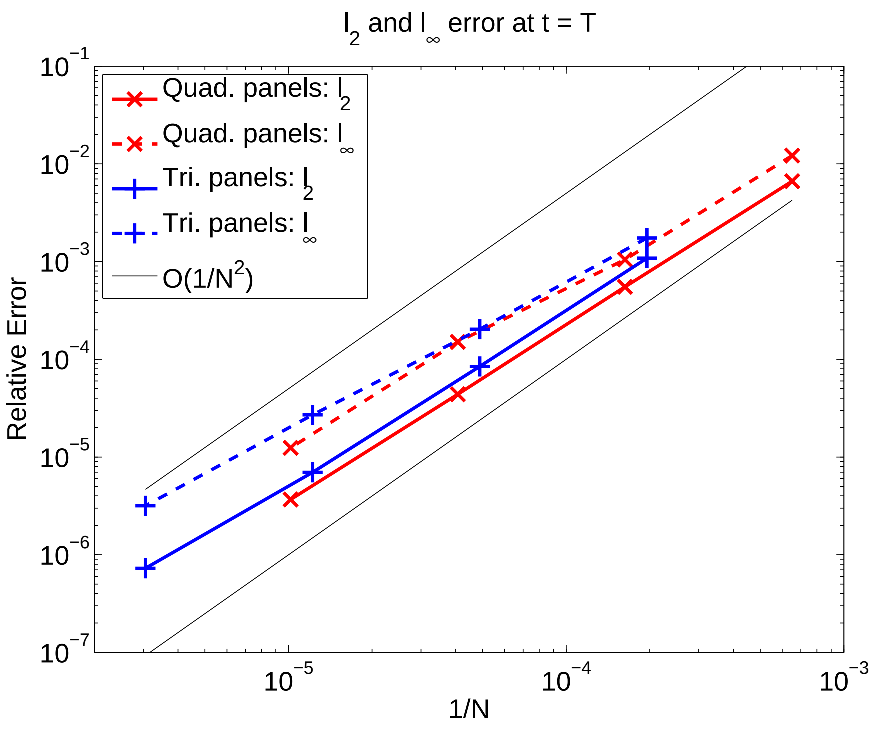


Figure 5.7: Advection of Gaussian hills tracer. Relative error measures l_2 and l_∞ for quadrilateral panels (red) and triangular panels (blue) as mesh size $\Delta\lambda$ decreases.

We also compare direct remeshing using SSRFPACK to Lagrangian remeshing with SSRFPACK for the Gaussian hills tracer advected by the non-divergent wind. This comparison was also performed for the BVE in the Rossby-Haurwitz wave test in figure 4.8. Results from the advection test case are presented in figure 5.8. In the Rossby-Haurwitz wave test, Lagrangian remeshing had a faster convergence rate than direct remeshing, but also had higher error for coarse panel sets (small N). In contrast, for this advection test case, both remeshing procedures have the same rate of convergence, but Lagrangian remeshing has better accuracy by nearly two orders of magnitude in both l_2 and l_∞ error measures. We attribute this difference to the fact that remeshing error in vorticity during a BVE computation feeds back into the velocity computations via the Biot-Savart integral. This feedback will affect error calculations especially in a test case designed around the stationary phase speed of the vorticity pattern like the Rossby-Haurwitz wave test from

	$\mathcal{K}_2$	$\mathcal{K}_\infty$
L-remesh, linear interp.	1.8566	1.2938
L-remesh, cubic interp.	3.5238	3.0225
Lin-Rood	1.9	1.3

Table 5.1: Convergence rates of l_2 error, $\mathcal{K}_2$, and l_∞ error, $\mathcal{K}_\infty$, as mesh size $\Delta\lambda \to 0$, defined by [105].

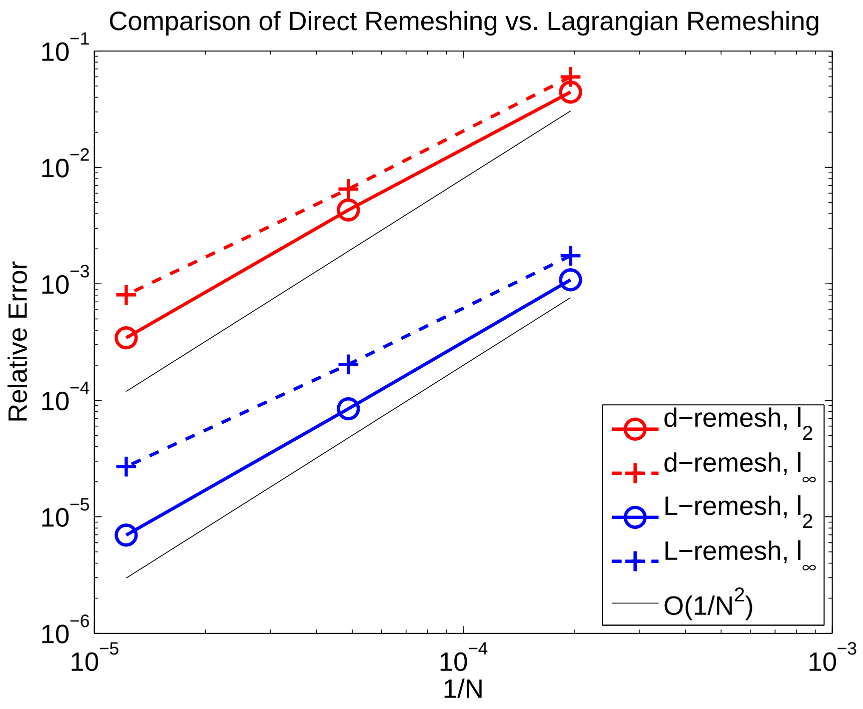


Figure 5.8: l_2 and l_∞ relative error for the Gaussian hills tracer advected by the non-divergent wind (5.3). All computations used $\Delta t = 0.0125$ and were remeshed every 20 time steps. Direct remeshing results are shown in red; Lagrangian remeshing in blue. Black lines have slopes of $1/N^2$.

section 4.2. In the advection test cases, velocity is simply prescribed from a function (5.3) regardless of tracer values, so the effects of remeshing error are confined to the tracer field.

Both direct and Lagrangian remeshing curves in table 5.1 and figure 5.8 show convergence rates higher than the $O(1/N) \sim O\left(\Delta\lambda^2\right)$ rate expected from midpoint rule quadrature. The high rate of convergence matches the degree of the interpolation scheme, not the quadrature scheme. This indicates that errors due to remeshing are more significant than discretization error due to quadrature and justifies the use of a high-order interpolation method.

The different effects of direct remeshing and Lagrangian remeshing are most visible in the discontinuous slotted cylinder test case, which again uses the non-divergent velocity field (5.3). Many advection schemes apply filters or limiters to preserve positivity or monotonicity of advected tracer values, and this test is designed to challenge schemes with a shape that is difficult to preserve due to the jump discontinuity at the boundary of each cylinder [105]. The goal for schemes in this test case is to produce monotone (non-oscillatory) results at $t = T$. We also use this test case as a way to expose smoothing introduced by the numerical advection scheme, which results in spurious mixing of the tracer.

In figure 5.9 we show results associated with test case 5 from [105], which advects the slotted cylinders tracer field with velocity field (5.3). The top row shows the results from applying direct remeshing, where ϕ was interpolated from a deformed set of panels to a new set of panels using SSRFPACK at each remeshing step. The bottom row shows the results from interpolating $\boldsymbol{\alpha}$ instead, using the Lagrangian remeshing procedure. Both calculations used $N = 20480$ triangular panels ($\Delta\lambda \approx 2^\circ$), $\Delta t = 0.0125$, and were remeshed every 20th time step.

It is evident that the effects of the interpolation scheme cause errors in the tracer field in the direct remeshing case. Overshoots and oscillations are apparent in both the $t = T/2$ and $t = T$ plots. Of course, interpolation errors are still present in the Lagrangian remeshing procedure, but they reside in the components of $\boldsymbol{\alpha}$, not ϕ. Each interpolated Lagrangian parameter $\boldsymbol{\alpha}_j$ will have some interpolation error in each of its radial, latitudinal, and longitudinal components. Renormalizing $\boldsymbol{\alpha}_j$ to $\left|\boldsymbol{\alpha}_j\right| = 1$ after the interpolation removes the radial error. The latitudinal and longitudinal interpolation errors that are still present now reside in the flow map, rather than the advected tracer. This means that some particles will be slightly out of position relative to their Lagrangian parameter, but the gains in tracer accuracy shown in figures 5.8 and 5.9 show that this effect is minimal. Visually, the slotted

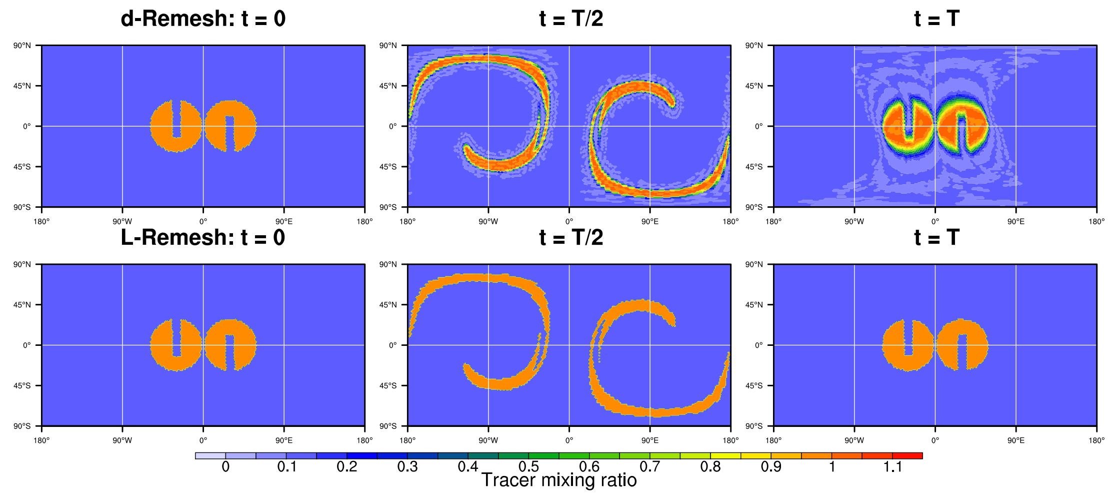


Figure 5.9: Advection of slotted cylinders by velocity field (5.3), $N = 20480$ triangular panels, $\Delta t = 0.0125$, $R_I = 20$. Direct remeshing (top row) causes overshoots and oscillations in the tracer field. Lagrangian remeshing (bottom row) introduces no oscillations and preserves the exact tracer range.

cylinders at $t = T$ look identical to their original distribution when Lagrangian remeshing is used.

As a final comparison for the slotted cylinders test case, we compare Lin-Rood and LPPM in figure 5.10. Despite its success as a dynamical core in CAM-FV and at solving the BVE, the Lin-Rood advection scheme struggles with the non-smooth slotted cylinders.

The cosine bells tracer field provide several data points for comparing advection schemes in [105]. The first is a measure of how well filaments are preserved. Filaments were a main focus of our work with the barotropic vorticity equation, and guided our selection of adaptive refinement criteria. In these test cases we use uniform (non-adaptive) panel sets and measure the preservation of tracer filaments at $t = T/2$, when the velocity field has deformed the tracer by the greatest amount. Filament preservation is measured with the function $A_F(\tau, t)$,

$$A_F(\tau, t) = \sum_{j \in G} A_j, \tag{5.7a}$$

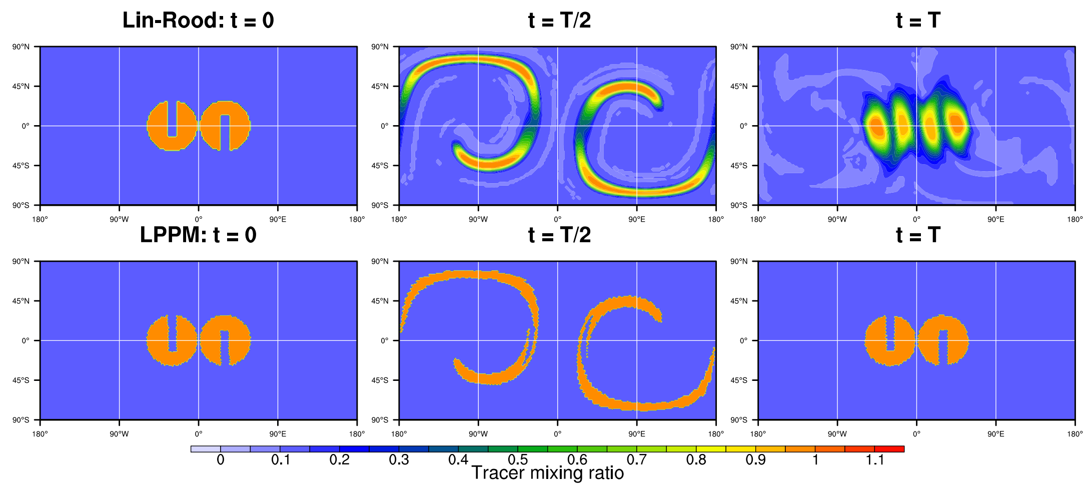


Figure 5.10: Advection of the slotted cylinders tracer by (5.3). Lin-Rood advection scheme from section 3.6 at 2° × 2° resolution, top row; Lagrangian Particle-Panel Method (LPPM) with $N = 20480$ triangular panels, bottom row.

where G is the set of panel indices whose active particles have tracer values that exceed τ,

$$G = \{j \in [1, N] : \phi_j > \tau\}. \tag{5.7b}$$

Thus the filament measure A_F represents the area of the sphere at time t with tracer values larger than τ. The filament error norm is defined as

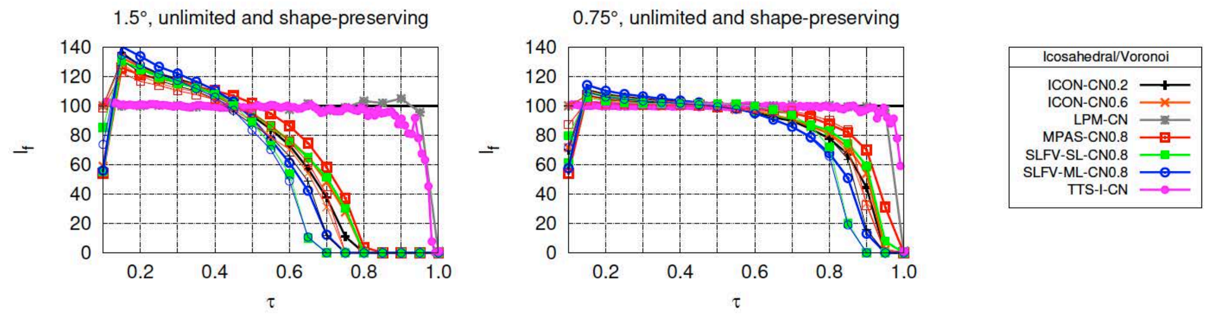
$$l_F(\tau, t) = \begin{cases} 100 \times \frac{A_F(\tau,t)}{A_F(\tau,0)} & \text{if } A_F(\tau, 0) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

A perfect scheme in this norm would exactly match the function

$$l_F^E(\tau, t) = \begin{cases} 100 & 0 \leq \tau < 1 \\ 0 & \tau = 1 \end{cases}. \tag{5.9}$$

at any time t. Diffusive schemes will have larger l_F for small τ due to smearing and smaller l_F for large τ due to dissipation of the tracer's maxima.

Figure 5.11 shows results from some of the participating models in [106]. Each model is plotted as a separate color and its corresponding acronym is shown in the legend. LPPM



(a) (b)

Figure 5.11: Filament diagnostic l_f at $t = T/2$. Each color corresponds to a different advection scheme from [106]. LPPM is listed as LPM-CN in this figure, and plotted as the gray curve. LPPM computations used triangular panels with (a) $N = 20480$ and (b) $N = 81920$.

is listed as LPM-CN in this plot according to the naming convention determined by the transport workshop, and plotted as the gray curve. The results from LPPM in this figure used triangular panels, $N = 20480$ (a) and $N = 81920$ (b), $\Delta t = 0.0125$ and Lagrangian remeshing using piecewise linear interpolation at intervals of twenty time steps. In this error measure there is not a visible difference between Lagrangian remeshing based on piecewise linear interpolation and cubic interpolation. Of the models shown in this figure, only LPPM and TTS-I-CN (pink curve) perform well at this diagnostic. Both are Lagrangian advection schemes, as TTS stands for trajectory-tracking scheme, and both are very close to the exact function (5.9). Details of each scheme are provided in [106].

The last important measure of LPPM as an advection scheme we mention in this chapter is its ability to preserve functional relations between correlated tracers. The ability to preserve relationships between correlated tracers is vital to chemistry models attached to a dynamical core. In an ideal dynamical core with exact tracer transport, only the chemistry parameterizations would alter the concentrations of chemical species. Errors in tracer values from the advection scheme can have adverse effects on chemistry parameterizations by introducing nonphysical states or unrealistic mixing [116, 105]. One of the main design goals of the Lin-Rood scheme was to provide a transport algorithm that preserves linear correlations between different tracers [116]. As models have become more complex, it is now desirable to preserve nonlinear correlations as well. A key feature of the LPPM advection scheme with Lagrangian remeshing is that, through the definition of the Lagrangian parameter and the flow map, it preserves linear and nonlinear correlations between tracers exactly. Preserving nonlinear correlations between tracers is a significant benefit of using LPPM as an advection scheme; comparisons with other advection schemes' performance on a correlated tracer test problem are presented in [106].

CHAPTER VI

Shallow Water Equations

The shallow water equations provide the next equation set for proving dynamical cores [187]. They include all the effects of vorticity that the barotropic vorticity equation captures, and by including effects of divergence, they also capture gravity waves. Several existing operational models, e.g. CAM-FV [115], use the shallow water equations in a 3D environment; the atmosphere is represented as a collection of vertical levels, and the shallow water equations are solved on each level. Adjacent levels are coupled with methods determined by the vertical discretization. Numerical models that do well in the shallow water environment are potentially well-suited for further development into a fully three-dimensional dynamical core that solves the primitive equations, with or without the hydrostatic approximation.

Derivations of the shallow water equations in a rotating frame are provided by [12] and [183]. Numerous different forms of the shallow water equations are given in [187] and [15]. Augenbaum [6] and Mesinger [130] provide Lagrangian representations of the spherical shallow water equations based on their advective form, with the two components of velocity and the fluid depth as prognostic variables. The Lagrangian form we propose here uses vorticity, divergence, and fluid depth as the prognostic variables, and we will again use Green's function and the Poisson problem to calculate fluid velocity.

The advective form of the shallow water equations has a momentum equation and a continuity equation,

$$\frac{d\boldsymbol{u}}{dt} = -f\hat{\boldsymbol{n}}\times\boldsymbol{u} - g\nabla h \tag{6.1a}$$

$$\frac{dh}{dt} = -h(\nabla\cdot\boldsymbol{u}), \tag{6.1b}$$

where the velocity, $\boldsymbol{u}$, and the Coriolis parameter, f, are the same as in our previous work with the barotropic vorticity equation. The fluid thickness, $h = h_S - h_B$, is the difference between the height of the fluid's free surface h_S and the height of the bottom topography, h_B. The constant g represents acceleration due to gravity.

We again use Green's function as the basis of our velocity calculation. The shallow water equations introduce an additional Poisson problem that results from the Helmholz-Hodge decomposition of the velocity [30],

$$\boldsymbol{u} = \nabla\psi\times\hat{\boldsymbol{n}} + \nabla\Phi \tag{6.2}$$

where ψ is the stream function, related to vorticity by equation (3.13). The velocity potential function Φ is defined as the solution of another Poisson problem, this one in terms of the divergence. Let δ denote the divergence of $\boldsymbol{u}$, then Φ satisfies

$$\delta = \nabla\cdot\boldsymbol{u} \tag{6.3a}$$

$$\nabla^2\Phi = \delta. \tag{6.3b}$$

We have used the symbol δ before to denote the Dirac delta function, but rely on the context to differentiate the two disparate uses of the same symbol.

We need to change equations (6.1) into a set of equations in terms of the variables ζ and δ so that we may use the Biot-Savart integral to solve them. For simplicity we assume planar flow in Cartesian geometry for the following derivation. Let $\boldsymbol{u} = [u, v, 0]^T$ be the velocity in the plane; the unit normal vector is $\hat{\boldsymbol{n}} = \hat{\boldsymbol{k}} = [0, 0, 1]^T$. Note that $\boldsymbol{u}\cdot\hat{\boldsymbol{n}} = \boldsymbol{u}\cdot\hat{\boldsymbol{k}} = 0$.

6.1 Shallow water vorticity equation

Starting with the momentum equation (6.1a) we apply a vector identity

$$\begin{aligned}(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} &= \nabla\left(\frac{1}{2}|\boldsymbol{u}|^2\right) - \boldsymbol{u}\times(\nabla\times\boldsymbol{u})\\ &= \nabla\left(\frac{1}{2}|\boldsymbol{u}|^2\right) - \boldsymbol{u}\times\boldsymbol{\zeta} \qquad (6.4)\end{aligned}$$

where $\boldsymbol{\zeta}$ is the relative vorticity, $\boldsymbol{\zeta} = \nabla\times\boldsymbol{u}$. Expanding the material derivative and replacing the advective derivative with (6.4), we have

$$\frac{\partial \boldsymbol{u}}{\partial t} - \boldsymbol{u}\times\boldsymbol{\zeta} = -f\hat{\boldsymbol{k}}\times\boldsymbol{u} - \nabla\left(gh + \frac{1}{2}|\boldsymbol{u}|^2\right). \qquad (6.5)$$

Next we apply the curl operator to (6.5), and the far right term vanishes,

$$\frac{\partial \boldsymbol{\zeta}}{\partial t} - \nabla\times(\boldsymbol{u}\times\boldsymbol{\zeta}) = -\nabla\times(f\hat{\boldsymbol{k}}\times\boldsymbol{u}). \qquad (6.6)$$

Two applications of a vector identity give

$$\begin{aligned}\nabla\times(\boldsymbol{u}\times\boldsymbol{\zeta}) &= \cancel{\boldsymbol{u}(\nabla\cdot\boldsymbol{\zeta})} - \boldsymbol{\zeta}(\nabla\cdot\boldsymbol{u}) + \cancel{(\boldsymbol{\zeta}\cdot\nabla)\boldsymbol{u}} - (\boldsymbol{u}\cdot\nabla)\boldsymbol{\zeta}\\ &= -\boldsymbol{\zeta}(\nabla\cdot\boldsymbol{u}) - (\boldsymbol{u}\cdot\nabla)\boldsymbol{\zeta} \qquad (6.7a)\\ \nabla\times(f\hat{\boldsymbol{k}}\times\boldsymbol{u}) &= f\hat{\boldsymbol{k}}(\nabla\cdot\boldsymbol{u}) - \cancel{\boldsymbol{u}(\nabla\cdot f\hat{\boldsymbol{k}})} + (\boldsymbol{u}\cdot\nabla)f\hat{\boldsymbol{k}} - \cancel{(f\hat{\boldsymbol{k}}\cdot\nabla)\boldsymbol{u}}\\ &= f\hat{\boldsymbol{k}}(\nabla\cdot\boldsymbol{u}) + (\boldsymbol{u}\cdot\nabla)f\hat{\boldsymbol{k}}. \qquad (6.7b)\end{aligned}$$

The cancellations in the above equations are due to orthogonality and to the fact that the divergence of a curl is zero. Using (6.7) in (6.6) yields the vorticity equation for rotating shallow water flow,

$$\frac{\partial \boldsymbol{\zeta}}{\partial t} + (\boldsymbol{u}\cdot\nabla)(\boldsymbol{\zeta} + f\hat{\boldsymbol{k}}) = -(\boldsymbol{\zeta} + f\hat{\boldsymbol{k}})(\nabla\cdot\boldsymbol{u}). \qquad (6.8)$$

Applying the dot product with the vector $\hat{\boldsymbol{k}}$ to both sides of (6.8) gives the scalar vorticity equation

$$\frac{\partial \zeta}{\partial t} + (\boldsymbol{u}\cdot\nabla)(\zeta + f) = -(\zeta + f)\delta. \qquad (6.9)$$

By combining the continuity equation (6.1b) and the vorticity equation (6.9) we find the potential vorticity $q = (\zeta + f)/h$, which is a material invariant,

$$\frac{dq}{dt} = \frac{d}{dt}\left(\frac{\zeta + f}{h}\right) = 0. \tag{6.10}$$

Potential vorticity in the shallow water equations is similar to absolute vorticity in the barotropic vorticity equation, with one important addition. Like the BVE's absolute vorticity, meridional motion in the SWE changes a particle's relative vorticity due to changes in the Coriolis parameter, f. SWE potential vorticity is also affected by changes in fluid thickness, h. If a particle has decreasing fluid thickness and does not change latitude, then its relative vorticity also decreases, in order to conserve q. Similarly, if a particle is stretched vertically and h increases along a constant latitude circle, then its relative vorticity also increases. A helpful image to guide this interpretation is that of a spinning figure skater on ice. As the skater retracts her arms and stretches them above her head, her rate of spin (vorticity) increases due to conservation of angular momentum; the same concept acts in the shallow water equations through the effects of changing h on potential vorticity q.

6.2 Shallow water divergence equation

To derive the divergence equation we apply the divergence operator to equation (6.5).

$$\frac{\partial \delta}{\partial t} - \nabla \cdot (\boldsymbol{u} \times \boldsymbol{\zeta}) = -\nabla \cdot (f\hat{\boldsymbol{k}} \times \boldsymbol{u}) - \nabla^2\left(gh + \frac{1}{2}|\boldsymbol{u}|^2\right). \tag{6.11}$$

We apply two vector identities to the $\nabla \cdot (\boldsymbol{u} \times \boldsymbol{\zeta})$ term,

$$\nabla \cdot (\boldsymbol{u} \times \boldsymbol{\zeta}) = \boldsymbol{\zeta} \cdot (\nabla \times \boldsymbol{u}) - \boldsymbol{u} \cdot (\nabla \times \boldsymbol{\zeta}) \tag{6.12a}$$

$$\boldsymbol{u} \cdot (\nabla \times \boldsymbol{\zeta}) = \boldsymbol{u} \cdot (\nabla \times \nabla \times \boldsymbol{u}) = \boldsymbol{u} \cdot \left(\nabla\delta - \nabla^2\boldsymbol{u}\right) \tag{6.12b}$$

$$\implies \nabla \cdot (\boldsymbol{u} \times \boldsymbol{\zeta}) = |\zeta|^2 - (\boldsymbol{u} \cdot \nabla)\delta + \boldsymbol{u} \cdot \nabla^2\boldsymbol{u}, \tag{6.12c}$$

where the vector Laplacian $\nabla^2 \boldsymbol{u}$ is defined as the Laplacian operator applied to each component of $\boldsymbol{u}$, so that $\nabla^2 \boldsymbol{u} = [\nabla^2 u, \nabla^2 v, 0]^T$. It is helpful to expand some terms of (6.11) and (6.12c) into their components:

$$|\zeta|^2 = \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right)^2 \tag{6.13a}$$

$$\nabla^2 \frac{1}{2} |\boldsymbol{u}|^2 = \left(\frac{\partial u}{\partial x}\right)^2 + u\frac{\partial^2 u}{\partial x^2} + \left(\frac{\partial u}{\partial y}\right)^2 + u\frac{\partial^2 u}{\partial y^2} + \left(\frac{\partial v}{\partial x}\right)^2 + v\frac{\partial^2 v}{\partial x^2} + \left(\frac{\partial v}{\partial y}\right)^2 + v\frac{\partial^2 v}{\partial y^2} \tag{6.13b}$$

$$\boldsymbol{u} \cdot \nabla^2 \boldsymbol{u} = u\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + v\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right). \tag{6.13c}$$

These terms combine and cancel in equation (6.11) to give the shallow water divergence equation

$$\frac{\partial \delta}{\partial t} + (\boldsymbol{u} \cdot \nabla)\delta = 2J(\boldsymbol{u}) - \delta^2 - 2\boldsymbol{u} \cdot \nabla^2 \boldsymbol{u} - (\boldsymbol{u} \cdot \nabla) f \hat{\boldsymbol{k}} + f\hat{\boldsymbol{k}} \cdot \boldsymbol{\zeta} - \nabla^2 gh, \tag{6.14}$$

where $J(\boldsymbol{u})$ is the Jacobian determinant of $\boldsymbol{u}$,

$$J(\boldsymbol{u}) = \frac{\partial u}{\partial x}\frac{\partial v}{\partial y} - \frac{\partial u}{\partial y}\frac{\partial v}{\partial x}. \tag{6.15}$$

In equation (6.14) we see that the divergence of fluid particles is affected by advection of both relative and planetary vorticity, which should be expected given that divergence also affects particles' relative vorticity via equation (6.9). The curvature of the height field also affects fluid particles' divergence via the term $\nabla^2 gh$, which is analogous to the divergence of the pressure gradient in the Euler equations. The Jacobian $J(\boldsymbol{u})$ and the vector Laplacian $\boldsymbol{u} \cdot \nabla^2 \boldsymbol{u}$ measure the stretching and strain encountered by a fluid particle due to the fluid velocity, which also affect divergence.

6.3 Lagrangian form

To represent the shallow water equations on the sphere, we use the above derivation which is written in vector form and independent of a coordinate system. The unit normal vector on the sphere $\hat{\boldsymbol{n}}$ is orthogonal to the tangential velocity and parallel to the vorticity vector (section 1.5), so $\boldsymbol{u} \cdot \boldsymbol{n} = 0$ as in the planar case.

Since $\frac{\partial f}{\partial t} = 0$, the material derivative of the Coriolis parameter f has only the advective derivative,

$$\frac{df}{dt} = (\boldsymbol{u} \cdot \nabla) f = 2\Omega \frac{dz}{dt}. \tag{6.16}$$

This allows us to simplify equations (6.9) and (6.14) and write the Lagrangian form of the rotating shallow water equations in terms of vorticity and divergence,

$$\frac{d\zeta}{dt} + \zeta\delta = -2\Omega\frac{dz}{dt} - f\delta \tag{6.17a}$$

$$\frac{d\delta}{dt} + \delta^2 = -2\Omega\frac{dz}{dt} + f\zeta + 2J(\boldsymbol{u}) - 2\boldsymbol{u}\cdot\nabla^2\boldsymbol{u} - \nabla^2 gh \tag{6.17b}$$

$$\frac{dh}{dt} + h\delta = 0 \tag{6.17c}$$

$$\boldsymbol{u}(\boldsymbol{x}) = -\frac{1}{4\pi}\int_S \frac{\boldsymbol{x}\times\tilde{\boldsymbol{x}}}{1-\boldsymbol{x}\cdot\tilde{\boldsymbol{x}}}\zeta(\tilde{\boldsymbol{x}})d\tilde{A} + \frac{1}{4\pi}\int_S \frac{\boldsymbol{x}\times\tilde{\boldsymbol{x}}}{1-\boldsymbol{x}\cdot\tilde{\boldsymbol{x}}}\delta(\tilde{\boldsymbol{x}})d\tilde{A} \tag{6.17d}$$

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{u}(\boldsymbol{x}). \tag{6.17e}$$

In the above equations, the prognostic variables are ζ, δ, h and $\boldsymbol{x}$. Equations (6.17a) – (6.17c) are the shallow water equations; equation (6.17d) is the Biot-Savart law and (6.17e) is the flow map.

Note that if h is constant and δ is zero in (6.17), the barotropic vorticity equation (3.25) is recovered. Apart from the forcing terms on the right hand side of (6.17b), the shallow water system looks like they could be solved with a straightforward extension of the methods developed and presented in chapters III and IV for the barotropic vorticity equation. Unfortunately, the forcing terms in (6.17b) involve spatial derivatives of the velocity, which are challenging to compute on nonuniform, possibly distorted meshes.

Applying the spatial derivative operators in (6.17b) to the integrals in (6.17d) increases the singularities in the integral kernels. The divergence of the Biot-Savart vector kernel is a delta function; its vector Laplacian is another derivative that must be understood in the sense of distributions [190], and cannot be computed directly.

We have already seen how topography becomes a forcing term in the stratospheric model of breaking Rossby waves in section 4.5. The forcing terms in (6.17b) are more complex, but are the main obstacle between algorithm II.1 and a shallow water equation solver.

6.4 Proposed numerical methods

Our existing algorithms are currently capable of solving the shallow water equations assuming we can evaluate the spatial derivatives of velocity, $J(\boldsymbol{u})$ and $\nabla^2\boldsymbol{u}$, and height $\nabla^2 gh$ on our particles and panels. These terms are challenging to evaluate in the Lagrangian frame; finite difference stencils are difficult to implement on moving, unstructured meshes. Several techniques have been introduced that we may apply to our methods.

Strategy 1: Separate grids. The first strategy we may use would maintain two separate grids. One is our set of particles and panels, and the other would be a latitude-longitude grid where finite differences are easily computed. In this approach, at each time increment the output from the Biot-Savart integral would be mapped to the latitude-longitude grid by an interpolation scheme. The forcing terms $J(\boldsymbol{u})$ and $\boldsymbol{u} \cdot \nabla^2\boldsymbol{u}$ would be evaluated using finite differences on the latitude-longitude grid and then mapped back to the Lagrangian particles and panels. This method is closely related to the Hamiltonian Particle Mesh method introduced in [60, 61] and similar to the shallow water equation solvers by Dritschel [49, 45, 163].

In Dritschel's methods, vorticity is maintained on a set of Lagrangian contours whose Lagrangian parameter α are the materially invariant potential vorticity q. Velocity is computed on an Eulerian mesh using a combination of spectral methods and finite differences and used to advect the Lagrangian contours, whose vorticity is mapped to the Eulerian grid and used in the next velocity computation. This approach can be interpreted as keeping the vorticity defined on one grid, the Lagrangian contours, and the divergence on the Eulerian grid. Since it is the divergence of the velocity field that is responsible for gravity waves, these methods have been described as a wave-vortex decomposition of the shallow water equations [45].

This strategy is a step away from the flow map which we have sought to maintain throughout the entirety of this thesis. The need for remapping the forcing terms to and from the fixed mesh at each time step is heavily reliant on interpolation, which we would like to avoid outside of the isolated remeshing steps.

Strategy 2: Local derivatives on Lagrangian panels. In their Lagrangian methods for solving the shallow water equations on the sphere, Augenbaum [6] and Mesinger [130] evaluated the necessary spatial derivatives in local neighborhoods centered at each fluid particle. Their discretizations were based on the advective form of the shallow water equations (6.1) and therefore did not require the same derivatives we require to solve the shallow water equations' vorticity-divergence form. However, their techniques are general enough to provide a method that we could use.

Mesinger used polynomial interpolation around each particle. Nearby particles were used to provide a least-squares fit to a quadratic interpolating polynomial centered at the location of the particle whose velocity was currently being computed. Assuming it is of high enough degree to have the correct number of derivatives, the polynomial can be differentiated to approximate the required forcing terms. As we would need to approximate the Laplacian, we would need at least a quadratic interpolating polynomial about each particle. We have already developed methods for building piecewise linear interpolating polynomials on each panel, using the point-location query algorithm and barycentric coordinates (section A.2). Extending this method to a higher-order approximation would be fairly straightforward, and similar to the compressible flow particle method introduced by [139].

Augenbaum's approach computed finite difference stencils on a Lagrangian Voronoi mesh. He was unable to produce high-order accurate spatial derivatives [6], but recent advances from other dynamical cores that use Voronoi grids may provide more accurate methods of computing derivatives. Thuburn *et al.* [178] provide a method for computing higher order derivatives of velocity to get vorticity and divergence on arbitrarily structured staggered grids for geophysical applications. This method has become the foundation of the Model for Prediction Across Scales (MPAS) dynamical core which uses variable resolution spherical centroidal Voronoi tessellations as its horizontal discretization [151, 162]. It may be possible to use a similar technique to find $J(\boldsymbol{u})$ and $\nabla^2\boldsymbol{u}$ on our moving sets of particles and panels.

Strategy 3: Differentiating the regularized Biot-Savart kernel. Forcing terms can be computed by applying a regularization to the Biot-Savart kernel $\boldsymbol{K}$. The smooth kernel

$$\boldsymbol{K}_\epsilon(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{\boldsymbol{x} \times \tilde{\boldsymbol{x}}}{1 - \boldsymbol{x} \cdot \tilde{\boldsymbol{x}} + \epsilon^2} \tag{6.18}$$

can be differentiated to provide approximations of $J(\boldsymbol{u})$ and $\nabla^2\boldsymbol{u}$. As these integral kernels must be summed over the whole sphere, this idea can be thought of as a global derivative calculation. To begin, we would only use the smoothed kernel to evaluate the forcing terms of (6.17b); the singular kernel $\boldsymbol{K}(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ given by (3.22) would still be used to calculate fluid velocity in (6.17d).

The divergence of (6.18) yields an approximate delta function; it is investigated in the appendix, section A.1.

Another, similar, option would use high-order accurate interpolation using radial basis functions that could be differentiated to evaluate the forcing terms without having to resort to a separate grid. Radial basis functions have been used to solve the shallow water equations in an Eulerian context in [58]. To use radial basis functions for interpolation requires the solution of a dense, possibly ill-conditioned interpolating matrix, however, and this method may therefore be prohibitively expensive for large N [56]. Partition of unity methods [189, 27] may reduce the cost of using radial basis functions.

Strategy 4: Particle Strength Exchange (PSE). Particle strength exchange is a method that originated in viscous vortex methods, and was designed specifically to compute the Laplacian of the fluid velocity on sets of Lagrangian particles [42, 36, 38, 97]. It replaces the Laplacian derivative with an interpolating kernel that approximates the action of the Laplacian operator on testing functions in the sense of distributions. Particle Strength Exchange techniques have been extended to compute arbitrary spatial derivatives in vortex methods by [53] and adapted to the 2D compressible Euler equations in [54]. PSE provides global integrals that approximate derivatives similar to the smoothed Biot-Savart kernel, but PSE kernels can be high-order accurate [144].

CHAPTER VII

Conclusions and Ongoing Work

7.1 Summary

We have presented a Lagrangian Particle-Panel Method (LPPM) for solving geophysical fluid dynamics equations on the sphere for applications related to the dynamical cores of weather and climate models. Green's function is used to solve the underlying PDEs, in contrast to most current dynamical cores which rely on spectral transforms, finite element methods, and finite volume methods. Green's function and the Biot-Savart law provide solutions comparable to today's operational models even with a simple midpoint quadrature rule.

Our algorithms are based on vortex methods, and their computational elements are Lagrangian point vortices. Point vortices can resolve multiple spatial scales, but the Lagrangian formulation of the equations of motion that the point vortices follow are subject to problems of mesh distortion. While mesh distortion difficulties have prevented previous Lagrangian methods for spherical geophysical fluid dynamics solvers from achieving high accuracy or solutions at long times, we find that by using adaptive refinement and a new Lagrangian remeshing procedure we are able to overcome this challenge.

The Lagrangian remeshing procedure preserves the flow map and the Lagrangian features of the flow by interpolating the Lagrangian parameter rather than the problem data.

This approach is applicable to any equation set that can be represented in a Lagrangian frame via the flow map (2.1). Lagrangian remeshing is better than direct remeshing at conserving integral invariants such as kinetic energy and total enstrophy using the same interpolation methods. As an advection scheme, LPPM is a monotonic, positivity preserving tracer transport algorithm that introduces no overshoots or oscillations to advected tracer data. It preserves linear and nonlinear tracer correlations exactly.

Although the barotropic vorticity equation is too simple to provide practical forecasts, we have found that it is capable of modeling the stratosphere, and the dynamics of sudden stratospheric warmings in particular. Our model solutions have shown a tripole vortex emerge from the breakdown of a monopole vortex situated over the winter hemisphere's pole. Like [8], this result suggests that although tripole vortices have only rarely been noted in observational data, they may be a fundamental feature of geophysical fluids. More studies involving experimentation, modeling, and observational data analysis are needed to investigate this conjecture further.

7.2 Additional test cases

The test cases presented in chapter IV are designed to either provide insight into the numerical behavior of our techniques or to represent characteristics of geophysical fluid flow that can be captured in a barotropic model. In this section we present additional test problems that are aligned with these two goals.

7.2.1 Merging vortices

Kent *et al.* [95] use the planar barotropic vorticity equation to examine the effects of numerical dissipation on kinetic energy and enstrophy spectra of model solutions. They compare the effects of a variety of different numerical schemes and different coefficients for diffusion or hyper-diffusion operators within each scheme. Different models' solutions, vorticity error, kinetic energy, and total enstrophy are examined. These measures are the same as we used to evaluate our results in chapter IV.

Their first test case is a vortex merger problem where two vortices of the same sign are initialized close to each other; a large vortex stretches a smaller vortex and eventually the two vortices merge. This test could be implemented using our methods by initializing two Gaussian vortices of the same sign in close proximity to each other on the sphere. Modifying the equations from section 4.3, this test would have initial vorticity distributions given by

$$\zeta_0(\boldsymbol{x}) = \zeta^*(\boldsymbol{x}) - \bar{\zeta} \tag{7.1a}$$

$$\omega_0(\boldsymbol{x}) = \zeta^*(\boldsymbol{x}) - \bar{\zeta} + 2\Omega z \tag{7.1b}$$

where ζ^* is the sum of two Gaussian distributions,

$$\zeta^*(\boldsymbol{x}) = A_0 e^{-2\beta_0^2(1-\boldsymbol{x}\cdot\boldsymbol{x}_0)} + A_1 e^{-2\beta_1^2(1-\boldsymbol{x}\cdot\boldsymbol{x}_1)}. \tag{7.1c}$$

A_0 is the maximum value of the first vortex, whose shape parameter is β_0 and whose center is at $\boldsymbol{x}_0 \in \mathcal{S}$. The second vortex is defined by parameters A_1, β_1 and $\boldsymbol{x}_1 \in \mathcal{S}$ in a similar manner. The Gaussian constant used to ensure that total vorticity is zero is the same as in section 4.3,

$$\bar{\zeta} = \frac{1}{4\pi} \sum_{j=1}^{N} \zeta^*(\boldsymbol{x}_j(0)) A_j. \tag{7.1d}$$

The choice of maximum values for each vortex A_0 and A_1 should be scaled relative to Ω so that

$$\mathcal{O}(0.1) \lesssim \frac{\max_{\mathcal{S}} |\zeta_0|}{2\Omega} \lesssim \mathcal{O}(10). \tag{7.2}$$

Higher values of this ratio will provide flows dominated by local velocities, less affected by the background rotation of the sphere. This would model smaller scale vortices such

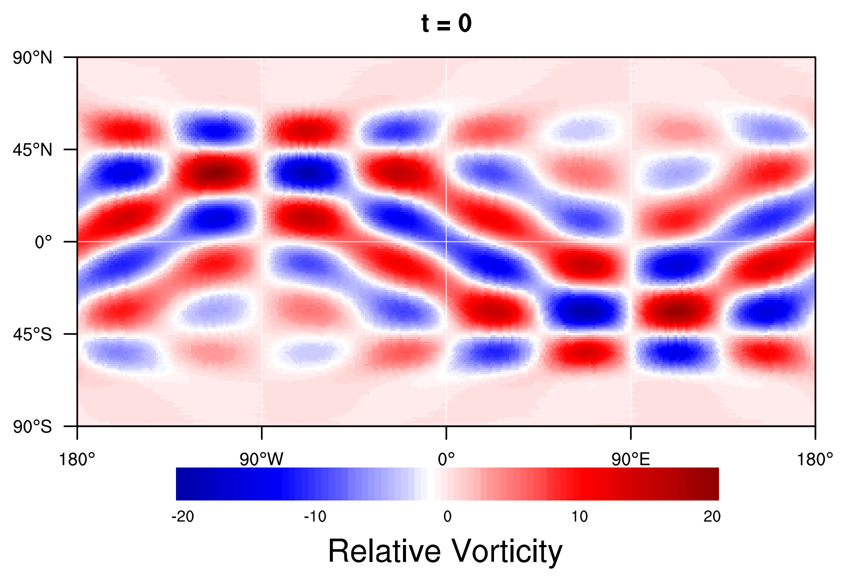


Figure 7.1: Initial relative vorticity for freely decaying turbulence test case.

as tropical cyclones. Consequently, higher values of A_0 and A_1 should be accompanied by larger values of β_0 and β_1 to produce smaller scale vortices. Flows with lower ratios (7.2) are more representative of planetary scale motion and will be more affected by the Coriolis force. The majority of test cases presented in chapter IV used ratios of $\max_{\mathcal{S}} |\zeta| / 2\Omega \sim \mathcal{O}(1)$ or less.

The most challenging test introduced by [95] is a freely decaying turbulence problem defined by an initial condition with many vortices. We adapt [95, test 4.3] by adding a shape parameter $e^{-\theta^8/2}$ to the to their initial vorticity distribution. This ensures that $|\zeta| \to 0$ as $\theta \to \pm\pi/2$ which maintains numerical stability on the Lin-Rood solver's latitude longitude grid. The initial vorticity distribution for this test case is given by

$$\zeta(\lambda,\theta) = \Big(4\pi \sin(4\lambda)\sin(8\theta) + 1.6\pi\cos(3\lambda)\cos(6\theta) + 1.2\pi\cos(5\lambda)\cos(10\theta) + 0.08\pi(\sin\lambda + \sin(2\theta))\Big)e^{-\theta^8/2}. \tag{7.3}$$

A plot of these initial conditions is shown in figure 7.1.

An interesting feature of this test case is that as t increases vortices merge and coalesce

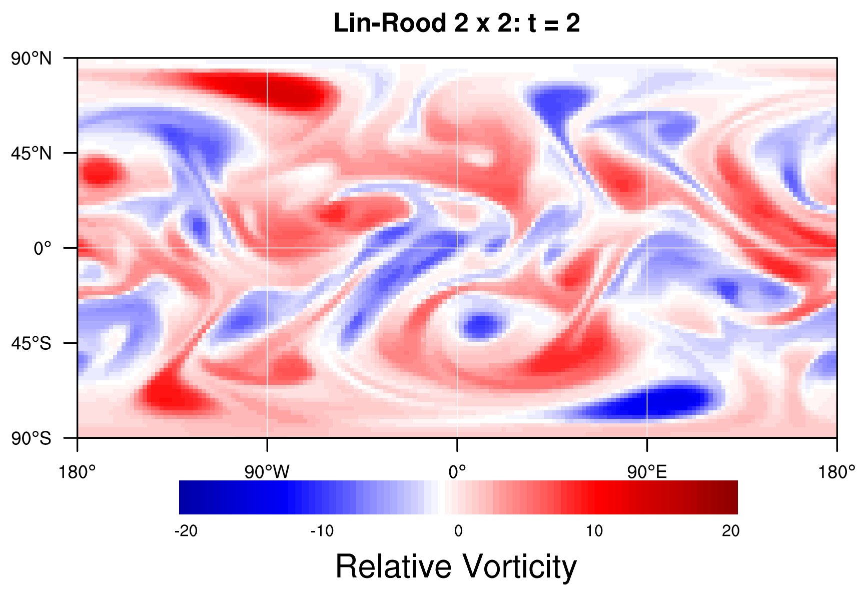


Figure 7.2: Relative vorticity for the freely decaying turbulence test case at $t = 2$. Lin-Rood solver, 2° resolution, $\Delta t = 0.0004$.

into a smaller number of larger scale vortices than existed at $t = 0$. The Lin-Rood BVE solver $t = 2$ solution using 2° resolution and $\Delta t = 0.0004$ is shown in figure 7.2.

The challenge for our method is that the initial conditions (7.3) cause immediate generation of small scales in the Lagrangian parameter α that quickly become too fine for a uniform set of panels to resolve. An example solution from a set of $N = 24576$ quadrilateral panels that used $\Delta t = 0.01$ and Lagrangian remeshing at intervals of 10 time steps is shown in figure 7.3. The checkerboard pattern of relative vorticity error along filaments between vortices will soon cause the computed solution to diverge from the correct solution.

Using adaptive Lagrangian remeshing restores accuracy but leads to a large increase in N as t increases. Results from a computation using $\epsilon_1 = 0.006$ and $\epsilon_2 = 0.15$ applied to a base mesh of $N = 6144$ quadrilateral panels are shown in figure 7.4. This computation used $\Delta t = 0.00125$ and adaptive Lagrangian remeshing every 20th time step. The figure shows both relative vorticity and the latitudinal component of the Lagrangian pa-

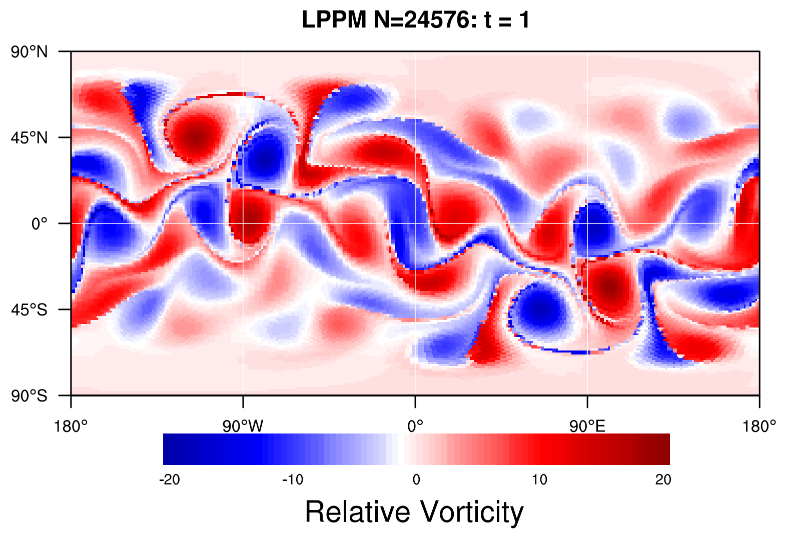
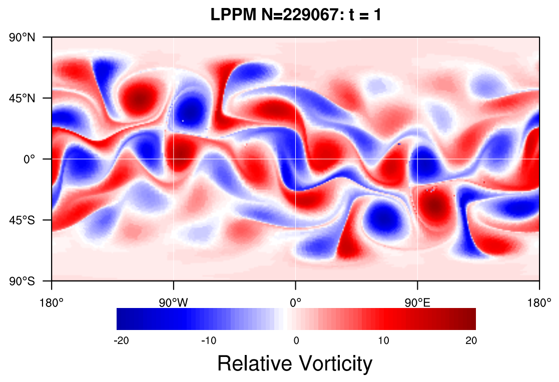


Figure 7.3: Relative vorticity at $t = 1$ computed by $N = 24576$ quadrilateral panels using $\Delta t = 0.01$ and Lagrangian remeshing at intervals of 10 time steps.

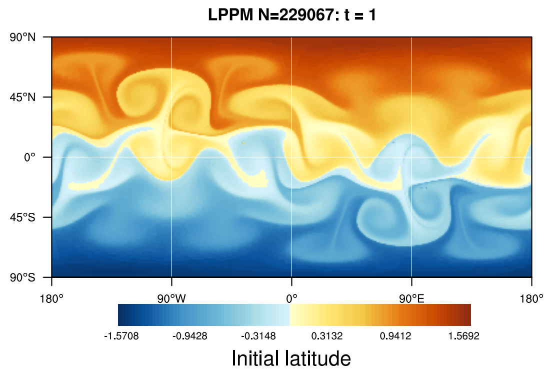
rameter, which shows many fine-scale filaments stretching between vortices. At $t = 0$ these tolerances return a mesh with $N = 22632$ panels, but by $t = 1$ they result in a mesh with $N = 229067$ panels. A plot of the mesh from this computation is shown in figure 7.5. Looking at the adaptive mesh, it is clear that a majority of the computational effort is being used to resolve the small scale filaments and the Lagrangian structures in the flow. Clearly such growth in N results in computations that become expensive at large t. Changing the Lagrangian remeshing procedure to interpolate to intermediate times instead of interpolating to $t = 0$ will begin to address this problem. We discuss this idea and several other strategies in various stages of development in section 7.3.

7.2.2 Modeling the upper troposphere

In section 4.5 we saw that our methods and the barotropic approximation of the stratosphere provide a realistic model of the dynamics associated with sudden stratospheric warmings. This suggests that our model, which can be thought of as a barotropic dynamical core, can be used to study other phenomena in the atmosphere which can be represented



(a)



(b)

Figure 7.4: Freely decaying turbulence test case computed on quadrilateral panels with adaptive Lagrangian remeshing. (a) Relative vorticity. (b) Particles' initial latitude, α_θ.

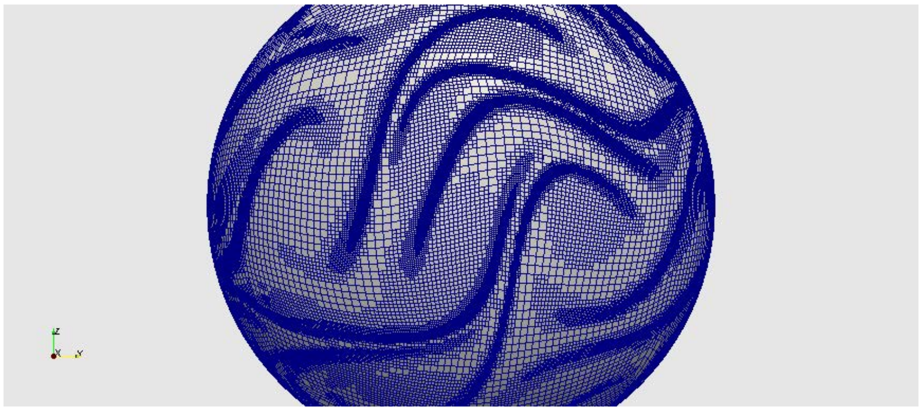


Figure 7.5: Adaptive mesh for computation shown in figure 7.4.

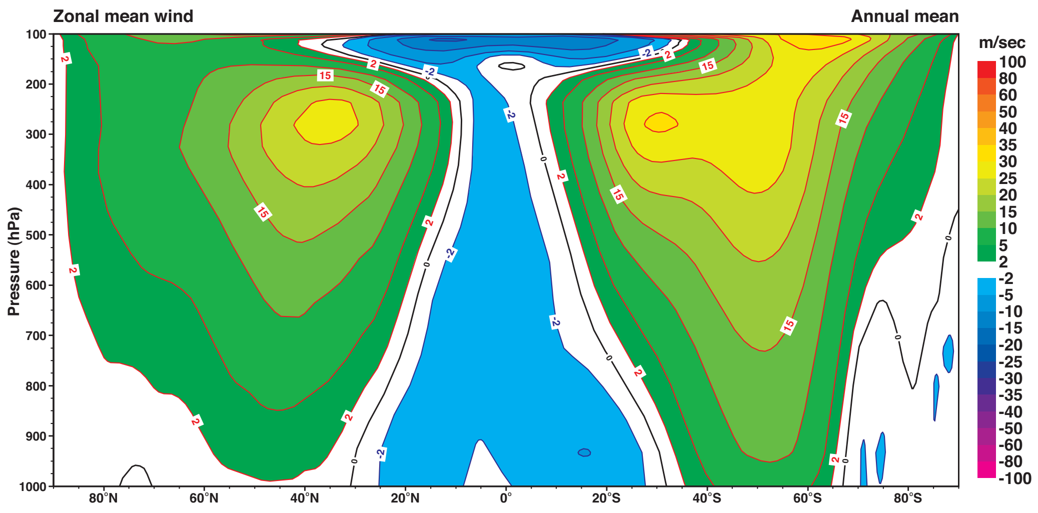


Figure 7.6: Annual-mean zonal-mean zonal wind profile from ECMWF ERA-40 reanalysis data. Vertical axis is pressure (hPa), horizontal axis is latitude, north to south, left to right.

with the barotropic vorticity equation. In this section, we introduce a technique for building general test cases that represent the upper troposphere (300 hPa). Users of the LPPM software may adapt this general technique to build test cases related to simulate specific events or flow features of their choosing.

The annual-mean zonal-mean zonal wind profile gives the meridionally- and time-averaged zonal wind u as a function of height. An example from European Center for Medium-range Weather Forecasting's (ECMWF) climate reanalysis data is shown in figure 7.6. These data are from the ERA-40 dataset, which includes data from the roughly 40 years spanning 1957-2002 [51]. If we take a cross section of the 300 hPa level of this plot, we see a mean zonal wind velocity profile that has an easterly maximum in the midlatitudes of each hemisphere, and a smaller magnitude westerly maximum near the equator. These features can be qualitatively approximated by Legendre polynomial $P_4^2(\sin\theta)$, figure 7.7. This function, scaled for velocities appropriate to the unit sphere, and its resulting

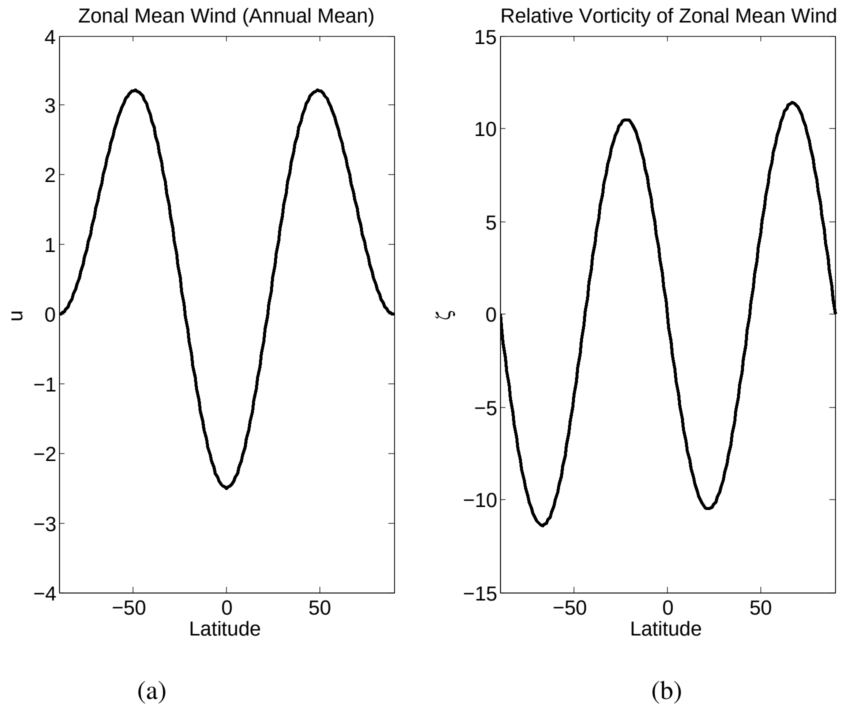


Figure 7.7: Model profiles of 300 mb annual-mean zonal-mean zonal wind (a) and its associated relative vorticity (b) from equation (7.4).

vorticity are

$$u(\lambda,\theta) = \frac{15}{6}\left(\cos^2\theta(7\sin^2\theta - 1)\right) \tag{7.4a}$$

$$\zeta(\lambda,\theta) = \frac{15}{24}\cos\theta(37\sin\theta - 35\sin(3\theta)). \tag{7.4b}$$

Like the steady zonal jet test problem from section 4.4, this purely zonal flow is an unstable steady solution of the barotropic vorticity equation that can be thought of as parallel shear flow on the sphere. Longitudinal perturbations can be added to (7.4) to simulate Rossby waves. Individual Gaussian vortices can be superposed on the mean zonal background wind to simulate interacting vortices in a realistic global circulation using the formulas from section 4.3. The vorticity in (7.4) satisfies $\int_S \zeta(\boldsymbol{x})\,dA = 0$ identically, so any perturbation vorticity ζ' added to (7.4) must satisfy $\int_S \zeta'(\boldsymbol{x})\,dA = 0$ to satisfy the Gaussian constraint on the Poisson problem. This can be done *a posteriori* using the $\bar{\zeta}$ formulas given in sections 4.3 and 7.2.1.

Other background profiles can be used as well. The ERA-40 dataset contains zonal mean wind profiles of each season, in addition to the annual profile. Also, any general initial condition may be approximated as a linear combination of spherical harmonics using spectral transform methods. So long as the fluid features can be reasonably characterized as incompressible and isothermal, all that is needed is an initial vorticity distribution to define the problem.

7.2.3 Tripole vortices

In section 4.5 we saw a tripole vortex emerge from a model of Rossby wave breaking near the stratospheric winter polar vortex. Many studies of stratospheric dynamics (i.e., [78, 93, 128, 124, 161]) use potential vorticity or temperature to measure and observe the polar vortex. A typical sudden stratospheric warming event has relative vorticity maxima that are only approximately 30% of the planetary vorticity [93], so it is possible that the emergence of tripole vortices have gone unnoticed against the background of planetary vorticity. The fact that a tripole has appeared in our results and in observations [161, 135] suggests that these vortices may play a more prominent role in geophysical fluid dynamics than previously thought.

In [8], tripole vortices are shown to emerge from a perturbation vorticity applied to a monopole vortex. In planar simulations, a Gaussian vortex is perturbed with a vortex quadrupole and tripoles are observed to emerge once the perturbation amplitude exceeds a critical value that depends on the Reynolds number [8]. Repeating these experiments in spherical geometry with rotation could shed light on the seemingly rare tripole vortex structures in geophysical applications, and pose a new way to view the behavior of the winter hemisphere polar vortex during increasingly frequent sudden stratospheric warming events.

7.3 Next steps

Our immediate plans for LPPM involve improving its efficiency and its accuracy during long time simulations. LPPM provides accurate solutions, comparable to today's existing dynamical cores, so long as the flow is resolved. If the flow develops features too small for the Lagrangian panels to resolve, our current algorithms lose conservation of energy and develop a checkerboard pattern in their relative vorticity distributions due to the combination of midpoint rule quadrature and the Lagrangian remeshing procedure. Thus, our first two tasks for improving LPPM are aimed at the quadrature scheme and the remeshing procedure.

7.3.1 Improved quadrature

The choice of the midpoint rule as our discretization method for the Biot-Savart integral was guided by its simplicity and the success it has had in other algorithms based on vortex methods. Although our data structures were built to compute midpoint rule approximations of integrals, we can implement an upgraded quadrature scheme without having to change the existing data structures. Currently, our software assigns the value of a variable ϕ or vorticity ζ at the active particle to the entire panel, as illustrated by figure 7.8 (a). This technique results in the integral approximation at each panel of

$$\int_{P_j} \phi(\boldsymbol{x})\, dA \approx \phi_j A_j \tag{7.5}$$

where ϕ_j is the data value carried by panel P_j's active particle and A_j is the panel area. The improved method, which is similar to the isoparametric method from [5], would use passive particle data as well as active particle data to build a piecewise linear sub grid reconstruction on each panel. As a first attempt, we may choose a straightforward implementation of barycentric weights on each triangular panel.

$$\int_{P_j} \phi(\boldsymbol{x})\, dA \approx \phi_1 A_1 + \phi_2 A_2 + \phi_3 A_3 \tag{7.6}$$

where $A_{1,2,3}$ are the areas of the sub triangles opposite vertex $\boldsymbol{y}_{1,2,3}$, shaded in colors in figure 7.8 (b). Tracer values $\phi_{1,2,3}$ are the values carried by panel P_j's passive particles $\boldsymbol{y}_{1,2,3}$. This method does not use the data carried by the active particle, and thus does not take advantage of all of the available data.

A better method that would generalize to polygons of any type would be to compute the midpoint rule on each sub triangle. For simplicity, assume panel P_j is triangular. Let A_i be the area of the panel's sub triangles, listed in counterclockwise order as shown in figure 1.7 (a), and let ϕ_1 correspond to the vertex incident to sub triangle T_1 and sub triangle T_3. Thus, in figure 7.8 (b), T_1 is colored green, T_2 is blue, and T_3 is orange. A quadrature scheme for a triangular panel using sub triangles with this ordering is

$$\int_{P_j} \phi(\boldsymbol{x})\, dA \approx \frac{1}{3}\left(\phi_1 + \phi_2 + \phi_j\right) A_1 + \frac{1}{3}\left(\phi_2 + \phi_3 + \phi_j\right) A_2 + \frac{1}{3}\left(\phi_3 + \phi_1 + \phi_j\right) A_3. \tag{7.7}$$

For an $n-$sided polygon we define

$$\overline{\phi_i} = \frac{1}{3}\left(\phi_i + \phi_{i+1} + \phi_j\right) \tag{7.8}$$

as the average value of ϕ on each sub triangle of panel P_j, where $i+1$ is cyclic in n, so that if $i = n$, then $i + 1 = 1$. Then (7.7) generalizes to

$$\int_{P_j} \phi(\boldsymbol{x})\, dA \approx \sum_{i=1}^{n} \overline{\phi_i} A_i. \tag{7.9}$$

The sub grid reconstruction quadrature scheme uses all of the data we currently carry in our data structures, which means it can be implemented quickly without having to change our fundamental code. The checkerboard error we have seen in relative vorticity, which is caused when a panel straddles a filament at the same or smaller scale than its edge length,

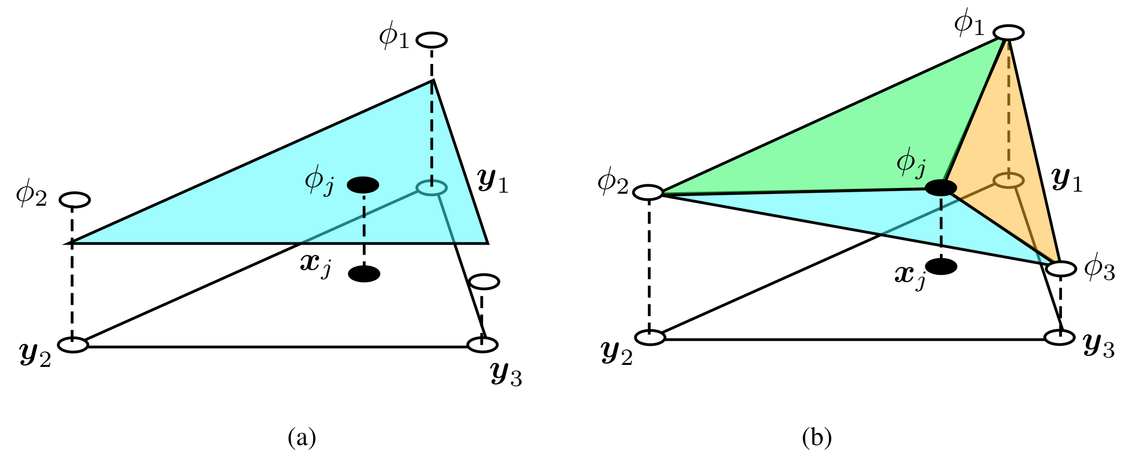


Figure 7.8: Triangular panel with active particle $\boldsymbol{x}_j$ and passive particles $\boldsymbol{y}_i$ for $i = 1, 2, 3$. Tracer values $\phi_{i,j}$ are shown as heights above their associated particle. Midpoint rule (a) assigns ϕ_j to the entire panel; tracer values at passive particles ϕ_i are not used in the quadrature rule. Improved method (b) uses all particle data in a piecewise linear sub-grid reconstruction.

will be reduced. Panels will now be able to resolve features at scales on the order of the distance between their active particle and their passive vertices. In chapter IV, we saw that checker boarding error in relative vorticity is associated with a loss of conservation of energy and enstrophy, so we expect that improving the quadrature scheme to use (7.9) instead of (7.5) will also improve the conservation properties of our model.

7.3.2 Lagrangian remeshing to intermediate times

We have shown that adaptive Lagrangian remeshing is capable of removing the error due to small scales developing, but this procedure comes with the high cost of rapidly increasing N, especially if the Lagrangian parameter is mapped all the way back to $t = 0$ at each remeshing. A new option for our remeshing procedure will redefine the Lagrangian parameter at each remeshing step, to avoid the accumulation of difficult to resolve small scale features in the flow map.

To illustrate the new procedure, we assume a set of particles $(\boldsymbol{x}_j(t), \boldsymbol{y}_i(t))$ has reached the first remeshing step $t = t_{rm}$. As before, a new set of particles $(\hat{\boldsymbol{x}}_k(t_{rm}), \hat{\boldsymbol{y}}_l(t_{rm}))$ are built.

The old particles have Lagrangian parameters defined at $t = 0$, so that

(7.10a) $$\boldsymbol{x}_j(0) = \boldsymbol{\alpha}_j$$

(7.10b) $$\boldsymbol{y}_i(0) = \boldsymbol{\alpha}_i.$$

A tracer ϕ is associated via the flow map to $\boldsymbol{x}_j$ and $\boldsymbol{y}_i$ by

(7.11) $$\phi_j = \phi(\boldsymbol{x}_j(t)) = \phi(\boldsymbol{\alpha}_j)$$

(7.12) $$\phi_i = \phi(\boldsymbol{y}_i(t)) = \phi(\boldsymbol{\alpha}_i).$$

In the intermediate stage of the new remeshing procedure, these Lagrangian parameters are assigned to the new particles using inverse interpolation, preserving the flow map as before. If particles $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ had been in the calculation at $t = 0$, they would satisfy (to within interpolation error), the same relationship as (7.10),

(7.13a) $$\hat{\boldsymbol{x}}_k(0) = \boldsymbol{\alpha}_k$$

(7.13b) $$\hat{\boldsymbol{y}}_l(0) = \boldsymbol{\alpha}_l$$

where $\boldsymbol{\alpha}_k$ and $\boldsymbol{\alpha}_i$ are the interpolated Lagrangian parameter.

New data are then assigned to particles $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ using the material invariant quantities of the current problem. For example, new values of an advected tracer ϕ would be assigned as

(7.14a) $$\hat{\phi}_k = \phi(\boldsymbol{\alpha_k})$$

(7.14b) $$\hat{\phi}_l = \phi(\boldsymbol{\alpha}_l).$$

Since this is the first remeshing, the original distribution ϕ can be used in (7.14). In the current remeshing procedure, the Lagrangian parameter associated with $t = 0$ is kept with the new particles, and the original distribution functions are used throughout the computation to assign new data at each remeshing.

In the new procedure, the interpolated Lagrangian parameter is replaced with particles' current positions. This creates a new Lagrangian parameter $\boldsymbol{\alpha}^{(1)}$ associated with $t = t_{rm}$ instead of $t = 0$,

$$\boldsymbol{\alpha}_k^{(1)} = \hat{\boldsymbol{x}}_k(t_{rm}) \tag{7.15a}$$

$$\boldsymbol{\alpha}_l^{(1)} = \hat{\boldsymbol{y}}_l(t_{rm}). \tag{7.15b}$$

The original set of particles $\boldsymbol{x}_j(t)$ and $\boldsymbol{y}_i(t)$ are discarded, and particles $\hat{\boldsymbol{x}}_k(t)$ and $\hat{\boldsymbol{y}}_l(t)$ are advanced in time until the next remeshing, $t = t_{rm+1}$. At the next remeshing, another set of particles is created. The Lagrangian parameter $\boldsymbol{\alpha}^{(1)}$ is interpolated to the new particles from $\hat{\boldsymbol{x}}_k(t_{rm+1})$ and $\hat{\boldsymbol{y}}_l(t_{rm+1})$. Materially invariant data are assigned to the new particles using $\hat{\boldsymbol{x}}_k$, $\hat{\boldsymbol{y}}_l$, and $\boldsymbol{\alpha}^{(1)}$. The interpolated Lagrangian parameter $\boldsymbol{\alpha}^{(1)}$ provides a link from the current time $t = t_{rm+1}$ to the previous remeshing, $t = t_{rm}$.

The new particles create $\boldsymbol{\alpha}^{(2)}$ by overwriting the interpolated $\boldsymbol{\alpha}^{(1)}$ with their current positions, which now provides a correspondence between the new particles and $\hat{\boldsymbol{x}}_k(t_{rm+1})$ and $\hat{\boldsymbol{y}}_l(t_{rm+1})$.

This procedure preserves the flow map over equal increments of time, whereas our original remeshing procedure preserved the flow map over increasingly large increments of time. Most importantly, the equal increments of time don't allow small scale features to accumulate in the flow map as they have, for example, in figure 7.5. At each remeshing the Lagrangian parameter is refreshed, and small-scale information held within it is erased, but through the assignment of materially invariant data via the flow map, small scale features in tracers and vorticity are maintained. We anticipate that implementing this change in the Lagrangian remeshing procedure may dramatically reduce the number of panels required to resolve flow features as t increases.

Redefining the Lagrangian parameter incrementally throughout the calculation would be better classified as a semi-Lagrangian method, rather than a fully Lagrangian method. It may allow us to continue to avoid the use of numerical diffusion, but maintaining the conservation properties of our Lagrangian remeshing procedure will be a more difficult challenge as the $t = 0$ Lagrangian data are discarded. We plan to include both the current Lagrangian remeshing procedure and this new semi-Lagrangian option in the next version of the LPPM software.

7.3.3 Treecode algorithms

Our current implementation of the Biot-Savart law integral (3.26) uses direct summation to calculate velocity. As N grows, the cost of the velocity computation scales as $\mathcal{O}\left(N^2\right)$ because direct summation calculates the interaction of every fluid particle with every active particle. However, the effect of each active particle on a particular location's velocity diminishes with distance; far away active particles have much less impact on a location's velocity than active particles in its immediate vicinity.

Decomposing the active particles into a quad tree can reduce the time required to compute the velocity at one location from an $\mathcal{O}(N)$ direct summation to an $\mathcal{O}(\log N)$ tree code sum by replacing computations of far-away interactions with more rapidly computed approximations. This would result in reducing the overall cost of each time step from $\mathcal{O}\left(N^2\right)$ to $\mathcal{O}(N \log N)$, a considerable savings.

Tree code algorithms are commonly applied in $N-$body problems for the sake of computational efficiency. They have been used for vortex sheet dynamics in 3D [118, 57, 94] as well as computations of interacting vortex sheets on a sphere [158]. Applying them to LPPM is essential in order to provide an efficient dynamical core at high-resolution and large N.

7.4 Future research

7.4.1 Shallow water test cases

Chapter VI presents the shallow water equations using a Lagrangian formulation of their vorticity-divergence form. While we are implementing the software improvements identified in section 7.3 we will also be experimenting with the strategies for evaluating spatial derivatives in Lagrangian particle methods which were introduced in chapter VI. Once we have chosen a suitable strategy for evaluating $J(\boldsymbol{u})$ and $\nabla^2\boldsymbol{u}$ when $\boldsymbol{u}$ is given by a Biot-Savart law, we will perform the shallow water dynamical core test cases in a very similar manner to the way we have evaluated the barotropic dynamical core produced in this thesis.

The majority of the shallow water test cases are due to Williamson *et al.* [187]. We have already completed an evaluation of our advection scheme in chapter V. The next test cases, like the ones presented in chapter IV, increase in complexity and include unstable steady states of the nonlinear shallow water equations.

Several test cases used to simulate realistic flows such as transitory low pressure systems and flow over topography are included, and high-resolution solutions from a proven spectral transform dynamical core are provided for comparison. Also included in these tests are a shallow water version of the RH4 test case from section 4.2. A Rossby-Haurwitz wave is not an analytic solution of the shallow water equations, so there is no exact solution to compare computed solutions to; instead, this test case looks for physically appropriate correlations between the fluid height h and the vorticity ζ. Cyclonic vorticity should correspond to smaller fluid thickness, while anticyclones should have larger fluid thickness.

Similar to our work with the stratosphere model (section 4.5), the authors of [187] suggest using past observations of 500 hPa flow to initialize a shallow water model, and to compare the resulting forecasts to observed data at corresponding times as a way to comment on the physical realism of the model solutions.

Galewsky *et al.* [67] provide an additional test case that is defined with balanced analytic initial conditions, and develops many realistic flow characteristics due to nonlinear interactions between vorticity and divergence as t increases. This test case is set up as a steady midlatitude zonal jet, like section 4.4, with a perturbation added to the height field. This perturbation causes growing oscillations in the jet due to the flow's barotropic instability, and like the later test cases in [187], computed solutions should be compared to high-resolution reference solutions for evaluation.

Once we are satisfied with our method's solutions of the standard shallow water dynamical core test cases, new test cases can be produced to examine flow features captured by the shallow water approximation. Significantly, these include interactions between waves and vorticity [164] and shallow water gravity waves, which can be used to model tsunamis [76].

7.4.2 Voronoi diagrams and Delaunay triangulations

We have mentioned the properties of Voronoi diagrams and Delaunay triangulations that make them attractive as the basis of a moving unstructured horizontal discretization of the unit sphere. However, we have also not yet had the success we had anticipated while using them. This is due to the challenge of performing adaptive refinement on a native Voronoi grid, and difficulties with the STRIPACK [146] software library's Voronoi algorithm [25].

Despite these difficulties, we also acknowledge that the success of our Lagrangian remeshing procedure is partially based on the Delaunay triangulation of our fluid particles. The interpolation library SSRFPACK [147] uses the Delaunay triangulation produced by STRIPACK to perform its cubic interpolation procedure.

We have begun developing our own implementation of the incremental Voronoi algorithm provided by [7] in Fortran, in order to experiment with adaptive refinement procedures that do not rely on recursive division of fixed-type polygons. Additional Voronoi diagram software is being developed for parallel computing environments by [90] and the Computational Geometry Algorithms Library (CGAL) [172, 40]. Both of these software packages are written in C++, which would require a major change to our existing software to use.

APPENDIX

APPENDIX

Interpolation of Scattered Data on a Sphere

The remeshing procedures introduced in chapter II are dependent upon accurate interpolation of the Lagrangain parameter from scattered data on the surface of a sphere. A review of many such interpolation methods is provided by [55], and several methods are investigated on spherical geodesic grids in [24].

The interpolation problem may be stated as follows: Given a set of scattered points $\{\boldsymbol{x}_i\}_{i=1}^n$ lying on $\mathcal{S} \subset \mathbb{R}^3$, and real numbers $\{\phi_i\}_{i=1}^n$ that represent data defined on $\{\boldsymbol{x}_i\}$, we seek a function g defined on $\mathcal{S}$ that either *interpolates* the data $\{\phi_i\}$ in the sense that

$$g(\boldsymbol{x}_i) = \phi_i, \quad i = 1, \dots, n \tag{ A.1a}$$

or *approximates* the data $\{\phi_i\}$ in the sense that

$$g(\boldsymbol{x}_i) \approx \phi_i, \quad i = 1, \dots, n. \tag{ A.1b}$$

Typically, we want the function g to be continuous, $g \in C^0$, but in many cases it may be desirable for g to possess at least a finite number of continuous derivatives $g \in C^r$ where $r \geq 1$ so that the surface

$$\mathcal{F} = \{(\boldsymbol{x}, g(\boldsymbol{x})) : \boldsymbol{x} \in \mathcal{S}\} \tag{ A.2}$$

is tangent plane continuous or differentiable [55].

A.1 δ-convolution approximation

One of our initial attempts at solving (A.1) was an infinitely smooth approximating function based on Green's function (3.17) and its definition as the solution of (3.18), which involves a delta function. A delta function $\delta_S(\boldsymbol{x}) = \delta_S(\lambda, \theta)$ on $\mathcal{S}$ is a distribution that satisfies the equations

$$\delta_S(\boldsymbol{x}) = 0 \quad \text{for } \boldsymbol{x} \neq 0, \tag{ A.3a}$$

$$\lim_{\epsilon \to 0} \int_{-\epsilon}^{\epsilon} \int_{-\epsilon}^{\epsilon} \delta_S(\lambda, \theta) \cos\theta \, d\theta d\lambda = 1, \tag{ A.3b}$$

$$\int_0^{2\pi} \int_{-\pi/2}^{\pi/2} \delta_S(\lambda - \tilde{\lambda}, \theta - \tilde{\theta}) \phi(\tilde{\lambda}, \tilde{\theta}) \cos\tilde{\theta} \, d\tilde{\theta} d\tilde{\lambda} = \phi(\lambda, \theta), \tag{ A.3c}$$

where $\phi : \mathcal{S} \mapsto \mathbb{R}$ is a scalar function.

Zemanian [190] provides a discussion that includes periodic test functions applicable to spherical domains, and Jackson [89] illustrates a formal change of variables transformation to derive delta functions on non-Euclidean spaces.

Definition A.1. The delta function on $\mathcal{S}$, $\delta_S(\boldsymbol{x}) = \delta_S(\lambda, \theta)$, is

$$\delta_S(\lambda, \theta) = \frac{1}{\cos\theta} \delta(\lambda) \delta(\theta), \tag{ A.4}$$

where $\delta(\lambda)$ and $\delta(\theta)$ are Dirac delta functions on $\mathbb{R}$.

Stakgold [165] defines a *delta-sequence* as a family of smooth functions $\delta_\epsilon(\boldsymbol{x})$ with a parameter ϵ that converge in the sense of distributions to a delta function as the parameter ϵ converges to zero,

$$\lim_{\epsilon \to 0} \delta_\epsilon(\boldsymbol{x}) = \delta(\boldsymbol{x}). \tag{ A.5}$$

Computationally we may use delta sequences to approximate the action of a delta function by choosing small positive numbers for the parameter ϵ. A delta sequence can be generated on $\mathcal{S}$ by applying the Laplace-Beltrami operator to the regularized Green's function

$$G_\epsilon(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{1}{4\pi} \log(1 - \boldsymbol{x} \cdot \tilde{\boldsymbol{x}} + \epsilon^2). \tag{ A.6}$$

This implies that the smoothing parameter ϵ has units of arc length, as the quantity $1 - \boldsymbol{x} \cdot \tilde{\boldsymbol{x}}$ is half the square of the chord distance between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ (1.15h). We also note that using the regularized Green's function in chapter III would lead to a vortex blob method. Applying the Laplace-Beltrami operator to (A.6), we arrive at the following definition [184].

Definition A.2. The function

$$\delta_\epsilon(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{-(1 - \boldsymbol{x} \cdot \tilde{\boldsymbol{x}})^2 + 2\epsilon^2 \boldsymbol{x} \cdot \tilde{\boldsymbol{x}}}{4\pi(1 - \boldsymbol{x} \cdot \tilde{\boldsymbol{x}} + \epsilon^2)^2} - \frac{1}{4\pi} \tag{ A.7a}$$

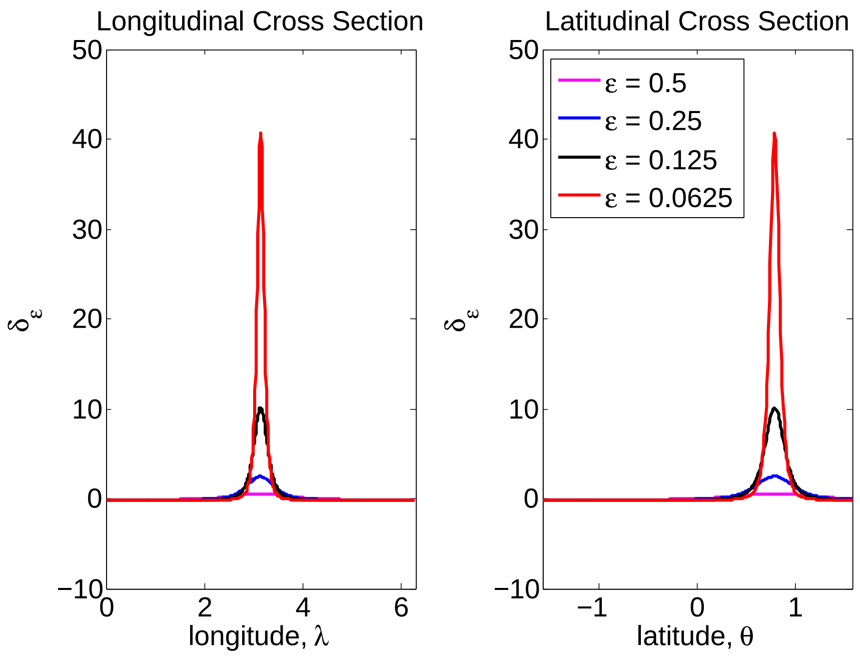
$$\delta_\epsilon(\lambda, \theta, \tilde{\lambda}, \tilde{\theta}) = \frac{-(1 - \cos\gamma)^2 + 2\epsilon^2 \cos\gamma}{4\pi(1 - \cos\gamma + \epsilon^2)^2} - \frac{1}{4\pi} \tag{ A.7b}$$

is a delta sequence on $\mathcal{S}$ as $\epsilon \to 0$, where $\cos\gamma$ is the cosine of the central angle between the vectors $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ given by (1.16). It represents a source at $\tilde{\boldsymbol{x}} \mapsto (\tilde{\lambda}, \tilde{\theta})$. Figure A.1 shows cross sections of this function for several values of the smoothing parameter ϵ. As $\epsilon \to 0$ the width of $\delta_\epsilon(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ decreases and its height increases to converge toward a function that satisfies (A.3).

Equations (A.7) and (A.3c) provide the basis for an approximation method for (A.1) via the identity

$$\phi(\boldsymbol{x}) = \lim_{\epsilon \to 0} \int_{\mathcal{S}} \delta_\epsilon(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \phi(\tilde{\boldsymbol{x}}) d\tilde{A}. \tag{ A.8}$$

We may approximate the integral (A.8) with particles and panels using (A.7) by considering the active particles as the sources $\tilde{\boldsymbol{x}}$ for some quantity $\phi(\boldsymbol{x})$ defined at each active particle. Let $\boldsymbol{x}_j(t)$ denote the active particle positions at time t and ϕ_j be the function's values defined on the active particles at time t. Quadrature weights are given by the panel areas A_j and the data values ϕ_j. We choose an $\epsilon > 0$. Let $\boldsymbol{x}_0 \in \mathcal{S}$ be a location where we



(a) (b)

Figure A.1: Plots of $\delta_\epsilon(\lambda, \theta, \lambda_0, \theta_0)$ for $\lambda_0 = \pi$, $\theta_0 = \pi/4$ for various smoothing parameters ϵ. (a) A longitudinal cross section along $\theta = \theta_0$. (b) A latitudinal cross section along $\lambda = \lambda_0$.

desire an approximation using g, then

$$\text{(A.9)} \qquad \phi(\boldsymbol{x}_0) \approx g(\boldsymbol{x}_0) = \frac{1}{4\pi}\sum_{j=1}^{N} \frac{-(1-\boldsymbol{x}_0\cdot\boldsymbol{x}_j(t))^2 + 2\epsilon^2\boldsymbol{x}_0\cdot\boldsymbol{x}_j(t)}{(1-\boldsymbol{x}_0\cdot\boldsymbol{x}_j(t)+\epsilon^2)^2}\phi_j A_j - \frac{1}{4\pi}\sum_{j=1}^{N}\phi_j A_j.$$

Approximations using equation (A.9) are subject to two types of error: discretization error due to quadrature and smoothing error due to the parameter ϵ. We apply this approximation to the problem of interpolating a cosine bell considered by [24]. The cosine bell is defined on the unit sphere by scaling the cosine bell tracer from [187],

$$\text{(A.10)} \qquad \phi_{CB}(\boldsymbol{x}) = \begin{cases} \frac{h_0}{2}\left(1+\cos\left(\frac{\pi\,\mathrm{dist}(\boldsymbol{x},\boldsymbol{x}_0)}{R}\right)\right) & \text{if } \mathrm{dist}(\boldsymbol{x},\boldsymbol{x}_0) < R \\ 0 & \text{otherwise} \end{cases},$$

where $\mathrm{dist}(\boldsymbol{x},\boldsymbol{x}_0)$ is the geodesic distance (1.19) between $\boldsymbol{x}$ and the center of the cosine bell $\boldsymbol{x_0} \mapsto (\lambda_0, \theta_0)$. The parameters are $h_0 = 0.0016$, $R = 1/3$, $\lambda_0 = \theta_0 = 0$.

Figure A.2 shows error convergence as N increases for various values of the smoothing

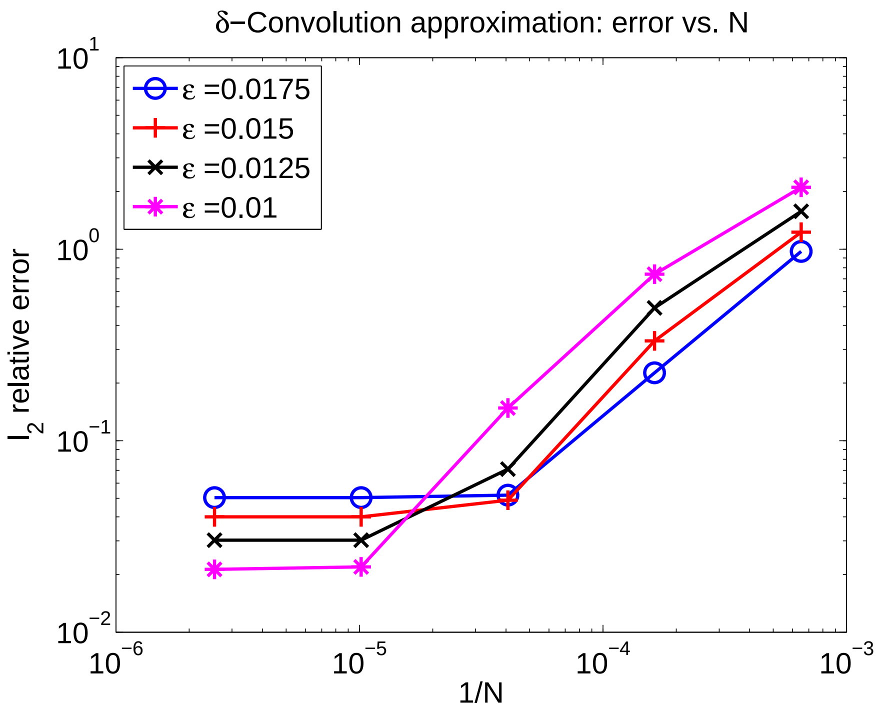


Figure A.2: Approximating a cosine bell on $\mathcal{S}$ with δ-convolution using uniform meshes: l_2 relative approximation error vs. N for various smoothing pameters ϵ.

parameter ϵ. The smallest smoothing parameters (black and magenta curves) require the largest N to resolve, because the smooth delta functions are finer than coarse grids' panels. This is an example of discretization error, as it is caused by unresolved features. As N increases, error converges to zero at a rate proportional to $1/N$, which is consistent with midpoint rule quadrature. In contrast, the blue and red curves show error decreasing as N increases until a saturation point is reached. The saturated error is smoothing error. For a fixed ϵ, smoothing error does not decrease as N increases because the approximating functions do not exactly satisfy (A.8).

The same data are shown in figure A.3, which plots the interpolation error as a function of the smoothing parameter ϵ for various grid sizes. Each curve has a minimum error which indicates the optimum choice of smoothing parameter for that grid size. To the right of the minimum for a particular curve, we see error due to smoothing because ϵ is

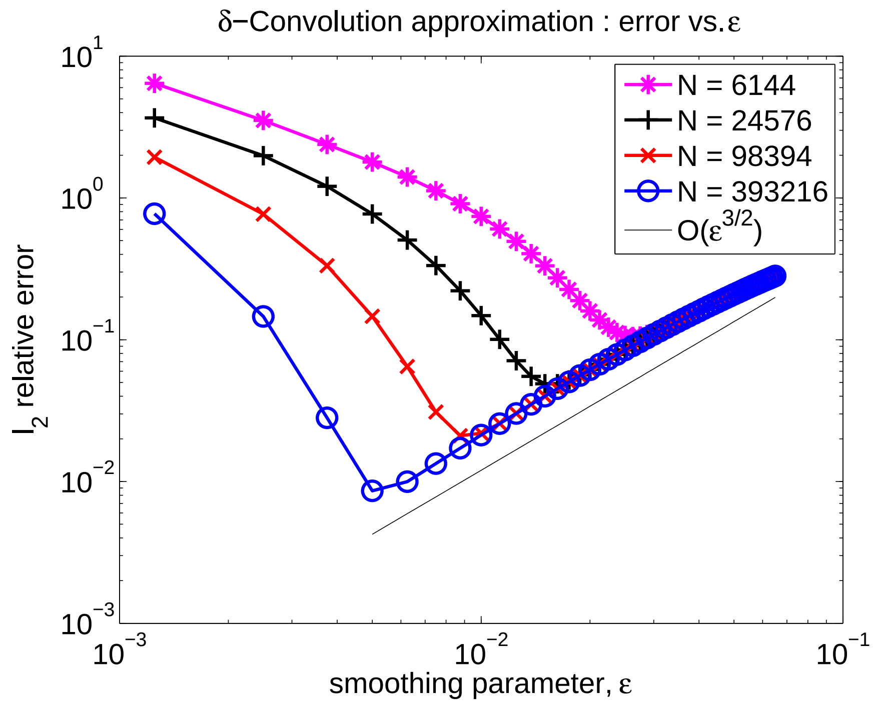


Figure A.3: Approximation error of a cosine bell with δ-convolution on quadrilateral panels for various N vs. smoothing parameter ϵ.

too large for that grid size. To the left of the minimum, the increased error is discretization error – the grid is not capable of resolving the increasingly fine peaks of the approximate delta functions. From right-to-left toward the minimum, error convergence as $\epsilon \to 0$, is $O\left(\epsilon^{3/2}\right)$, as indicated by the slope of the thin black line.

These results suggest that an optimal value for the ratio of mesh size $\Delta\lambda$ to smoothing parameter ϵ is approximately

$$\frac{\Delta\lambda}{\epsilon} \in [0.75, 1). \tag{A.11}$$

For the computations shown in figure A.3, the optimal ratio is $\Delta\lambda/\epsilon = 0.782$.

In vortex blob methods, convergence proofs are based on an assumption that the smoothing parameter ϵ varies as a fractional power of the mesh size [74, 142, 14], and in practice numerical accuracy is lost when smoothed particles cease to overlap [38, 97]. This loss of accuracy is depicted in figure A.3 by the increase in error as $\epsilon \to 0$. The ratio (A.11) is another verification of these results. It is important to distinguish this approximation error due to particles not overlapping at small values of the smoothing parameter from the velocity error discussed in chapter IV.

Point vortices do not overlap at all, and correspond to the limit $\epsilon \to 0$ of a vortex blob method. We saw that point vortices were capable of accurately approximating velocity fields via the Biot-Savart law in chapter IV, which at first seems contrary to the results shown in figure A.3. However, the Biot-Savart kernel (3.22) has a weaker singularity than the approximate δ-kernel (A.7) and point vortices are therefore capable of converging to the correct *velocity* field solution without smoothing. Point vortices would require smoothing to approximate the *vorticity* field, but this was not the objective of our work with the barotropic vorticity equation in chapters III and IV.

Approximation with δ-convolution is similar to interpolation with radial basis functions (RBF). In both methods, the influence of each source data point is a function of the radial distance from that point. As an approximation method, in δ-convolution there is no need to solve a dense $N \times N$ linear system for the interpolation weights, as there is in an RBF method. This avoids an expensive and ill-conditioned calculation [56], but at the same time the lack of an interpolation matrix is likely responsible for the lower accuracy of δ-convolution approximations on this cosine bell interpolation problem compared to results from collocation methods using RBF [24].

The advantages of δ-convolution approximation are that it provides an infinitely smooth approximating surface, and, as a quadrature based method, it is easily implemented on the particle-panel data structures we have already developed. It is inexpensive relative to an interpolating method that would require the solution of a dense $N \times N$ interpolating linear system. It also provides a first step toward a means of approximating the forcing terms of the shallow water equations, as discussed in the regularized Biot-Savart kernel strategy of chapter VI.

The disadvantage of approximating surfaces with δ-convolution is its accuracy. The errors shown in figures A.2 and A.3 for delta-convolution approximation are much larger than both piecewise linear interpolation and cubic polynomial interpolation methods at the same grid resolutions [24]. Improving the quadrature scheme (chapter VII) and applying our adaptive refinement algorithms to δ-convolution approximation problems could improve the accuracy. These items are marked for future study.

A.2 Piecewise linear interpolation and Shepard's method

The first local interpolation algorithm we discuss uses barycentric coordinates on a triangle to perform linear interpolation. Linear interpolation requires three source data points to define a local plane. We describe an interpolation procedure that will work on polygons of arbitrary type, using the sub triangles from the triangular decomposition of our Lagrangian panels (figure 1.7).

Let $\boldsymbol{x}_0$ represent a point where we need an interpolated value. First, $\boldsymbol{x}_0$ is located within the panel set using the point-query algorithm shown in figure 2.8. Within that panel, the sub triangle that contains $\boldsymbol{x}_0$ is located. This sub triangle provides the source data for the linear interpolation procedure illustrated in figure A.4.

One of the sub triangle's vertices is the panel's active particle $\boldsymbol{x}_j$; the other two vertices $\boldsymbol{y}_A$ and $\boldsymbol{y}_B$ are passive particles. Let A_T denote the area of the entire sub triangle. The interpolation point $\boldsymbol{x}_0$ introduces three smaller triangles within the panel sub triangle shown in figure A.4.

Each of these three smaller triangles provide the interpolation weights w_i for $i = 1, 2, 3$,

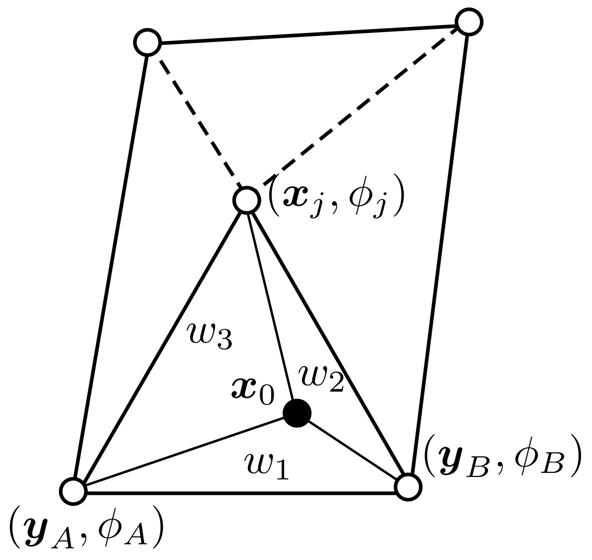

$(\boldsymbol{x}_j, \phi_j)$
w_3
$\boldsymbol{x}_0$
w_2
$(\boldsymbol{y}_B, \phi_B)$
w_1
$(\boldsymbol{y}_A, \phi_A)$

Figure A.4: Linear interpolation on a panel sub-triangle with barycentric weights.

(A.12a) $$w_1 = \frac{\text{Area}(\boldsymbol{x}_0, \boldsymbol{y}_A, \boldsymbol{y}_B)}{A_T}$$

(A.12b) $$w_2 = \frac{\text{Area}(\boldsymbol{x}_0, \boldsymbol{y}_B, \boldsymbol{x}_j)}{A_T}$$

(A.12c) $$w_3 = \frac{\text{Area}(\boldsymbol{x}_0, \boldsymbol{x}_j, \boldsymbol{y}_A)}{A_T},$$

where the area function Area$(\boldsymbol{x}_A, \boldsymbol{x}_B, \boldsymbol{x}_C)$ is given by (1.22). Note that $\sum_{i=1}^3 w_i = A_T$.

The weights (A.12) provide the interpolating function

(A.13) $$\phi(\boldsymbol{x}_0) \approx g(\boldsymbol{x}_0) = \phi_j w_1 + \phi_A w_2 + \phi_B w_3.$$

The definition of the interpolating weights ensures that condition (A.1a) is met at each vertex of the sub triangle.

Another first-order accurate local interpolation scheme is given by Shepard's method [26, 24]. If the destination point $\boldsymbol{x}_0$ is equal to any of the particles in the panel, then the data associated with that particle is assigned to $\boldsymbol{x}_0$ to satisfy (A.1a); otherwise interpolation weights are computed as inverse powers of distance between the destination point and each source data point. The weight associated with the active particle is $w_0 = \text{dist}(\boldsymbol{x}_0, \boldsymbol{x}_j)^{-p}$, and each passive particle has a weight given by $w_i = \text{dist}(\boldsymbol{x}_0, \boldsymbol{y}_i)^{-p}$ for $i = 1, \ldots, 4$. The parameter p is a positive real number; typically $p = 2$, but we have found better accuracy with $p = 3$, which was suggested by [26]. Shepard's interpolation method is given by

$$g(\boldsymbol{x}_0) = \begin{cases} g_0 & \text{if } \boldsymbol{x_0} = \boldsymbol{x}_j \\ g_i & \text{if } \boldsymbol{x_0} = \boldsymbol{y}_i \quad 1 \le i \le 4 \\ \frac{1}{\sum_{i=0}^4 w_i} \left(\sum_{i=0}^4 \phi_i w_i \right) & \text{otherwise} \end{cases} \cdot \tag{A.14}$$

As it is a local method, it begins by first locating the panel that contains $\boldsymbol{x}_0$ using our point query algorithm (figure 2.8). Shepard's method uses all particles (both active and passive) from the panel containing $\boldsymbol{x}_0$.

This method is subject to possible ill conditioning when $\boldsymbol{x}_0$ is very close to one of the particle locations; in this case the weight associated with that particle approaches infinity. A variation that avoids having to compare equivalence of finite precision real numbers adds a smoothing parameter to the weight,

$$w_i = \left(\text{dist}(\boldsymbol{x}_0, \boldsymbol{y}_i) + \epsilon\right)^{-p}. \tag{A.15}$$

This variation makes the method an approximation scheme, rather than an interpolation scheme.

The addition of a smoothing parameter makes this version of Shepard's method similar to the global approximation using δ-convolution introduced in the previous section. Numerical experiments with each of the methods introduced so far, δ-convolution, piecewise linear interpolation, Shepard's interpolation, and the smoothed variation of Shepard's method shows that they are only first-order accurate in N as $N \to \infty$. Used as part of the Lagrangian remeshing procedure introduced in this thesis, they give similar results in both accuracy and convergence rates to the piecewise linear data shown in figure 5.6.

As established in the barotropic vorticity equation and advection equation results, the accuracy of our algorithm is most affected by the accuracy of the interpolation method used to perform the Lagrangian remeshing. We conclude that higher order interpolation methods are necessary, which is why we chose the cubic Hermite interpolation method provided by the SSRFPACK software mentioned in chapter II.

A.3 High-order interpolation

Developing high order accurate methods on our current data structures could begin with the interpolation and approximation schemes discussed in the previous section. The global δ-convolution approximation could be made more accurate with a higher order quadrature scheme, or more accurate interpolation kernels [53]. The local piecewise linear interpolation scheme could be modified to use higher degree interpolating polynomials. This may require adding source data to the interpolation from panels adjacent to the panel that contains the destination point, but the data structures introduced in chapter II are capable of quickly providing adjacency information. Similar to methods used by [130], a polynomial of the desired degree could be obtained by least-squares fitting of data from active and passive particles from any number of adjacent panels.

As discussed in previous chapters, we have so far relied on the SSRFPACK software library [147] for remeshing interpolation. It provides a cubic Hermite interpolation of scattered data using the Delaunay triangulation of the source point locations. We chose SSRFPACK because it is accurate, convenient, and efficient for geophysical applications (it is included in distributions of the NCAR Command Language [180]). It is written in Fortran 77 and was therefore easy to link with our code. In addition, its use of Delaunay triangulations provided a quick introduction to their associated algorithms and Voronoi diagrams, which we had wanted to investigate. Perhaps most significantly, these algorithms complete in $O(N)$ time, which means that a remeshing procedure using SSRFPACK will run in asymptotically smaller time than the time required to compute one time step, which currently is $O\left(N^2\right)$, due to the direct summation used to compute the Biot-Savart integral. This will still be an advantage of using SSRFPACK even after the addition of tree code algorithms, which will reduce time step computations to $O(N \log N)$.

Of course, other interpolation methods are possible; the review provided by [55] contains 206 references and is already fifteen years old. SSRFPACK is familiar to the dynamical core community, but Radial Basis Functions (RBF) are becoming more common in vortex methods [9, 10], and increasingly found in geophysical applications as well [184, 189, 24]. Additionally, recent efforts have produced RBF methods that provide interpolating functions that possess useful properties in addition to a specified number of continuous derivatives. A procedure for using RBF interpolation of scattered data that provides a divergence-free interpolating vector function was presented by [133, 65]; methods that provide irrotational vector interpolations are possible using the same technique [64]. Partition of unity methods, when combined with RBF, may reduce the computational cost of inverting the dense, ill-conditioned interpolation matrix typically associated with RBF methods [27, 189].

The data structures we use, face-edge-vertex relations and winged-edges specifically, originated in the fields of computer graphics and computational geometry. Methods of interpolating scattered data developed for those fields may provide a suitable method for remeshing interpolations in our algorithms. An example of a possible method from those fields that may be of use is the plate-fitting method for approximating surfaces defined by point clouds [44].

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] J.C. Adams, W.S. Brainerd, R.A. Hendrickson, R.E. Maine, J.T. Martin, and B.T. Smith. *The Fortran 2003 Handbook*. Springer, 2009.

[2] J.M. Alam and J.C. Lin. Toward a fully-Lagrangian atmospheric modeling system. *Monthly Weather Review*, 136:4653–4667, 2008.

[3] J. Allen. NASA Visible Earth. `http://visibleearth.nasa.gov/`, 20-NOV-2006. iceland_tmo_2006324_lrg.jpg.

[4] J. Allen. NASA Visible Earth. `http://visibleearth.nasa.gov/`, 28-OCT-2012. sandy_vir_2012302.jpg.

[5] K. Atkinson. Numerical integration on the sphere. *Journal of the Australian Mathematical Society (Series B)*, 23:332–347, 1982.

[6] J. Augenbaum. A Lagrangian method for the shallow water equation based on Voronoi mesh: Flows on a rotating sphere. In M.J. Fritts, W.P. Crowley, and H. Trease, editors, *The Free-Lagrange Method*, pages 54–86. Springer-Verlag, 1985.

[7] J. Augenbaum and C. Peskin. On the construction of the Voronoi mesh on a sphere. *Journal of Computational Physics*, 59:177–192, 1985.

[8] L. Barba and A. Leonard. Emergence and evolution of tripole vortices from net-circulation initial conditions. *Physics of Fluids*, 19:017101, 2007.

[9] L.A. Barba, A. Leonard, and C.B. Allen. Advances in viscous vortex methods– meshless spatial adaptation based on radial basis function interpolation. *International Journal for Numerical Methods in Fluids*, 47:387–421, 2005.

[10] L.A. Barba and L.F. Rossi. Global field interpolation for particle methods. *Journal of Computational Physics*, 229:1292–1310, 2010.

[11] A. P. Bassom and A. D. Gilbert. The spiral wind-up of vorticity in an inviscid planar vortex. *Journal of Fluid Mechanics*, 371:109–140, 1998.

[12] G.K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, 2000.

[13] J.R. Baumgardner and P.O. Frederickson. Icosahedral discretization of the two-sphere. *SIAM Journal of Numerical Analysis*, 22:1107–1115, 1985.

[14] J.T. Beale and A. Majda. High order accurate vortex methods with explicit velocity kernels. *Journal of Computational Physics*, 58:188–208, 1985.

[15] J. Behrens. *Adaptive atmospheric modeling: Key techniques in grid generation, data structures, and numerical operations with applications*. Springer, 2006.

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

[17] A.J. Bernoff and L.F. Lingevitch. Rapid relaxation of an axisymmetric vortex. *Physics of Fluids*, 6:3717–3723, 1994.

[18] V. Bjerknes. Das Problem der Wettervorhersage, betrachtet vom Standpunkte der Mechanik und der Physik (The problem of weather forecasting as a problem in mechanics and physics). *Meteorologische Zeitschrift*, 21:1–7, 1904. English translation by Y. Mintz, 1954, reproduced in *The Life Cycles of Extratropical Cyclones*, 1999, American Meteorological Society.

[19] V.A. Bogomolov. Dynamics of vorticity at the sphere. *Fluid Dynamics*, 6:863–870, 1977.

[20] C. Börgers and C.S. Peskin. A Lagrangian method based on the Voronoi diagram for the incompressible Navier-Stokes equations on a periodic domain. In M.J. Fritts, W.P. Crowley, and H. Trease, editors, *The Free-Lagrange Method*, pages 87–113. Springer-Verlag, 1985.

[21] J.P. Boyd. Barotropic equatorial waves: The nonuniformity of the equatorial beta-plane. *Journal of the Atmospheric Sciences*, 42:1965–1967, 1985.

[22] J.P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover, 2nd, revised edition, 2001.

[23] J.P. Boyd and C. Zhou. Three ways to solve the Poisson equation on a sphere with Gaussian forcing. *Journal of Computational Physics*, 228:4702–4713, 2009.

[24] M.F. Carfora. Interpolation on spherical geodesic grids: A comparitive study. *Journal of Computational and Applied Mathematics*, 210:99–105, 2007.

[25] M. Caroli, P.M.M. de Castro, S. Loriot, O. Rouiller, M. Teillaud, and C. Wormser. Robust and efficient Delaunay triangulations of points on or close to a sphere. In *Proceedings of the 9th International Symposium on Experimental Algorithms (SEA)*, pages 462–473. LNCS 6049, May 2010.

[26] R. Cavoretto and A. De Rossi. Numerical comparision of different weights in Shepard's interpolants on the sphere. *Applied Mathematical Sciences*, 4:3425–3435, 2010.

[27] R. Cavoretto and A. De Rossi. Spherical interpolation using the partition of unity method: An efficient and flexible algorithm. *Applied Mathematics Letters*, 25:1251–1256, 2012.

[28] J. Charney. On the scale of atmospheric motions. *Geofysiske Publikasjoner*, 17:1–17, 1948.

[29] J.G. Charney, R. Fjörtoft, and J. von Neumann. Numerical integration of the barotropic vorticity equation. *Tellus*, 2:237–254, 1950.

[30] A. J. Chorin and J. E. Marsden. *A Mathematical Introduction to Fluid Mechanics*. Springer, 3rd edition, 2000.

[31] A.J. Chorin. Numerical study of slightly viscous flow. *Journal of Fluid Mechanics*, 57:785–796, 1973.

[32] A.J. Chorin and P.S. Bernard. Discretization of a vortex sheet, with an example of roll-up. *Journal of Computational Physics*, 13:423–429, 1973.

[33] J. Christiansen. Numerical simulation of hydrodynamics by the method of point vortices. *Journal of Computational Physics*, 13:363–379, 1973.

[34] J. Cohen, M. Barlow, and K. Saito. Decadal fluctuations in planetary wave forcing modulate global warming in late boreal winter. *Journal of Climate*, 22:4418–4426, 2009.

[35] P. Colella and P.R. Woodward. The piecewise parabolic method (PPM) for gas-dynamical simulations. *Journal of Computational Physics*, 54:174–201, 1984.

[36] R. Cortez. Convergence of high-order deterministic particle methods for the convection-diffusion equation. *Communications on Pure and Applied Mathematics*, 50:1235–1260, 1997.

[37] G.-H. Cottet, J. Goodman, and T.Y. Hou. Convergence of the grid-free point vortex method for the three-dimensional Euler equations. *SIAM Journal on Numerical Analysis*, 28:291–307, 1991.

[38] G.-H. Cottet and P. D. Koumoutsakos. *Vortex Methods: Theory and Practice*. Cambridge University Press, 2000.

[39] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry*. Springer, 3rd edition, 2008.

[40] P.M.M de Castro. Personal communication, 2012.

[41] P.M.M. de Castro and O. Devillers. A pedagogic JavaScript program for point location strategies. In *SoCG 2011: Proceedings of the 27th Annual ACM Symposium on Computational Geometry*, pages 295–296. ACM, 2011. `http://www-sop.inria.fr/geometrica/demo/point_location_strategies/`.

[42] P. Degond and S. Mas-Gallic. The weighted particle method for convection-diffusion equations. *Mathematics of Computation*, 53:485–525, 1989.

[43] J.M. Dennis, J. Edwards, K.J. Evans, O. Guba, P.H. Lauritzen, A.A. Mirin, A. St-Cyr, M.A. Taylor, and P.H. Worley. CAM-SE: A scalable spectral element dynamical core for the Community Atmosphere Model. *International Journal of High Performance Computing Applications*, 26:74–89, 2012.

[44] U. Dietz. Fair surface reconstruction from point clouds. In M. Daehlen, T. Lyche, and L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces II*. Vanderbilt University Press, 1998.

[45] D. G. Dritschel. A new spherical shallow-water model: building in wave-vortex decomposition. In *Seminar on Recent Developments in Numerical Methods for Atmospheric and Ocean Modeling*. ECMWF, September 2004. Available at `http://www.ecmwf.int/newsevents/meetings/annual_seminar/2004`.

[46] D. G. Dritschel and L. M. Polvani. The roll-up of vorticity strips on the surface of a sphere. *Journal of Fluid Mechanics*, 234:47–69, 1992.

[47] D.G. Dritschel. Contour dynamics and contour surgery : Numerical algorithms for extended, high-resolution modelling of vortex dynamics in two-dimensional, inviscid, incompressible flows. *Computer Physics Reports*, 10:77–146, 1989.

[48] D.G. Dritschel. On the persistence of non-axisymmetric vortices in inviscid two-dimensional flows. *Journal of Fluid Mechanics*, 371:141–155, 1998.

[49] D.G. Dritschel, L.M. Polvani, and A.R. Mohebalhojeh. The Contour-Advective Semi-Lagrangian algorithm for the shallow water equations. *Monthly Weather Review*, 127:1551–1565, 1999.

[50] G. Dziuk. Finite elements for the Beltrami operator on arbitrary surfaces. In *Partial differential equations and calculus of variations*, volume 1357, pages 142–155. Springer Lecture Notes in Mathematics, 1988.

[51] ECMWF. Re-analysis Project. ERA-40 Atlas Pressure Level Climatologies. `http://www.ecmwf.int/research/era/ERA-40_Atlas/docs/section_D25/parameter_zmzwtp.html`.

[52] M. Ehrendorfer. *Spectral Numerical Weather Prediction Models*. SIAM, 2012.

[53] J. Eldridge, A. Leonard, and T. Colonius. A general deterministic treatment of derivatives in particle methods. *Journal of Computational Physics*, 180:686–709, 2002.

[54] J.D. Eldridge, T. Colonius, and A. Leonard. A vortex particle method for two-dimensional compressible flow. *Journal of Computational Physics*, 179:371–399, 2002.

[55] G. Fasshauer and L. Schumaker. Scattered data fitting on the sphere. In M. Daehlen, T. Lyche, and L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces II*. Vanderbilt University Press, 1998.

[56] G.E. Fasshauer. *Meshfree Approximation Methods with MATLAB*. World Scientific, 2007.

[57] H. Feng, L. Kaganovsky, and R. Krasny. Azimuthal instability of a vortex ring computed by a vortex sheet panel method. *Fluid Dynamics Research*, 41, 2009.

[58] N. Flyer and G.B. Wright. A radial basis function method for the shallow water equations on the sphere. *Proceedings of the Royal Society A*, 465:1949–1976, 2009.

[59] A. Folch, A. Costa, and S. Basart. Validation of the FALL3D ash dispersion model using observations of the Eyjafjallajökull volcanic ash clouds. *Atmospheric Environment*, 48:165–183, 2012.

[60] J. Frank and S. Reich. A particle-mesh method for the shallow water equations near geostrophic balance. *Journal of Computational Physics*, 180:407–426, 2002.

[61] J. Frank and S. Reich. The Hamiltonian particle-mesh method for the spherical shallow water equations. *Atmospheric Science Letters*, 5:89–95, 2004.

[62] W. Freeden. On integral formulas of the (unit) sphere and their application to numerical computation of integrals. *Computing*, 25:131–146, 1980.

[63] S. Fujiwhara. On the growth and decay of vortical systems. *Quarterly Journal of the Royal Meteorological Society*, 49:75–104, 1923.

[64] E.J. Fuselier, F.J. Narcowich, J.D. Ward, and G.B. Wright. Error and stability estimates for divergence-free RBF interpolants on the sphere. *Mathematics of Computation*, 78:2157–2186, 2009.

[65] E.J. Fuselier and G. B. Wright. Stability and error estimates for vector field interpolation and decomposition on the sphere with RBFs. *SIAM Journal of Numerical Analysis*, 47:3213–3239, 2009.

[66] K. Gade. A non-singular horizontal representation. *The Journal of Navigation*, 63:395–417, 2010.

[67] J. Galewsky, R. K. Scott, and L. M. Polvani. An initial-value problem for testing numerical models of the global shallow water equations. *Tellus*, 56:429–440, 2004.

[68] A. E. Gill. *Atmosphere-Ocean Dynamics*. Academic Press, 1982.

[69] J. Gleick. *Chaos: Making a New Science*. Penguin Books, revised edition, 2008.

[70] F.I. Gonzalez, E.N. Milburn, H.M. Bernard, and J.C. Newman. Deep-Ocean Assessment and Reporting of Tsunamis (DART®): Brief Overview and Status Report. In *Proceedings of the International Workshop on Tsunami Disaster Mitigation*, January 1998. National Data Buoy Center `http://www.ndbc.noaa.gov/dart/dart.shtml`.

[71] J. Goodman, T.Y. Hou, and J. Lowengrub. Convergence of the point vortex method for the 2-D Euler equations. *Communications on Pure and Applied Mathematics*, XLIII:415–430, 1990.

[72] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer, 2nd edition, 1983.

[73] R. Haberman. *Applied Partial Differential Equations*. Pearson Prentice Hall, 4th edition, 2004.

[74] O.H. Hald. Convergence of vortex methods for Euler's equations. II. *SIAM Journal on Numerical Analysis*, 16, 1979.

[75] A.B. Hansen, B. Sorensen, P. Tarning-Anderson, J.H. Christensen, J. Brandt, and E. Kaas. The hybrid Eulerian Lagrangian scheme tested with chemistry. *Geoscientific Model Development Discussions*, 5:3695–3732, 2012.

[76] S. Harig, Chaeroni, W.S. Pranowo, and J. Behrens. Tsunami simulations on several scales. *Ocean Dynamics*, 58:429–440, 2008.

[77] B. Haurwitz. The motion of atmospheric disturbances on the spherical earth. *Journal of Marine Research*, 3:254–267, 1940.

[78] P. Haynes. Stratospheric dynamics. *Annual Review of Fluid Mechanics*, 37:263–93, 2005.

[79] C.W. Hirt, A.A. Amsden, and J.L. Cook. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *Journal of Computational Physics*, 14:227–253, 1974.

[80] J. R. Holton. *An Introduction to Dynamic Meteorology*. Elsevier Academic Press, 4th edition, 2004.

[81] T.Y. Hou and J. Lowengrub. Convergence of the point vortex method for the 3-D Euler equations. *Communications on Pure and Applied Mathematics*, XLIII:965–981, 1990.

[82] T.Y Hou, J. Lowengrub, and R. Krasny. Convergence of a point vortex method for vortex sheets. *SIAM Journal on Numerical Analysis*, 28:308–320, 1991.

[83] W. Huang and R.D. Russell. *Adaptive Moving Mesh Methods*. Springer, 2010.

[84] Kitware Inc. *The VTK User's Guide*, 11th edition, 2010. VTK Version 5.8, `http://www.vtk.org`.

[85] C. Jablonowski, M. Herzog, J.E. Penner, R.C. Oehmke, Q.F. Stout, B. Van Leer, and K.G. Powell. Block-structured adaptive grids on the sphere: Advection experiments. *Monthly Weather Review*, 134:3691–3713, 2006.

[86] C. Jablonowski, P.H. Lauritzen, M.A. Taylor, and R.D. Nair. The dynamical core experiment: An overview of the 2008 NCAR ASP colloquium. In *13th Annual CCSM Workshop*. National Center for Atmospheric Research, 2008.

[87] C. Jablonowski, R. C. Oehmke, and Q. F. Stout. Block-structured adaptive meshes and reduced grids for atmospheric general circulation models. *Philosophical Transactions of the Royal Society A*, 367:4497–4522, 2009.

[88] C. Jablonowski and D.L. Williamson. The pros and cons of diffusion, filters, and fixers in atmospheric general circulation models. In P.H. Lauritzen, M.A. Taylor, C. Jablonowski, and R.D. Nair, editors, *Numerical Techniques for Global Atmospheric Models*, chapter 13. Springer, 2011.

[89] J.D. Jackson. *Mathematics for Quantum Mechanics*. Dover, 1962.

[90] D.W. Jacobsen, M. Gunzburger, T. Ringler, J. Burkardt, and J. Peterson. Parallel algorithm for spherical Delaunay triangulations and spherical centroidal Voronoi tesselations. *Journal of Computational Geometry*, Submitted, 2013.

[91] L. Ju, T. Ringler, and M. Gunzburger. Voronoi tesselations and their application to climate and global modeling. In P.H. Lauritzen, M.A. Taylor, C. Jablonowski, and R.D. Nair, editors, *Numerical Techniques for Global Atmospheric Models*, chapter 10. Springer, 2011.

[92] M. N. Juckes. Personal communication, 2012.

[93] M.N. Juckes and M.E. McIntyre. A high-resolution one-layer model of breaking planetary waves in the stratosphere. *Nature*, 328:590–596, 1987.

[94] L. Kagonovskiy. Adaptive panel representation for oblique collision of two vortex rings. *International Journal of Non-Linear Mechanics*, 46:9–13, 2011.

[95] J. Kent, J. Thuburn, and N. Wood. Assessing implicit large eddy simulation for two-dimensional flow. *Quarterly Journal of the Royal Meteorological Society*, 138:365–376, 2012.

[96] Y. Kimura and H. Okamoto. Vortex motion on a sphere. *Journal of the Physical Society of Japan*, 56:4203–4206, 1987.

[97] P. Koumoutsakos. Flow simulations using particles. *Annual Review of Fluid Mechanics*, 37:457–487, 2005.

[98] P. Koumoutsakos and A. Leonard. High-resolution simulations of the flow around an impulsively started cylinder using vortex methods. *Journal of Fluid Mechanics*, 296:1–38, 1995.

[99] R. Krasny. Desingularization of periodic vortex sheet roll-up. *Journal of Computational Physics*, 65:292–313, 1986.

[100] R. Krasny. A study of singularity formation in a vortex sheet by the point-vortex approximation. *Journal of Fluid Mechanics*, 167:65–93, 1986.

[101] R. Krasny. Computation of vortex sheet roll-up in the Trefftz plane. *Journal of Fluid Mechanics*, 184:123–155, 1987.

[102] R. Krasny. Vortex sheet computations: roll-ups, wakes, separation. In C.R. Anderson and C. Greengard, editors, *Proceedings of the AMS Summer Seminar in Vortex Dynamics and Vortex Methods*, Lectures in Applied Mathematics, pages 385–402. American Mathematical Society, 1991.

[103] N. Kuring. Ocean Color Web, NASA Visible Earth. `http://visibleearth.nasa.gov/`, 30-OCT-2012. sandyhalf_vir_2012304.jpg.

[104] P. H. Lauritzen, E. Kaas, and B. Machenhauer. A mass-conservative semi-implicit semi-Lagrangian limited-area shallow-water model on the sphere. *Monthly Weather Review*, 134:1205–1221, 2006.

[105] P. H. Lauritzen, W. C. Skamarock, M. J. Prather, and M. A. Taylor. A standard test case suite for two-dimensional linear transport on the sphere. *Geoscientific Model Development Discussions*, 5:189–228, 2012.

[106] P.H. Lauritzen, N. Andronova, P. Bosler, D. Calhoun, T. Enomoto, L. Dong, S. Dubey, O. Guba, A.B. Hansen, C. Jablonowski, H.-M. Juang, E. Kaas, J. Kent, R. Muller, J.E. Penner, M.J. Prather, D. Reinert, W.C. Skamarock, B. Sorensen, M.A. Taylor, P.A. Ullrich, and J. B. White III. A standard test case suite for two-dimensional linear transport on the sphere : Results from a collection of state-of-the-art schemes. *Geoscientific Model Development Discussions*, 2013. (In preparation).

[107] P.H. Lauritzen, R.D. Nair, and P.A. Ullrich. A conservative semi-Lagrangian multi-tracer transport scheme (CSLAM) on the cubed-sphere grid. *Journal of Computational Physics*, 229:1401–1424, 2010.

[108] P.H. Lauritzen, M.A. Taylor, C. Jablonowski, and R.D. Nair, editors. *Numerical Techniques for Global Atmospheric Models*. Springer, 2011.

[109] P.H. Lauritzen, P.A. Ullrich, and R.D. Nair. Atmospheric transport schemes: Desirable properties and a semi-Lagrangian view on finite-volume discretizations. In P.H. Lauritzen, M.A. Taylor, C. Jablonowski, and R.D. Nair, editors, *Numerical Techniques for Global Atmospheric Models*, chapter 8. Springer, 2011.

[110] A. Leonard. Vortex methods for flow simulation. *Journal of Computational Physics*, 37:289–335, 1980.

[111] R. J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhäuser, 1992.

[112] R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. SIAM, 2007.

[113] M.N. Levy, R.D. Nair, and H.M. Tufo. A high-order element-based Galerkin method for the barotropic vorticity equation. *International Journal for Numerical Methods in Fluids*, 59:1369–1387, 2009.

[114] J.C. Lin, D. Brunner, and C. Gerbig. Studying atmospheric transport through Lagrangian models. *EOS*, 92(21):177–184, 2011.

[115] S.J. Lin. A "vertically Lagrangian" finite-volume dynamical core for global models. *Monthly Weather Review*, 132:2293–2307, 2004.

[116] S.J. Lin and R.B. Rood. Multidimensional flux-form semi-Lagrangian transport schemes. *Monthly Weather Review*, 2046:2046–2070, 1996.

[117] S.J. Lin and R.B. Rood. An explicit flux-form semi-Lagrangian shallow-water model on the sphere. *Quarterly Journal of the Royal Meteorological Society*, 123:2477–2498, 1997.

[118] K. Lindsay and R. Krasny. A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow. *Journal of Computational Physics*, 172:879–907, 2001.

[119] E. Lorenz. Deterministic non-periodic flow. *Journal of the American Meteorological Society*, 20:130–141, 1963.

[120] A. J. Majda and A. L. Bertozzi. *Vorticity and Incompressible Flow*. Cambridge University Press, 2002.

[121] J. Makino. An efficient parallel algorithm for $O(N^2)$ direct summation method and its variations on distributed-memory parallel machines. *New Astronomy*, 7:373–384, 2002.

[122] R. Malek-Madani. *Advanced Engineering Mathematics with Mathematica and MATLAB*. Addison-Wesley-Longman, 1998.

[123] R. Malek-Madani. *Physical Oceanography*. CRC Press, 2012.

[124] G.L. Manney, K. Kruger, J.L. Sabutis, and S.A. Sena. The remarkable 2003-2004 winter and other recent warm winters in the Arctic stratosphere since the late 1990s. *Journal of Geophysical Research*, 110:D04107, 2005.

[125] L.G. Margolin. An introduction to 'An arbitrary Lagrangian-Eulerian computing method for all flow speeds'. *Journal of Computational Physics*, 135:198–202, 1997.

[126] MATLAB. Version 7.12 (R2011a). The MathWorks Inc., 2011. `http://www.mathworks.com`.

[127] D. McCormack. *Scientific Software Development with Fortran*. `lulu.com`, 2009.

[128] M.E. McIntyre and T.N. Palmer. The 'surf zone' in the stratosphere. *Journal of Atmospheric and Terrestrial Physics*, 48:825–849, 1984.

[129] J.L. Mead. The shallow water equations in Lagrangian coordinates. *Journal of Computational Physics*, 200:654–669, 2004.

[130] F. Mesinger. Numerical integration of the primitive equations with a floating set of computation points: Experiments with a global barotropic model. *Monthly Weather Review*, 99:15–29, 1971.

[131] R.D. Nair and P.H. Lauritzen. A class of deformational flow test cases for linear transport problems on the sphere. *Journal of Computational Physics*, 229:8868 – 8887, 2010.

[132] R.D. Nair, J.S. Scroggs, and F.H.M. Semazzi. A forward-trajectory global semi-Lagrangian transport scheme. *Journal of Computational Physics*, 193:275–294, 2003.

[133] F.J. Narcowich, J.D. Ward, and G. B. Wright. Divergence-free RBFs on surfaces. *Journal of Fourier Analysis and Applications*, 13:643–663, 2007.

[134] F. Nebeker. *Calculating the Weather: Meteorology in the 20th Century*. Academic Press, 1995.

[135] P. Newman and H. Riebeek. NASA Earth Observatory. `http://earthobservatory.nasa.gov/IOTD/view.php?id=36972`, Jan-Feb 2009. Data provided by the Goddard Modeling and Assimilation Office.

[136] P. K. Newton. *The N-Vortex Problem.* Springer, 2001.

[137] P. K. Newton and H. Shokraneh. The N-vortex problem on a rotating sphere. I. Multi-frequency configurations. *Proceedings of the Royal Society A*, 462:149–169, 2006.

[138] P.K. Newton and T. Sakajo. The N-vortex problem on a rotating sphere. III. Ring configurations coupled to a background field. *Proceedings of the Royal Society A*, 463:961–977, 2007.

[139] M. Nitsche and J.H. Strickland. Extension of the gridless vortex method into the compressible flow regime. *Journal of Turbulence*, 3:7, 2002.

[140] A. Okabe, B. Boots, K. Sugihara, and S. Nok Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Ltd., 2nd edition, 2000.

[141] Open MPI. Versions 1.4.2 and 1.6.2. `http://www.open-mpi.org/`, 2012.

[142] M. Perlman. On the accuracy of vortex methods. *Journal of Compuational Physics*, 59:200–223, 1985.

[143] R.D. Pingree and B. Le Cann. Three anticyclonic Slope Water Oceanic eDDIES (SWODDIES) in the southern Bay of Biscay in 1990. *Deep Sea Research, A*, 39:1147–1175, 1992.

[144] P. Poncet. Finite difference stencils based on particle strength exchange schemes for improvement of vortex methods. *Journal of Turbulence*, 7:24, 2006.

[145] P.A. Raviart. Particle approximation of first order systems. *Journal of Computational Mathematics*, 4:50–61, 1986.

[146] R. Renka. Algorithm 772: STRIPACK : Delaunay triangulation and Voronoi diagram on the surface of a sphere. *ACM Transactions on Mathematical Software*, 23:416–434, 1997.

[147] R. Renka. Algorithm 773: SSRFPACK: Interpolation of scattered data on the surface of a sphere with a surface under tension. *ACM Transactions on Mathematical Software*, 23:435–442, 1997.

[148] R.J. Renka. Interpolatory tension splines with automatic selection of tension factors. *SIAM Journal on Scientific and Statistical Computing*, 8:393–415, 1987.

[149] L.F. Richardson. *Weather Prediction by Numerical Process*. Cambridge University Press, 1922.

[150] T.D. Ringler. Momentum, vorticity, and transport : Considerations in the design of a finite-volume dynamical core. In P.H. Lauritzen, M.A. Taylor, C. Jablonowski, and R.D. Nair, editors, *Numerical Techniques for Global Atmospheric Models*, chapter 7. Springer, 2011.

[151] T.D. Ringler, L. Ju, and M. Gunzburger. A multiresolution method for climate system modeling: Application of spherical centroidal Voronoi tesselations. *Ocean Dynamics*, 58:475–498, 2008.

[152] T.D. Ringler and D.A. Randall. A potential enstrophy and energy conserving numerical scheme for solution of the shallow-water equations on a geodesic grid. *Monthly Weather Review*, 130:1397–1410, 2002.

[153] C. Ronchi, R. Iacono, and P.S. Paolucci. The 'cubed sphere': A new method for the solution of partial differential equations in spherical geometry. *Journal of Computational Physics*, 124:93–114, 1996.

[154] L. Rosenhead. The formation of vortices from a surface of discontinuity. *Proceedings of the Royal Society A*, 134:170–192, 1932.

[155] L.F. Rossi, J.F. Lingevitch, and A.J. Bernoff. Quasi-steady monopole and tripole attractors for relaxing vortices. *Physics of Fluids*, 9:2329–2338, 1997.

[156] D. Rossinelli and P. Koumoutsakos. Vortex methods for incompressible flow simulations on the GPU. *Visual Computing*, 24:699–708, 2008.

[157] R. Sadourny. Conservative finite-difference approximations of the primitive equations on quasi-uniform spherical grids. *Monthly Weather Review*, 100:136–144, 1972.

[158] T. Sakajo. An extension of Draghicescu's fast tree-code algorithm to the vortex method on a sphere. *Journal of Computational and Applied Mathematics*, 225:158–171, 2009.

[159] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2006.

[160] J. Schmaltz. NASA Visible Earth. `http://visibleearth.nasa.gov/`, 23-OCT-2012. sandy_tmo_2012297.jpg.

[161] A. Simmons, M. Hortal, G. Kelly, A. McNally, A. Untch, and S. Uppala. ECMWF analyses and forecasts of stratospheric winter polar vortex breakup: September 2002 in the southern hemisphere and related events. *Journal of the Atmospheric Sciences*, 62:668–689, 2005.

[162] W.C. Skamarock, J.B. Klemp, M.G. Duda, L.D. Fowler, S.-H. Park, and T.D. Ringler. A multi-scale nonhydrostatic atmospheric model using centroidal Voronoi tesselations and C-grid staggering. *Monthly Weather Review*, 140:3090–3105, 2012.

[163] R. K. Smith and D. G. Dritschel. Revisiting the Rossby-Haurwitz wave test case with contour advection. *Journal of Computational Physics*, 217:473–484, 2006.

[164] C. Snyder, D. Muraki, R. Plougonven, and F. Zhang. Inertia-gravity waves generated within a dipole vortex. *Journal of the Atmospheric Sciences*, 64:4417–4431, 2007.

[165] I. Stakgold. *Green's Functions and Boundary Value Problems*. John Wiley & Sons, 2nd edition, 1998.

[166] A. Staniforth and J. Côté. Semi-Lagrangian integration schemes for atmospheric modeling : A review. *Monthly Weather Review*, 119:2206–2223, 1991.

[167] A. Staniforth and J. Thuburn. Horizontal grids for global weather and climate prediction models: A review. *Quarterly Journal of the Royal Meteorological Society*, 138:1–26, 2012.

[168] P. N. Swarztrauber. The approximation of vector functions and their derivatives on the sphere. *SIAM Journal on Numerical Analysis*, 18:191–210, 1981.

[169] P.N. Swarztrauber. The vector harmonic transform method for solving partial differential equations in spherical geometry. *Monthly Weather Review*, 121:3415–3437, 1993.

[170] Global Ocean Observing System. International Argo Program. `http://www.argo.ucsd.edu` and `http://argo.jcommops.org`.

[171] T. Takemura, H. Nakamura, M. Takigawa, H. Kondo, T. Satomura, T. Miyasaka, and T. Nakajima. A numerical simulation of global transport of particles emittted from the Fukushima Daiichi nuclear power plant. *SOLA*, 7:101–104, 2011.

[172] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.1 edition, 2012. http://www.cgal.org/Manual/4.1/doc_html/cgal_manual/packages.html.

[173] M.C. Thompson and J.H. Ferziger. An adaptive multigrid technique for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 82:94–121, 1989.

[174] A.J. Thorpe, H. Volkert, and M.J. Ziemiannński. The Bjerknes' circulation theorem: A historical perspective. *Bulletin of the American Meteorological Society*, 84:471–479, 2003.

[175] J. Thuburn. Some conservation issues for the dynamical cores of NWP and climate models. *Journal of Computational Physics*, 227:3715–3730, 2008.

[176] J. Thuburn. Conservation in dynamical cores: What, how, and why? In P.H. Lauritzen, M.A. Taylor, C. Jablonowski, and R.D. Nair, editors, *Numerical Techniques for Global Atmospheric Models*, chapter 11. Springer, 2011.

[177] J. Thuburn. Some basic dynamics relevant to the design of atmospheric model dynamical cores. In P.H. Lauritzen, M.A. Taylor, C. Jablonowski, and R.D. Nair, editors, *Numerical Techniques for Global Atmospheric Models*, chapter 1. Springer, 2011.

[178] J. Thuburn, T.D. Ringler, W.C. Skamarock, and J.B. Klemp. Numerical representation of geostrophic modes on arbitrarily structured C-grids. *Journal of Computational Physics*, 228:8321–8335, 2009.

[179] L.N. Trefethen and D. Bau III. *Numerical Linear Algebra*. SIAM, 1997.

[180] UCAR/NCAR/CISL/VETS. The NCAR Command Language (Version 6.0.0) [Software], 2012. `http://dx.doi.org/10.5065/D6WD3XH5`.

[181] P.A. Ullrich, C. Jablonowski, J. Kent, P.H. Lauritzen, R.D. Nair, and M.A. Taylor. Test case document. In *Dynamical Core Model Intercomparison Project (DCMIP)*, 2012. `http://www.earthsystemcog.org/projects/dcmip-2012/`.

[182] P. Vachal, R.V. Garimella, and M.J. Shashkov. Untangling of 2D meshes in ALE simulations. *Journal of Computational Physics*, pages 627–644, 2004.

[183] G. K. Vallis. *Atmospheric and Oceanic Fluid Dynamics*. Cambridge University Press, 2006.

[184] L. Wang. *Radial basis functions and vortex methods and their application to vortex dynamics on a rotating sphere*. PhD thesis, The University of Michigan, 2010.

[185] H.F. Weinberger. *A First Course in Partial Differential Equations with Complex Variables and Transform Methods*. Dover, 1995.

[186] A.A. White, B.J. Hoskins, I. Roulstone, and A. Staniforth. Consistent approximate models of the global atmosphere: Shallow, deep, hydrostatic, quasi-hydrostatic, and non-hydrostatic. *Quarterly Journal of the Royal Meteorological Society*, 131:2081–2107, 2005.

[187] D. L. Williamson, J. B. Drake, J. J. Hack, R. Jakob, and P. N. Swartztrauber. A standard test set for numerical approximations to the shallow water equations in spherical geometries. *Journal of Computational Physics*, 102:211–224, 1992.

[188] D.L. Williamson. The evolution of dynamical cores for global atmospheric models. *Journal of the Meteorological Society of Japan*, 85B:241–269, 2007.

[189] G. Wright. A radial basis function partition of unity method for divergence-free approximation of vector fields on the sphere. In *SIAM Annual Meeting*. SIAM, July 2012.

[190] A. H. Zemanian. *Distribution Theory and Transform Analysis*. Dover, 1965.

[191] D. Zwillinger. *CRC Standard Mathematical Tables and Formulae*. Chapman & Hall/CRC Press, 31st edition, 2003.