2009

Integration of Higher-Order Physics in the Community Ice Sheet Model: Scientific and Software Concerns

Timothy Joseph Bocek
The University of Montana

INTEGRATION OF HIGHER-ORDER PHYSICS IN THE COMMUNITY ICE SHEET MODEL: SCIENTIFIC AND SOFTWARE CONCERNS

By

Timothy J. Bocek

Bachelor of Science, Computer Science, Washington State University, Pullman, WA, 2007

Thesis

presented in partial fulfillment of the requirements
for the degree of

Master of Science
in Computer Science

The University of Montana
Missoula, MT

Summer 2009

Approved by:

Dr. Perry Brown, Dean
Graduate School

Dr. Jesse Johnson, Chair
Computer Science

Dr. Joel Henry
Computer Science

Dr. Emily Stone
Mathematical Sciences

Bocek, Timothy J., M.S., August 2009 Computer Science

Integration of Higher-Order Physics in the Community Ice Sheet Model: Scientific and Software Concerns

Chairperson: Dr. Jesse Johnson

The Community Ice Sheet Model (CISM) is a next-generation land ice model that is designed to answer important questions regarding the response of Earth's land ice to climate forcing. The program extends Glimmer, an ice sheet model based on the shallow ice approximation. This thesis concerns a project to ready CISM for these questions by integrating an ice velocity diagnostic based on a first-order approximation of the Navier-Stokes equations. I present in detail the derivation of the first-order momentum balance equations for both the interior of an ice sheet and a variety of boundary conditions. I discuss the numerical techniques used to build and solve a finite difference approximation of these equations, as well as the software engineering process and design solutions used to integrate this model with the rest of CISM. I then build a case for the correctness of the integrated model by presenting the results of numerous experiments that compare this model to data, exact solutions, and a collection of similar models. I find that in most cases the integrated model performs favorably in model intercomparisons and in comparisons to exact solutions. I finally present possible future directions for the CISM project with respect to higher-order ice modeling, as well as lessons learned for future maintainers.

TABLE OF CONTENTS

ABSTRACT ii

CHAPTER 1 INTRODUCTION 1

1.1 Motivation for ice sheet modeling 1

1.2 A Taxonomy of Ice Sheet Models 2

1.3 Introducing CISM 4

1.4 Thesis Organization 5

CHAPTER 2 THE FIRST ORDER MOMENTUM BALANCE FOR ICE SHEET MODELS 7

2.1 Momentum Balance in the Ice Sheet Interior 7

2.1.1 Conservation Equations 7

2.1.2 Deviatoric Stresses 10

2.1.3 The Constitutive Relationship 12

2.1.4 Stress and Velocity 15

2.2 Boundary Conditions 16

2.2.1 Stress-Free Surface 17

2.2.2 Basal Boundary Conditions 18

2.2.3 Stress-Free Base Condition 22

2.2.4 Ice Shelf Lateral Boundary Condition 23

CHAPTER 3 NUMERICAL APPROXIMATION 27
3.1 Rescaled Vertical Coordinate 27
3.1.1 Derivation of Rescaling Parameters 29
3.1.2 Transformation of Equations 30
3.2 Finite Difference Approximation 32
3.3 Solving the Linear System 37
3.4 Nonlinear Iteration 39
CHAPTER 4 SOFTWARE INTEGRATION 41
4.1 Code Structure 43
4.2 Engineering Process 49
4.2.1 Preparation of standalone code 49
4.2.2 Initial diagnostic integration 50
4.2.3 Physics and numerics refinement 51
4.2.4 Software design refinement 51
4.2.5 Initial prognostic integration 52
4.3 Testing Process 52
CHAPTER 5 MODEL VERIFICATION 55
5.1 ISMIP-HOM Experiments 56
5.1.1 Description of Experiments 56
5.1.2 Results 60
5.1.3 ISMIP-HOM D 65
5.1.4 Effects of grid selection 66
5.2 Idealized ice shelf experiments 67
5.2.1 Constant thickness 68
5.2.2 Van der Veen Ice Tongue 69

5.3 Ross ice shelf experiment 72
5.3.1 Description of Experiment 72
5.3.2 Results 73
CHAPTER 6 CONCLUSION 77
6.1 Lessons Learned 77
6.2 Towards a true community model 79
APPENDIX A GLIDE CONFIGURATION WITH HIGHER-ORDER OPTIONS 82
A.1 Introduction 82
A.2 Documentation 82
APPENDIX B GLIDE NETCDF VARIABLES 91
B.1 Introduction 91
B.2 Documentation 91
APPENDIX C SPARSE MATRIX DOCUMENTATION 96
C.1 User documentation 96
C.1.1 Modules 96
C.1.2 Setting up a linear system 96
C.1.3 Solving a linear system 98
C.1.4 Handling errors 101
C.2 Developer documentation 102
BIBLIOGRAPHY 102

LIST OF TABLES

Table 2.1 Physical constants in first-order ice sheet model 9

Table 5.1 ISMIP-HOM A Refinement Results 67

Table 5.2 ISMIP-HOM C Refinement Results 68

Table 5.3 Constants for ice tongue experiment 70

Table 5.4 ROSS error intercomparison 76

LIST OF FIGURES

Figure 3.1 Sigma coordinate system 28

Figure 3.2 Mask field 35

Figure 3.3 Numeric discretization at shelf front 36

Figure 4.1 Illustration of staggered and nonstaggered grids. 42

Figure 4.2 Conceptual static code structure 45

Figure 4.3 Implemented static code structure 47

Figure 5.1 Results for ISMIP-HOM A 61

Figure 5.2 Results for ISMIP-HOM B 63

Figure 5.3 Results for ISMIP-HOM C 64

Figure 5.4 Van der Veen ice tongue velocity comparison 71

Figure 5.5 Map of velocities computed for EISMINT-Ross 74

Figure 5.6 EISMINT-Ross RIGGS station comparisons 75

CHAPTER 1 INTRODUCTION

1.1 Motivation for Ice Sheet Modeling

Within the last several years, the causes and effects of climate change have become a central issue. Investigating this issue poses two broad questions: what climate change scenarios are likely given different levels of anthropogenic climate forcing, and given a scenario, what will be the ecological and economic impact? Ten percent of the world's population lives less than one meter above sea level, and even a relatively small change in sea level could have very destructive effects. Therefore, central to the latter question is the issue of sea level rise.

It is thought that the change in sea level over the next century will be caused by two factors: melting ice, and thermal expansion of the oceans [Watson, 2001]. Although thermal expansion will likely contribute more change than melting ice, it is a relatively well-understood process and there is little uncertainty [Watson, 2001]. Additionally, though the melting of alpine glaciers has potential to contribute more change than the melting of major ice sheets, the problem is not as heavily studied as it is thought that these will be fully melted by end of the century [Watson, 2001]. I therefore turn my attention in this thesis to the issue of melting of the Earth's major ice sheets in Greenland and Antarctica.

The Antarctic and Greenland ice sheets, if fully melted, would contribute enough

water to Earth's oceans to raise the sea level 64 m [Lythe and Vaughan, 2001] [Bamber et al., 2001] [Watson, 2007]. On the other hand, the influence of glaciers and ice sheets on sea level rise from 1993 to 2003 is estimated at $1.2 \pm 0.4mm/a$ [Watson, 2007]. Naively assuming that these trends remain constant during the next century, we will see at most 16 cm of sea level change. Clearly, neither of these are realistic scenarios. More informed estimates predict that melting ice during the next century could contribute as much as 34 cm to sea level rise, but could also reduce the sea level by up to 9 cm [Watson, 2001]. Improving upon these estimates requires an improved understanding of how the Earth's glaciers, including the Greenland and Antarctic ice sheets, will respond to climate change. The ability to model these systems is therefore an important effort towards the overall goal of understanding the impacts of climate change. Often, these models take the form of computer simulations based on fluid dynamics.

1.2 A Taxonomy of Ice Sheet Models

To fully model the motion of ice, one must solve a number of coupled equations. Ignoring concerns such as temperature advection, there are two such equations. The diagnostic equation specifies a field of velocities within the ice sheet, and the prognostic or transport equation uses the computed velocities to specify how the thickness of the ice sheet changes over time. Because thickness gradients in part determine velocity, this leads to a coupled situation; while time-stepping the thickness equations, one must recompute the velocities at regular intervals. This thesis will focus mainly on the diagnostic equation.

At some level, nearly all ice sheet diagnostic models treat ice as an incompressible non-Newtonian fluid and begin by modeling the balance of stresses that arises from

the Stokes equations for incompressible flow [Hooke, 1998]. Models mainly differ in how many of these stresses are included in the model and how many are neglected in order to simplify the computation. A model that includes everything is referred to as a full-Stokes model. These are the "holy grail" of ice sheet models, and can currently be deployed for small-scale work but are still impractical on continental scales due to their computational intensity. Therefore, most models apply simplifying assumptions to reduce computational complexity.

The most basic simplification arises from the assumption that all ice flow is due to vertical deformation in response to the gravitational driving stress. This assumption is valid as long as the ice is very thin in comparison to its extent; it is therefore referred to as the Shallow Ice Approximation (SIA) [Hutter, 1983]. This is a popular model due to its computational simplicity: ice physics are based only on local geometry gradients [Hutter, 1983], and therefore the influence of the ice sheet on a given point is limited to the immediate vicinity. Additionally, the SIA assumes that the ice is "glued" to the bed - it does not its self take into account the fact that ice can slide over a lubricated bed [Bueler and Brown, 2009], but this shortcoming is often ameliorated by computing basal velocities in a separate, simple model known as a sliding law and adding them to the SIA-derived velocities (e.g. [Greve, 1997], [Huybrechts, 1999]).

While valid for large portions of the Antarctica and Greenland ice sheets, the SIA fails in several critical regions: namely, streams of fast-moving ice as well as ice shelves, or regions of floating ice. In these cases, changes in the ice sheet's geometry and velocity can occur on relatively short timescales, which the shallow ice approximation fails to capture [Watson, 2007]. Remote sensing data have confirmed that these short time scale effects are observed, thereby further calling the suitability of models based on the SIA into question [Oppenheimer and Alley, 2007].

The Shallow Shelf Approximation (SSA) ameliorates these concerns for ice shelf

regions by modeling vertically averaged ice motion instead the local balance of gravitational driving stress [Morland, 1987] [Weis et al., 1999]. Unlike the SIA, the SSA models longitudinal stresses, or the ability for ice to push and pull upstream and downstream ice. This leads to a critical computational difference: the physics in SSA models are based on *non-local* effects. While solving for velocities in a SIA model is little more than a matter of numeric differentiation, solving for velocities in a model that applies the SSA or a higher approximation requires solving a partial differential equation. The addition of longitudinal stresses allows the SSA to capture salient physics in ice shelf and ice stream regions. Because of the vertically averaged nature, however, the SSA does not perform well in areas where there is little velocity at the base of the ice sheet, and it is these areas where the SIA performs well. Some models have treated grounded ice using the SIA and floating ice using the SSA, though complexity arises in these models when treating the grounding line and other boundaries between SIA-governed flow and SSA-governed flow Hindmarsh [2006]. More modern attempts construct a hybrid SIA-SSA model by using the latter as a more realistic sliding law for the former [Pollard and DeConto, 2007], [Bueler and Brown, 2009]. On the whole, however, it is difficult to treat an entire system holistically using only shallow approximations.

Modern computational prowess has advanced to the point where it is feasible to discard these simplifications and treat a sheet-shelf system holistically using a single model. Though full Stokes is still out of reach, it is possible to apply fewer simplifying assumptions and build a higher-order model. These higher-order models differ from the SIA by including a treatment of longitudinal stresses, and differ from both of the shallow approximations by incorporating lateral drag as well [Hindmarsh, 2004]. Higher-order models differ from full-Stokes models by excluding the vertical shear stresses on horizontal planes [Pattyn, 2003]; this will be discussed in much more

detail in Chapter 2.

Higher-order stress treatments can be either vertically averaged or fully three-dimensional [Hindmarsh, 2004]. Three-dimensional models require more time to compute but do not require an artificial recovery of the third dimension in order to compute temperature advection within the ice sheet. In a numerical comparison of ice sheet solutions using perturbative methods, Hindmarsh [2004] found three-dimensional methods to be vastly superior to SIA models and slightly better than vertically averaged (SSA) models. However, though the vertically explicit treatment does only slightly better for regions where the SSA is valid, it is able to compute velocities in regions where the SSA cannot due to a low velocity at the base of the ice sheet. Therefore, unlike the SSA, higher-order approximations are able to treat an entire sheet-stream-shelf system holistically.

1.3 Introducing CISM

A problem currently facing the global community of ice sheet modelers is fragmentation of community software. It has been suggested [Oppenheimer and Alley, 2007] that a "community ice sheet model", providing a framework consisting of existing implementations of most model components as well as facilities to couple to larger climate models, would be beneficial to the community. This would provide a starting point for researchers interested in improving models, and would represent an open, off-the-shelf software component for researchers investigating the impact of climate change. This "community model" approach has been used before, as in NCAR's Community Climate System Model (CCSM) [Drake et al., 2005], to which the community ice sheet model will contribute land ice physics.

Currently, Glimmer is the primary candidate for becoming this community ice

sheet model. As described by Rutt et al. [2009], Glimmer is a ice model based on the shallow-ice approximation that includes thermomechanical coupling and sliding laws. The strengths of this model is that it is includes facilities for coupling with global climate models, is supported by an active development community that focuses both on scientific and software engineering concerns, and is already a strong implementation of the SIA from which to begin work on extensions.

As we have seen, however, the set of physics currently implemented in Glimmer have proven inadequate. In order to be useful as a next-generation ice sheet model, Glimmer must be able to make predictions based on higher-order physical approximations. A development effort to this end is being headed by the University of Montana and Los Alamos National Laboratories (LANL) To reflect both the new model's role in response to the recommendation above as well as its role as a component in CCSM, the new version of the model has been called the Community Ice Sheet Model (CISM). More recently, out of respect for the model's history and pedigree, the full name of the model has been chosen to be Glimmer/CISM. When referring to the model in this thesis, I will use "Glimmer" to refer to the original version of the model that we have been extending, and "CISM" to refer to the version of the model that is a result of this development.

1.4 Thesis Organization

The rest of this thesis is organized as follows:

- **Chapter 2** provides a detailed derivation of the first order diagnostic model implemented, including a discussion of boundary conditions.
- **Chapter 3** discusses the methods used to solve the equations derived in Chapter

2, including the finite difference discretization and the numerical methods for solution.

- **Chapter 4** presents a software engineering case study of integrating the first order model with CISM.

- **Chapter 5** describes a number of simple experiments to verify the correctness of the first order model implementation using both exact solutions and intercomparison projects.

- **Chapter 6** contains concluding remarks and suggests directions for future work.

CHAPTER 2 THE FIRST ORDER MOMENTUM BALANCE FOR ICE SHEET MODELS

I have integrated into CISM an improved version of Frank Pattyn's first order model [Pattyn, 2003]. This model simplifies the full Stokes equations by neglecting vertical resistive stresses [Pattyn, 2003]. This arises from the simplifying assumption that the pressure at any point in the ice is due only to the weight of the ice above it and not due to resistance to motion; this is also known as the cryostatic approximation. As we will see, this assumption has a major computational advantage. A proper full-Stokes model solves for four three-dimensional fields: three components of velocity and a scalar pressure [Versteeg and Malalasekera, 1995]. By assuming the hydrostatic approximation, we no longer need to solve for pressure, and can reconstruct the vertical velocity field using the fact that ice is incompressible [Pattyn, 2003]. This reduces the complexity of the computation considerably, reducing the number of variables from four to two: namely, the 3D horizontal fields of velocity.

2.1 Momentum Balance in the Ice Sheet Interior

2.1.1 Conservation Equations

To derive the first-order model, I begin by stating the laws of conservation of mass, momentum, and energy as they apply to an incompressible fluid [Versteeg and Malalasekera, 1995]. Consider an infinitely small "control volume" of ice. The law

of conservation of mass in this context states that the rate of increase of the mass of the control volume must equal the net rate of flow into the control volume [Versteeg and Malalasekera, 1995]. With an incompressible fluid such as ice, the mass of a control volume cannot change, so this law reduces to the fact that any flow into the control volume must be balanced by flow out of the control volume [Versteeg and Malalasekera, 1995]. Stated formally,

$$\nabla \cdot \mathbf{v} = 0 \tag{2.1}$$

where $\mathbf{v}$ is the velocity vector.

Conservation of momentum is a statement of Newton's second law as applied to fluid dynamics: the rate of change of momentum equals the sum the forces [Versteeg and Malalasekera, 1995]. If one considers a control volume of ice, the rate of change of momentum (or the density times the acceleration) equals the sum of the forces due to stress on the control volume and the force of gravity [Pattyn, 2003]. Formally,

$$\rho_i \frac{d\mathbf{v}}{dt} = \nabla \cdot \mathbf{T} + \rho_i \mathbf{g} \tag{2.2}$$

where $\mathbf{T}$ is the total stress tensor, and ρ_i and g are the density of ice and gravitational acceleration respectively. The value of these and other physical constants are given in Table 2.1. Here I can neglect the non-gravitational acceleration term due the Froude number of ice being of the order 10^{-12} (the acceleration term is 10^{12} times smaller than the gravitational terms).

$$0 = \nabla \cdot \mathbf{T} + \rho_i \mathbf{g} \tag{2.3}$$

The conservation of energy is a statement of the first law of thermodynamics: the

Symbol	Meaning	Value
g	Acceleration due to gravity	$9.81\ m\ s^{-2}$
ρ_i	Density of ice	$910\ kg\ m^{-3}$
ρ_w	Density of ocean water	$1028\ kg\ m^{-3}$
n	Exponent in Glen's flow law	3
k_i	Thermal conductivity of ice	$6.62 \times 10^{-7}\ J\ m^{-1}\ K^{-1}\ yr^{-1}$
c_p	Heat capacity of ice	$2009J\ kg^{-1}\ K^{-1}$

Table 2.1 Physical constants in first-order ice sheet model

rate of change of energy is equal to the rate of heat addition plus the rate of work done [Versteeg and Malalasekera, 1995]. Formally,

$$\rho_i \frac{d(c_p\theta)}{dt} = \nabla(k_i \nabla\theta) + \Phi. \tag{2.4}$$

Here θ is the ice temperature, k_i and c_p are the thermal conductivity and heat capacity of ice, and Φ is the rate of heating as a result of internal deformation. This equation is used to described the flow of temperature through an ice sheet, which in turn affects the hardness of the ice and therefore its velocity. In general, I would need to develop this equation alongside the equations for conservation of mass and momentum. Although Pattyn's original model *did* include temperature as a dependant variable, temperature advection in CISM is handled by a different software component [Rutt et al., 2009]. I will therefore turn my attention away from solving for the temperature field and instead focus on solving for the velocities within the ice.

I now begin to develop the first-order diagnostic model by applying our simplifying assumptions. Because the model was originally developed by Pattyn [2003], I will follow his derivation throughout, changing notation and expanding discussion as needed. I expand the conservation of mass, using the common notation of referring

to velocity components as u, v, and w:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{2.5}$$

I define the coordinate system so that the top of the ice sheet is at $z = 0$, with positive numbers denoting lower elevations. This restricts gravitational influence to the z dimension, and allows me to expand the momentum equation as

$$\begin{aligned} \frac{\partial T_{xx}}{\partial x} + \frac{\partial T_{xy}}{\partial y} + \frac{\partial T_{xz}}{\partial z} &= 0 \\ \frac{\partial T_{yx}}{\partial x} + \frac{\partial T_{yy}}{\partial y} + \frac{\partial T_{yz}}{\partial z} &= 0 \\ \frac{\partial T_{zx}}{\partial x} + \frac{\partial T_{zy}}{\partial y} + \frac{\partial T_{zz}}{\partial z} &= \rho_i g. \end{aligned} \tag{2.6}$$

Because vertical resistive stresses are neglected, I drop the T_{zx} and T_{zy} derivatives in the third equation

$$\frac{\partial T_{zz}}{\partial z} = \rho_i g. \tag{2.7}$$

A proper solution to this system of equations would give a velocity vector field and a stress tensor field. In order to do that, however, I would need to solve for ten variables (neglecting T_{zx} and T_{zy}) using a system of four equations. To say that this is grossly underdetermined would be an understatement! My strategy, then, is to relate stress and velocity in such a way that I need only solve for the three velocity components.

2.1.2 Deviatoric Stresses

Some additional transformations, however, are needed before directly relating the stresses and velocities. First, it has been established that the strain rates in ice do not depend on cryostatic pressure but on the differences between stresses in points on the ice sheet of similar depth [Hooke, 1998]. I can estimate the cryostatic pressure as the mean of the normal stresses

$$P = \frac{1}{3}(T_{xx} + T_{yy} + T_{zz}). \tag{2.8}$$

Thus, I define the deviatoric stress tensor as a stress tensor that neglects the stresses arising from cryostatic pressure

$$T'_{ij} = T_{ij} - \frac{1}{3}\delta_{ij}(T_{xx} + T_{yy} + T_{zz}). \tag{2.9}$$

Here, $i, j \in x, y, z$. The Kroneker Delta, δ_{ij}, is one if $i = j$, and zero otherwise.

In order to substitute deviatoric stresses for total stresses, I form the system of equations

$$\begin{aligned} T_{xx} &= T'_{xx} + \frac{1}{3}(T_{xx} + T_{yy} + T_{zz}) \\ T_{yy} &= T'_{yy} + \frac{1}{3}(T_{xx} + T_{yy} + T_{zz}). \end{aligned} \tag{2.10}$$

Solving the linear system for T_{xx} and T_{yy} gives

$$T_{xx} = 2T'_{xx} + T'_{yy} + T_{zz} \tag{2.11}$$
$$T_{yy} = 2T'_{yy} + T'_{xx} + T_{zz}.$$

I can remove the T_{zz} term by vertically integrating $\frac{\partial T_{zz}}{\partial z} = \rho_i g$ from the surface z_s to a height z in the ice sheet, resulting in

$$T_{zz}(z) = -\rho_i g(z_s - z). \tag{2.12}$$

Thus:

$$T_{xx} = 2T'_{xx} + T'_{yy} - \rho_i g(z_s - z) \tag{2.13}$$
$$T_{yy} = 2T'_{yy} + T'_{xx} - \rho_i g(z_s - z).$$

2.1.3 The Constitutive Relationship

I relate the deviatoric stress tensor to velocity gradients by equating each with the time derivative of strain, a measure of ice deformation. The deviatoric stress $\mathbf{T}'$ and strain rate $\dot{\epsilon}$ are related nonlinearly by a Glen-type flow law [Paterson, 1994]

$$T'_{ij} = 2\mu\dot{\epsilon}_{ij} \tag{2.14}$$

with the effective viscosity μ given by

$$\mu = \frac{A^{\frac{-1}{n}}}{2}\dot{\epsilon}^{\frac{1-n}{n}}. \tag{2.15}$$

Here n is the flow law exponent, and defines the strength of the nonlinearity between stress and strain rate, usually taken to be 3. A is a thermomechanical coupling parameter, usually given by an Arrhenius relationship [Pattyn, 2003]. This is the ice hardness parameter that, as mentioned earlier, is computed using equations derived from the conservation of energy. Again, it is not elaborated in this model because its computation is the responsibility of other CISM modules [Rutt et al., 2009]. For many basic experiments it is taken as a constant 10^{-16} (as in Pattyn et al. [2008]) or 10^{-18} (as in MacAyeal et al. [1996]). Finally, $\dot{\epsilon}$ is the second invariant of the strain rate tensor [Pattyn, 2003] [Hooke, 1998], given by

$$\dot{\epsilon} = \sqrt{\dot{\epsilon}_{xy}^2 + \dot{\epsilon}_{yz}^2 + \dot{\epsilon}_{zx}^2 - \dot{\epsilon}_{xx}\dot{\epsilon}_{yy} - \dot{\epsilon}_{yy}\dot{\epsilon}_{zz} - \dot{\epsilon}_{zz}\dot{\epsilon}_{xx}} + \epsilon_0 \tag{2.16}$$

.

ϵ_0 is a small regularization (currently 10^{-30}) that is added in practice to $\dot{\epsilon}$ in order to avoid a division by zero, particularly in the case of a frozen bed [Pattyn, 2003].

I now formulate the viscosity term μ in terms of velocities rather than strain rates. In a full Stokes model, the relationship between strain rates and velocity is defined as [Hooke, 1998]

$$\begin{pmatrix} \dot{\epsilon}_{xx} & \dot{\epsilon}_{xy} & \dot{\epsilon}_{xz} \\ \dot{\epsilon}_{yx} & \dot{\epsilon}_{yy} & \dot{\epsilon}_{yz} \\ \dot{\epsilon}_{zx} & \dot{\epsilon}_{zy} & \dot{\epsilon}_{zz} \end{pmatrix} = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{1}{2}(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) & \frac{1}{2}(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}) \\ \frac{1}{2}(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}) & \frac{\partial v}{\partial y} & \frac{1}{2}(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}) \\ \frac{1}{2}(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}) & \frac{1}{2}(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}) & \frac{\partial w}{\partial z} \end{pmatrix}. \tag{2.17}$$

A second simplifying assumption of the first order model, left out until now for clarity, is that I can neglect the horizontal gradients of the vertical velocity as they are much smaller than the vertical gradients of the horizontal velocity. I apply this assumption to arrive at

$$\begin{pmatrix} \dot{\epsilon}_{xx} & \dot{\epsilon}_{xy} & \dot{\epsilon}_{xz} \\ \dot{\epsilon}_{yx} & \dot{\epsilon}_{yy} & \dot{\epsilon}_{yz} \\ \dot{\epsilon}_{zx} & \dot{\epsilon}_{zy} & \dot{\epsilon}_{zz} \end{pmatrix} = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{1}{2}(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) & \frac{1}{2}\frac{\partial u}{\partial z} \\ \frac{1}{2}(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}) & \frac{\partial v}{\partial y} & \frac{1}{2}\frac{\partial v}{\partial z} \\ \frac{1}{2}\frac{\partial u}{\partial z} & \frac{1}{2}\frac{\partial v}{\partial z} & \frac{\partial w}{\partial z} \end{pmatrix}. \tag{2.18}$$

This simplifying assumption has allowed us to neglect the vertical component of velocity when solving for the velocity fields, then reconstruct it using the incompressibility condition. In order to remove our dependence on the vertical velocity entirely, I use the law of conservation of mass to remove all $\dot{\epsilon}_{zz}$ factors from the second invariant of the strain rate tensor. Rearranging terms from the conservation of mass equation:

$$\frac{\partial w}{\partial z} = -\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right). \tag{2.19}$$

From the strain rate tensor definition above, this is equivalent to

$$\dot{\epsilon}_{zz} = -(\dot{\epsilon}_{xx} + \dot{\epsilon}_{yy}). \tag{2.20}$$

Performing this substitution in the second invariant of the strain rate tensor given above and rearranging terms leads to an alternate form that has no dependence on the vertical velocity:

$$\dot{\epsilon} = \sqrt{\dot{\epsilon}_{xy}^2 + \dot{\epsilon}_{yz}^2 + \dot{\epsilon}_{xz}^2 + \dot{\epsilon}_{xx}\dot{\epsilon}_{yy} + \dot{\epsilon}_{yy}^2 + \dot{\epsilon}_{xx}^2} + \epsilon_0. \tag{2.21}$$

I can now finally expand the strain rate invariant factor in the viscosity and perform substitutions so that viscosity is formulated in terms of velocity gradients rather than strain rates:

$$\mu = \frac{A^{\frac{-1}{n}}}{2}\dot{\epsilon}^{\frac{1-n}{n}} \tag{2.22}$$

$$\mu = \frac{A^{\frac{-1}{n}}}{2}(\dot{\epsilon}_{xy}^2 + \dot{\epsilon}_{yz}^2 + \dot{\epsilon}_{xz}^2 + \dot{\epsilon}_{xx}\dot{\epsilon}_{yy} + \dot{\epsilon}_{yy}^2 + \dot{\epsilon}_{xx}^2)^{\frac{1-n}{2n}} \tag{2.23}$$

$$\mu = \frac{A^{\frac{-1}{n}}}{2}\left(\frac{1}{4}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2 + \frac{1}{4}\left(\frac{\partial u}{\partial z}\right)^2 + \frac{1}{4}\left(\frac{\partial v}{\partial z}\right)^2 + \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + \frac{\partial u}{\partial z}\frac{\partial v}{\partial y}\right)^{\frac{1-n}{2n}}. \tag{2.24}$$

2.1.4 Stress and Velocity

Given this relationship of both deviatoric stresses and velocities to strain rates, I can now achieve our goal of reducing the original conservation equations to a system of differential equations for the velocity components only. This will lead us to a set of coupled elliptic partial differential equations that can be numerically approximated to find the u and v components of the velocity field. Let us return to the first two equations in the system resulting from the conservation of momentum

$$\begin{aligned} \frac{\partial T_{xx}}{\partial x} + \frac{\partial T_{xy}}{\partial y} + \frac{\partial T_{xz}}{\partial z} &= 0 \\ \frac{\partial T_{yx}}{\partial x} + \frac{\partial T_{yy}}{\partial y} + \frac{\partial T_{yz}}{\partial z} &= 0. \end{aligned} \tag{2.25}$$

I want to express this equation in terms of the deviatoric stress rather than the total stress. Applying the substitutions derived above leads to

$$\frac{\partial}{\partial x}(2T'_{xx} + T'_{yy}) + \frac{\partial T'_{xy}}{\partial y} + \frac{\partial T'_{xz}}{\partial z} = \rho_i g \frac{\partial z_s}{\partial x} \tag{2.26}$$
$$\frac{\partial}{\partial y}(2T'_{yy} + T'_{xx}) + \frac{\partial T'_{xy}}{\partial y} + \frac{\partial T'_{yz}}{\partial z} = \rho_i g \frac{\partial z_s}{\partial y}.$$

Using Glen's flow law and the simplification of strain rates given in Equation **??**, the stress tensor can be written as

$$\begin{pmatrix} T'_{xx} & T'_{xy} & T'_{xz} \\ T'_{yx} & T'_{yy} & T'_{yz} \\ T'_{zx} & T'_{zy} & T'_{zz} \end{pmatrix} = \begin{pmatrix} 2\mu\dot{\epsilon}_{xx} & 2\mu\dot{\epsilon}_{xy} & 2\mu\dot{\epsilon}_{xz} \\ 2\mu\dot{\epsilon}_{yx} & 2\mu\dot{\epsilon}_{yy} & 2\mu\dot{\epsilon}_{yz} \\ 2\mu\dot{\epsilon}_{zx} & 2\mu\dot{\epsilon}_{zy} & 2\mu\dot{\epsilon}_{zz} \end{pmatrix} = \begin{pmatrix} 2\mu\frac{\partial u}{\partial x} & \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) & \mu\frac{\partial u}{\partial z} \\ \mu(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}) & 2\mu\frac{\partial v}{\partial y} & \mu\frac{\partial v}{\partial z} \\ \mu\frac{\partial u}{\partial z} & \mu\frac{\partial v}{\partial z} & 2\mu\frac{\partial w}{\partial z} \end{pmatrix}. \tag{2.27}$$

Directly substituting into the force balance equations gives

$$\frac{\partial}{\partial x}\left(4\mu\frac{\partial u}{\partial x} + 2\mu\frac{\partial v}{\partial y}\right) + \frac{\partial}{\partial y}\left(\mu\frac{\partial u}{\partial y} + \mu\frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial z}\left(\mu\frac{\partial u}{\partial z}\right) = \rho_i g\frac{\partial z_s}{\partial x} \tag{2.28}$$
$$\frac{\partial}{\partial y}\left(4\mu\frac{\partial v}{\partial y} + 2\mu\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial y} + \mu\frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial z}\left(\mu\frac{\partial v}{\partial z}\right) = \rho_i g\frac{\partial z_s}{\partial y}.$$

These equations show that the first order model can be thought of as augmenting the SSA with the vertical diffusion of velocity $\frac{\partial}{\partial z}\left(\mu\frac{\partial u}{\partial z}\right)$. If this term were removed, a vertically averaged membrane stress would be equated with the source term, which is the basis of the SSA.

Finally, I expand the nested derivatives and rearrange terms based on the observation that each equation in this coupled system solves for a one component. Thus, each equation has the component that it is solving for (u and v respectively) on

the left-hand side, and terms related to the orthogonal velocity component (v and u respectively) on the right-hand side:

$$\begin{aligned} U: 4\frac{\partial\mu}{\partial x}\frac{\partial u}{\partial x} + \frac{\partial\mu}{\partial y}\frac{\partial u}{\partial y} + \frac{\partial\mu}{\partial z}\frac{\partial u}{\partial z} + \mu\left(4\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) \\ = \rho_i g\frac{\partial z_s}{\partial x} - 2\frac{\partial\mu}{\partial x}\frac{\partial v}{\partial y} - \frac{\partial\mu}{\partial y}\frac{\partial v}{\partial x} - 3\mu\frac{\partial^2 v}{\partial x\partial y} \\ V: 4\frac{\partial\mu}{\partial y}\frac{\partial v}{\partial y} + \frac{\partial\mu}{\partial x}\frac{\partial v}{\partial x} + \frac{\partial\mu}{\partial z}\frac{\partial v}{\partial z} + \mu\left(4\frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial z^2}\right) \\ = \rho_i g\frac{\partial z_s}{\partial y} - 2\frac{\partial\mu}{\partial y}\frac{\partial u}{\partial x} - \frac{\partial\mu}{\partial x}\frac{\partial u}{\partial y} - 3\mu\frac{\partial^2 u}{\partial x\partial y}. \end{aligned} \tag{2.29}$$

2.2 Boundary Conditions

The interface of the ice sheet and a different medium such as ocean, air, or bedrock provides a constraint on the stress tensor at the interface. These constraints translate to a (usually Neumann-type) boundary condition on the differential equations for the velocity field. These boundary conditions and their derivations will now be presented.

2.2.1 Stress-Free Surface

The stress-free boundary condition occurs at the surface of the ice sheet and specifies that the shear stress parallel to the surface is zero [van der Veen, 1999]. This condition can be formalized [MacAyeal, 1996b] as

$$\mathbf{T}\cdot\hat{\mathbf{n}}_\mathbf{s} = 0. \tag{2.30}$$

Where $\mathbf{T}$ is the total (not deviatoric) stress tensor and $\hat{\mathbf{n}}_\mathbf{s}$ is a unit vector normal to the surface of the ice sheet. I derive $\hat{\mathbf{n}}_\mathbf{s}$ by normalizing

$$\mathbf{n_s} = \begin{pmatrix} -\frac{\partial z_s}{\partial x} \\ -\frac{\partial z_s}{\partial y} \\ 1 \end{pmatrix}, \tag{2.31}$$

where z_s is the elevation of the top of the ice sheet. Expanding the tensor dot product and ignoring the vertical resistive stresses gives the force balance for the upper boundary condition

$$\begin{aligned} T_{xx}\frac{\partial z_s}{\partial x} + T_{xy}\frac{\partial z_s}{\partial y} - T_{xz} &= 0 \\ T_{yx}\frac{\partial z_s}{\partial x} + T_{yy}\frac{\partial z_s}{\partial y} - T_{yz} &= 0 \\ -T_{zz} &= 0. \end{aligned} \tag{2.32}$$

Using the transformation from total stress to deviatoric stress, and observing that the term $\rho_i g(z_s - z)$ is zero since $z = z_s$, I arrive at the stress-free surface in terms of deviatoric stresses

$$\begin{aligned} (2T'_{xx} + T'_{yy})\frac{\partial z_s}{\partial x} + T'_{xy}\frac{\partial z_s}{\partial y} - T'_{xz} &= 0 \\ (2T'_{xx} + T'_{yy})\frac{\partial z_s}{\partial y} + T'_{yx}\frac{\partial z_s}{\partial x} - T'_{yz} &= 0. \end{aligned} \tag{2.33}$$

Applying the constitutive relationship, and noting that I can divide through the constant factor μ, I arrive at

$$\left(4\frac{\partial u}{\partial x}+2\frac{\partial v}{\partial y}\right)\frac{\partial z_s}{\partial x}+\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\frac{\partial z_s}{\partial y}-\frac{\partial u}{\partial z}=0 \tag{2.34}$$
$$\left(4\frac{\partial v}{\partial y}+2\frac{\partial u}{\partial x}\right)\frac{\partial z_s}{\partial y}+\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\frac{\partial z_s}{\partial x}-\frac{\partial v}{\partial z}=0.$$

Expanding and rearranging as I did for the interior equations,

$$4\frac{\partial u}{\partial x}\frac{\partial z_s}{\partial x}+\frac{\partial u}{\partial y}\frac{\partial z_s}{\partial y}-\frac{\partial u}{\partial z}=-2\frac{\partial v}{\partial y}\frac{\partial z_s}{\partial x}-\frac{\partial v}{\partial x}\frac{\partial z_s}{\partial y} \tag{2.35}$$
$$4\frac{\partial v}{\partial y}\frac{\partial z_s}{\partial y}+\frac{\partial v}{\partial x}\frac{\partial z_s}{\partial x}-\frac{\partial v}{\partial z}=-2\frac{\partial u}{\partial x}\frac{\partial z_s}{\partial y}-\frac{\partial u}{\partial y}\frac{\partial z_s}{\partial x}.$$

2.2.2 Basal Boundary Conditions

A number of different models exist to approximate the effects of different bed compositions. In the simplest case corresponding to a SIA model without a sliding law, the ice is simply frozen to the bed. In this case, I apply a Dirichlet boundary at the base the ice shelf that specifies that the velocity there is zero. A more complex bed model with a high bed strength has also been used successfully.

More interesting results, however, come from allowing the ice sheet to slide over the bed according to a model of the tendency of the bed to deform as ice moves over it [Schoof, 2006]. The two most commonly used models for bed deformation are for plastic and power law materials. Plastic bed models are believed to be appropriate for ice that rests on marine sediments, whereas a power law relation for bed strength may be more appropriate for ice resting on a layer of exposed bedrock.

A model for basal traction in cases where the bed consists of exposed bedrock is formulated as a power law material. At the moment, I restrict the power law exponent

to be 1, resulting in a linear model. To derive the force balance in this case, I divorce ourselves from a particular coordinate system and observe that given any unit vector tangential to the bedrock $\hat{\mathbf{t}}$, the traction can be expressed by taking the projection of basal drag onto that vector [Pattyn et al., 2008], as in

$$\beta^2 \hat{\mathbf{t}} \cdot \mathbf{v} = \hat{\mathbf{t}} \cdot (\mathbf{T} \cdot \hat{\mathbf{n}}_\mathbf{b}) = \tau_b, \tag{2.36}$$

where $\hat{\mathbf{n}}_\mathbf{b}$ is a unit vector normal to the lower surface and pointing into the bedrock that I obtain by normalizing

$$n_b = \begin{pmatrix} \frac{\partial z_b}{\partial x} \\ \frac{\partial z_b}{\partial y} \\ -1 \end{pmatrix}, \tag{2.37}$$

where z_b is the elevation of the base of the ice sheet. Because of the dot product, this produces only one equation, but since I am solving for the x and y components of velocity, I need two. But, because the equation is true for any tangent vector, and there are infinitely many tangent vectors to a surface, I can produce a linearly independent set of equations by choosing linearly independent tangent vectors. I choose

$$\mathbf{t_x} = \begin{pmatrix} 1 \\ 0 \\ \frac{\partial z_b}{\partial x} \end{pmatrix} \tag{2.38}$$
$$\mathbf{t_y} = \begin{pmatrix} 0 \\ 1 \\ \frac{\partial z_b}{\partial y} \end{pmatrix},$$

which work nicely with the coordinate system to break τ_b into components τ_{bx} and τ_{by} This gives the system of equations

$$\begin{aligned} \beta^2 \frac{\mathbf{t_x}}{||\mathbf{t_x}||} \cdot \mathbf{v} &= \frac{\mathbf{t_x}}{||\mathbf{t_x}||} \cdot \left(\mathbf{T} \frac{\mathbf{n_b}}{||\mathbf{n_b}||} \right) = \tau_{bx} \\ \beta^2 \frac{\mathbf{t_y}}{||\mathbf{t_y}||} \cdot \mathbf{v} &= \frac{\mathbf{t_y}}{||\mathbf{t_y}||} \cdot \left(\mathbf{T} \frac{\mathbf{n_b}}{||\mathbf{n_b}||} \right) = \tau_{by}. \end{aligned} \tag{2.39}$$

Expanding the vector and tensor products, I get

$$\begin{aligned} \beta^2 ||\mathbf{n_b}|| \left(u + w \frac{\partial z_b}{\partial x} \right) &= T_{xx} \frac{\partial z_b}{\partial x} + T_{xy} \frac{\partial z_b}{\partial y} - T_{xz} - T_{zz} \frac{\partial z_b}{\partial x} \\ \beta^2 ||\mathbf{n_b}|| \left(u + w \frac{\partial z_b}{\partial y} \right) &= T_{yx} \frac{\partial z_b}{\partial x} + T_{yy} \frac{\partial z_b}{\partial y} - T_{yz} - T_{zz} \frac{\partial z_b}{\partial y}. \end{aligned} \tag{2.40}$$

I make the assumption here that the vertical velocity component at the base is much larger than the horizontal component, which is valid at the bed as long as bedrock gradients are not too large and melt rates are negligible. Hence, I drop the w terms in the left hand side and substitute deviatoric stresses for total stresses, observing

that the pressure at the base is $\rho_i g H$, giving

$$\begin{aligned}\beta^2 u||\mathbf{n_b}|| &= (2T'_{xx} + T'_{yy} - \rho_i g H)\frac{\partial z_b}{\partial x} + T'_{xy}\frac{\partial z_b}{\partial y} - T'_{xz} + \rho_i g H \frac{\partial z_b}{\partial x} \\ \beta^2 v||\mathbf{n_b}|| &= (2T'_{yy} + T'_{xx} - \rho_i g H)\frac{\partial z_b}{\partial y} + T'_{yx}\frac{\partial z_b}{\partial x} - T'_{yz} + \rho_i g H \frac{\partial z_b}{\partial y}.\end{aligned} \tag{2.41}$$

Finally, the pressure terms cancel and I apply the constitutive relationship to reach

$$\begin{aligned}\beta^2 u||\mathbf{n_b}|| &= 2\mu\left(2\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\frac{\partial z_b}{\partial x} + \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\frac{\partial z_b}{\partial y} - \mu\frac{\partial u}{\partial z} \\ \beta^2 v||\mathbf{n_b}|| &= 2\mu\left(2\frac{\partial v}{\partial y} + \frac{\partial u}{\partial x}\right)\frac{\partial z_b}{\partial y} + \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\frac{\partial z_b}{\partial x} - \mu\frac{\partial u}{\partial z}.\end{aligned} \tag{2.42}$$

Finally, rearranging terms gives

$$\begin{aligned}U : 4\mu\frac{\partial u}{\partial x}\frac{\partial z_b}{\partial x} + \mu\frac{\partial u}{\partial y}\frac{\partial z_b}{\partial y} - \mu\frac{\partial u}{\partial z} &= \beta^2 u||\mathbf{n_b}|| - 2\mu\frac{\partial v}{\partial y}\frac{\partial z_b}{\partial x} - \mu\frac{\partial v}{\partial x}\frac{\partial z_b}{\partial y} \\ V : 4\mu\frac{\partial v}{\partial y}\frac{\partial z_b}{\partial y} + \mu\frac{\partial v}{\partial x}\frac{\partial z_b}{\partial x} - \mu\frac{\partial v}{\partial z} &= \beta^2 v||\mathbf{n_b}|| - 2\mu\frac{\partial u}{\partial x}\frac{\partial z_b}{\partial y} - \mu\frac{\partial u}{\partial y}\frac{\partial z_b}{\partial x}.\end{aligned} \tag{2.43}$$

The formulation for basal traction due a to plastic bed is given by Schoof [2006] as

$$\tau_0 \hat{\mathbf{t}} \cdot \frac{\mathbf{v}}{|\mathbf{v}|} = \hat{\mathbf{t}} \cdot (\mathbf{T}\hat{\mathbf{n}}_\mathbf{b}) = \tau_b, \tag{2.44}$$

where $|\mathbf{v}| = (u^2 + v^2)^{\frac{1}{2}}$.

The rest of the formulation in the linear bed case is valid for this case as well, though with $\beta^2 = \frac{\tau_0}{(u^2+v^2)^{\frac{1}{2}}}$. However, is a much harder problem to solve because the strength of the nonlinearity is greater. In general, the type of boundary condition

is prescribed based on the type of till that the glacier flows over: in the case of ice flow directly over bedrock, a linear bed model is preferred, but in cases where ice is flowing over a deformable till the plastic bed model is preferred. When working with real-world ice geometries, nontrivial inverse modelling is needed to specify the exact nature of the basal boundary condition [Bueler and Brown, 2009].

2.2.3 Stress-Free Base Condition

Finally, we must consider the lower boundary condition of an ice shelf. In this case, I balance the outward-facing lower surface normal with hydrostatic pressure MacAyeal [1996b]. Assuming the ice shelf is floating in equilibrium, the lower surface will be at elevation $-\frac{\rho_i}{\rho_w}H$. Then,

$$\mathbf{T}\hat{\mathbf{n}}_\mathbf{b} = -\rho_w g(\frac{\rho_i}{\rho_w}H)\hat{\mathbf{n}}_\mathbf{b} \tag{2.45}$$

where again, $\hat{n}_b$ is the outward-facing normal unit vector to the ice surface that is a normalization of

$$\mathbf{n_b} = \begin{pmatrix} \frac{\partial z_b}{\partial x} \\ \frac{\partial z_b}{\partial y} \\ -1 \end{pmatrix}, \tag{2.46}$$

Unlike the linear bed case above, I do not formulate this in terms of tangents to the lower surface because there is no resistance; the formulation of this boundary condition more closely resembles the stress-free boundary condition above. Expanding the tensor product and canceling the normalization factor gives three equations

$$\begin{aligned} T_{xx}\frac{\partial z_b}{\partial x} + T_{xy}\frac{\partial z_b}{\partial y} - T_{xz} &= -\rho_i g H \frac{\partial z_b}{\partial x} \\ T_{xy}\frac{\partial z_b}{\partial x} + T_{yy}\frac{\partial z_b}{\partial y} - T_{yz} &= -\rho_i g H \frac{\partial z_b}{\partial y} \\ -T_{zz} &= \rho_i g H. \end{aligned} \tag{2.47}$$

Notice that I have simplified the right-hand side in a somewhat revealing way. Because I have assumed hydrostatic equilibrium, of course the ice shelf will float so as to equalize pressure inside and outside of it. This is exactly what the right-hand-side, which is the same as the cryostatic pressure at the base of the ice sheet, is telling us.

Next I substitute in deviatoric stresses. When I do this, the final equation becomes tautological (again reflecting hydrostatic equilibrium) so I remove it and are left with the two equations

$$\begin{aligned} (2T'_{xx} + T'_{yy} + \rho_i g H)\frac{\partial z_b}{\partial x} + T'_{xy}\frac{\partial z_b}{\partial y} - T'_{xz} &= \rho_i g H \frac{\partial z_b}{\partial x} \\ T'_{xy}\frac{\partial z_b}{\partial x} + (2T'_{yy} + T'_{xx} + \rho_i g H)\frac{\partial z_b}{\partial y} - T'_{yz} &= \rho_i g H \frac{\partial z_b}{\partial y}. \end{aligned} \tag{2.48}$$

Notice here that I can cancel $\frac{\partial z_b}{\partial x}\rho_i g H$ and $\frac{\partial z_b}{\partial y}\rho_i g H$, leaving me with a stress-free base. From here on the basal boundary condition can be easily adapted from the stress-free surface by replacing surface gradients with bed gradients Pattyn [2003]. Observe also that this equation is equivalent to the hard bed model with $\beta^2 = 0$, which is how it is actually implemented in the software.

2.2.4 Ice Shelf Lateral Boundary Condition

The marine front of an ice shelf is subject to back stresses from seawater "pushing back" on the shelf front. This boundary condition is often presented in a vertically integrated form as originally developed by Morland [1987]. However, because the first-order model is vertically explicit, I start with a vertically explicit form, with

$$\mathbf{T}(z) \cdot \mathbf{n} = \rho_w g z \mathbf{n}, z < 0. \tag{2.49}$$

below the water level. Due to the approximation of atmospheric pressure as 0, above the water level it becomes

$$\mathbf{T}(z) \cdot \mathbf{n} = 0, z \geq 0. \tag{2.50}$$

If we consider a two-dimensional map of an ice shelf, then $\mathbf{n}$ is a vector that is normal to the curve formed by the shelf front, pointing outward, with $n_z = 0$. Because it is not strictly true that ice calves perpendicularly to the water's surface, equating n_z with zero is an approximation. This integral assumes a coordinate system where z is strictly elevation (that is, z=0 at sea level, values increase as we ascend).

For most of the following derivation, I will neglect the above-water case and focus on the case where $z < 0$. Applying the tensor product and using the assumptionthat $n_z = 0$, this leads to the system of equations

$$\begin{aligned} T_{xx} n_x + T_{xy} n_y &= \rho_w g z n_x \\ T_{yx} n_x + T_{yy} n_y &= \rho_w g z n_y \\ T_{zx} n_x + T_{zy} n_y &= 0. \end{aligned} \tag{2.51}$$

Under the assumption that vertical resistive stresses are zero, we can drop the third equation. Substituting deviatoric stresses for total stresses,

$$(2T'_{xx} + T'_{yy} - \rho_i g(z_s - z))n_x + T'_{xy}n_y = \rho_w g z n_x$$

$$T'_{yx}n_x + (2T'_{yy} + T'_{xx} - \rho_i g(z_s - z))n_y = \rho_w g z n_y \tag{2.52}$$

$$\tag{2.53}$$

Applying the constitutive relationship and replacing strain rates with velocities:

$$2\mu\left[\left(2\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)n_x + \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)n_y\right] = n_x\left(\rho_i g\left(z_s - z\right) + \rho_w g z\right)$$

$$2\mu\left[\frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)n_x + \left(\frac{\partial u}{\partial x} + 2\frac{\partial v}{\partial y}\right)n_y\right] = n_y\left(\rho_i g\left(z_s - z\right) + \rho_w g z\right) \tag{2.54}$$

$$\tag{2.55}$$

Here I re-apply one of the main assumptions of the SSA: the assumption of plug flow, or a lack of vertical dependence of velocity in ice shelf regions. However, plug flow is not necessarily the case at the marine front of an ice shelf, as the stress at each vertical layer would be different. In fact, the SSA is known to treat this area of the shelf incorrectly, though differences are "winnowed out" within a few ice thicknesses of the shelf [Morland, 1987] [MacAyeal, 1996b]. This assumption, however, allows a return to a vertically averaged form of the right hand side by integrating over the cryostatic and hydrostatic pressures

$$\frac{1}{H}\left[\int_0^{-\frac{\rho_i}{\rho_w}H} \rho_w g z dz + \int_{\left(1-\frac{\rho_i}{\rho_w}\right)H}^{-\frac{\rho_i}{\rho_w}H} \rho_i g z dz\right] = \frac{1}{2}\rho_i g H\left(1 - \frac{\rho_i}{\rho_w}\right). \tag{2.56}$$

Thus, I finally write the equations for the lateral boundary condition as

$$2\mu\left[\left(2\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)n_x+\frac{1}{2}\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)n_y\right]=\frac{1}{2}\rho_i g H\left(1-\frac{\rho_i}{\rho_w}\right)n_x \tag{2.57}$$
$$2\mu\left[\frac{1}{2}\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)n_x+\left(\frac{\partial u}{\partial x}+2\frac{\partial v}{\partial y}\right)n_y\right]=\frac{1}{2}\rho_i g H\left(1-\frac{\rho_i}{\rho_w}\right)n_y,$$

and rearrange terms to get

$$4\mu\frac{\partial u}{\partial x}n_x+\mu\frac{\partial u}{\partial y}n_y=\frac{1}{2}\rho_i g H\left(1-\frac{\rho_i}{\rho_w}\right)n_x-2\mu\frac{\partial v}{\partial y}n_x-\mu vxn_y \tag{2.58}$$
$$4\mu\frac{\partial v}{\partial y}n_y+\mu\frac{\partial v}{\partial x}n_x=\frac{1}{2}\rho_i g H\left(1-\frac{\rho_i}{\rho_w}\right)n_y-2\mu\frac{\partial u}{\partial x}n_u-\mu uyn_x$$

CHAPTER 3 NUMERICAL APPROXIMATION

The higher-order velocity equations presented Chapter 2 are a reduced-complexity version of the Navier-Stokes equations for fluid dynamics of a non-Newtonian fluid. Unfortunately, this formulation of the Navier-Stokes equations does not have a known general analytical solution, though solutions do exist for special cases. Therefore, we must "solve" the equations using numerical rather than analytical techniques. The need to solve these equations for real-world problems involving ice whose geometry has no analytical form further necessitates numerical approximation. In this case, Pattyn applies a finite-difference approximation to solve the velocity equations on a three-dimensional grid that is regular in the horizontal dimensions and irregular in the vertical dimension. This chapter presents the numerical techniques used to arrive at this solution.

3.1 Rescaled Vertical Coordinate

Both Pattyn's standalone model and Glimmer rescale the vertical coordinate by the ice thickness [Pattyn, 2003] [Rutt et al., 2009]. This is similar to the s-coordinate used in atmospheric modeling, first described by Phillips [1957]. For consistency with CISM's documentation I will refer to the new coordinate as σ rather than the ζ that Pattyn uses or the s that is often seen outside of ice sheet modeling. This coordinate is defined such that $\sigma \in [0, 1]$, where 0 is always at the surface of the ice sheet and 1

is always at the base. An elevation z, then, is mapped to a vertical coordinate σ as

$$\sigma = (z_s - z)/H \tag{3.1}$$

where z_s is the ice surface elevation and H is the ice thickness [Pattyn, 2003].

Computational nodes at the same σ level thus have the same *percentage* of ice above and below them rather than being at the same elevation. The vertical coordinate is thus described as "topography following" in that changes in the topography are reflected by changes in the levels, as illustrated in Figure 3.1.

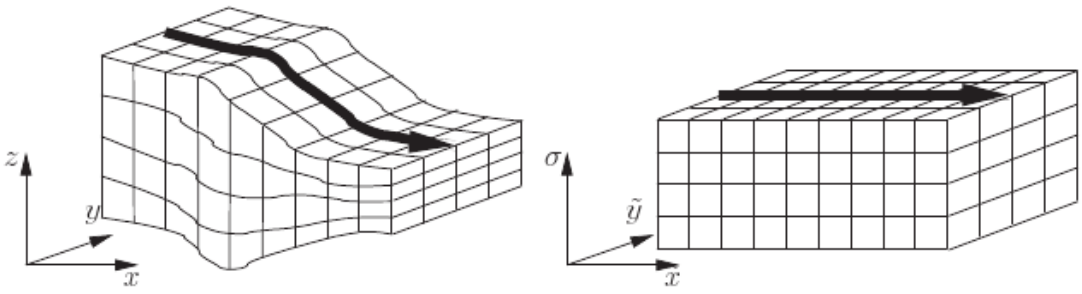

Figure 3.1 An illustration of the rescaled vertical coordinate used in CISM, from the Glimmer documentation [Hagdorn et al., 2006].

There are several advantages of using such a coordinate system. First a vertically unscaled model would require an ice grid consisting of several voxels, many of which on each vertical column would be above or below the ice sheet at that column and would thus not contain useful data. For a discussion of a vertically explicit model, see Bueler et al. [2008]. A vertically rescaled system, regardless of the topography, fully utilizes the vertical layers in each column. In addition, such a system allows for an uneven spacing to be used in the vertical, allowing for greater computational

resolution where needed.

3.1.1 Derivation of Rescaling Parameters

As a result of this transformation, a coordinate (x, y, z) is mapped to (x', y', σ). Application of the chain rule shows that that function derivatives must be re-written (using f as a generic function and $\frac{\partial f}{\partial x}$ as an example) as Pattyn [2003]

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial x'}\frac{\partial x'}{\partial x} + \frac{\partial f}{\partial y'}\frac{\partial y'}{\partial x} + \frac{\partial f}{\partial \sigma}\frac{\partial \sigma}{\partial x}. \tag{3.2}$$

Similarly for $\frac{\partial f}{\partial y}$ and $\frac{\partial f}{\partial z}$. Pattyn simplifies this by assuming

$$\frac{\partial x'}{\partial x}, \frac{\partial y'}{\partial y} = 1 \tag{3.3}$$

and

$$\frac{\partial x'}{\partial y}, \frac{\partial x'}{\partial z}, \frac{\partial y'}{\partial x}, \frac{\partial y'}{\partial z} = 0. \tag{3.4}$$

This assumption is valid if the bed and surface gradients are not too large [Pattyn, 2003]. This simplifies the above to

$$\begin{aligned} \frac{\partial f}{\partial x} &= \frac{\partial f}{\partial x'} + \frac{\partial f}{\partial \sigma}\frac{\partial \sigma}{\partial x} \\ \frac{\partial f}{\partial y} &= \frac{\partial f}{\partial y'} + \frac{\partial f}{\partial \sigma}\frac{\partial \sigma}{\partial y} \\ \frac{\partial f}{\partial z} &= \frac{\partial f}{\partial \sigma}\frac{\partial \sigma}{\partial z}. \end{aligned} \tag{3.5}$$

Because the algebraic manipulation from here on is tedious but not particularly difficult, I will not present it in full and will instead cite Pattyn's result [Pattyn,

2003]. Pattyn defines rescaling parameters a_x and a_y (which can be seen as $\frac{\partial\sigma}{\partial x}$ and $\frac{\partial\sigma}{\partial y}$); b_x and b_y; and c_{xy}. Following Pattyn, I present only the x and z derivative cases, as the y derivative case is analogous to the x derivative case. The rescaling parameters are

$$a_x = \frac{1}{H}(\frac{\partial z_s}{\partial x'} - \sigma\frac{\partial H}{\partial x'}) \tag{3.6}$$

$$\begin{aligned} b_x &= \frac{\partial a_x}{\partial x'} + a_x\frac{\partial a_x}{\partial \sigma} \\ &= \frac{1}{H}(\frac{\partial^2 z_s}{\partial x'^2} - \sigma\frac{\partial^2 H}{\partial x'^2} - 2a_x\frac{\partial H}{\partial x'}) \end{aligned} \tag{3.7}$$

$$\begin{aligned} c_{xy} &= \frac{\partial a_y}{\partial x'} + a_x\frac{\partial a_y}{\partial \sigma} \\ &= \frac{\partial a_x}{\partial y'} + a_y\frac{\partial a_x}{\partial \sigma}. \end{aligned} \tag{3.8}$$

Using these parameters, expressions for the derivatives become

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial x'} + a_x\frac{\partial f}{\partial \sigma} \tag{3.9}$$

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial^2 f}{\partial x'^2} + b_x\frac{\partial f}{\partial \sigma} + a_x^2\frac{\partial^2 f}{\partial \sigma^2} + 2a_x\frac{\partial^2 f}{\partial x'\partial \sigma} \tag{3.10}$$

$$\frac{\partial f}{\partial z} = -\frac{1}{H}\frac{\partial f}{\partial \sigma} \tag{3.11}$$

$$\frac{\partial^2 f}{\partial z^2} = \frac{1}{H^2}\frac{\partial^2 f}{\partial \sigma^2} \tag{3.12}$$

$$\frac{\partial^2 f}{\partial x\partial z} = \frac{1}{H}(\frac{1}{H}\frac{\partial H}{\partial x'}\frac{\partial f}{\partial \sigma} - \frac{\partial^2 f}{\partial x'\partial \sigma} - a_x\frac{\partial^2 f}{\partial \sigma^2}) \tag{3.13}$$

$$\frac{\partial^2 f}{\partial x\partial y} = \frac{\partial^2 f}{\partial x'\partial y'} + c_{xy}\frac{\partial f}{\partial \sigma} + a_y\frac{\partial^2 f}{\partial x'\partial \sigma} + a_x\frac{\partial^2 f}{\partial y'\partial \sigma} + a_x a_y\frac{\partial^2 f}{\partial \sigma^2}. \tag{3.14}$$

3.1.2 Transformation of Equations

Given the definitions above, deriving rescaled versions of the equations presented in Chapter 2 is "simply" a matter of performing the correct substitutions.

On the interior of the domain, the velocity equations become

$$\begin{aligned}
&4\mu_{x'}\frac{\partial u}{\partial x'} + \mu_{y'}\frac{\partial u}{\partial y'} + 4\mu\frac{\partial^2 u}{\partial x'^2} \\
&\quad + \left(4a_x\mu_{x'} + \mu(4b_x + b_y) + a_y\mu_{y'} + \frac{1}{H^2}\frac{\partial \mu}{\partial \sigma}\right)\frac{\partial u}{\partial \sigma} \\
&\quad + \mu\frac{\partial^2 u}{\partial y'^2} + \mu\left(4a_x^2 + a_y^2 + \frac{1}{H^2}\right)\frac{\partial^2 u}{\partial \sigma^2} + 8a_x\mu\frac{\partial^2 u}{\partial x'\partial \sigma} \\
&\quad + 2a_y\mu\frac{\partial^2 u}{\partial y'\partial \sigma} = \rho g\frac{\partial z_s}{\partial x'} - \mu_{y'}\frac{\partial v}{\partial x'} - 2\mu_{x'}\frac{\partial v}{\partial y'} \\
&\quad - (2a_y\mu_{x'} + a_x\mu_{y'} + 3c_{xy}\mu)\frac{\partial v}{\partial \sigma} - 3a_xa_y\mu\frac{\partial^2 v}{\partial \sigma^2} \\
&\quad - 3\mu\frac{\partial^2 v}{\partial x'\partial y'} - 3a_y\mu\frac{\partial^2 v}{\partial x'\partial \sigma} - 3a_x\mu\frac{\partial^2 v}{\partial y'\partial \sigma},
\end{aligned} \tag{3.15}$$

where

$$\mu_{x'} = \frac{\partial \mu}{\partial x'} + a_x\frac{\partial \mu}{\partial \sigma}.$$

The stress-free surface condition becomes

$$\begin{aligned}
&4\frac{\partial z_s}{\partial x'}\frac{\partial u}{\partial x'} + \frac{\partial z_s}{\partial y'}\frac{\partial u}{\partial y'} + (4a_x\frac{\partial z_s}{\partial x'} + a_y\frac{\partial z_s}{\partial y'} + \frac{1}{H})\frac{\partial u}{\partial \sigma} \\
&\quad = -2\frac{\partial z_s}{\partial x'}\frac{\partial v}{\partial y'} - \frac{\partial z_s}{\partial y'}\frac{\partial v}{\partial x'} - (2a_y\frac{\partial z_s}{\partial x'} + a_x\frac{\partial z_s}{\partial y'})\frac{\partial v}{\partial \sigma}.
\end{aligned} \tag{3.16}$$

The hard bed boundary condition becomes

$$4\mu\frac{\partial z_b}{\partial x'}\frac{\partial u}{\partial x'} + \mu\frac{\partial z_b}{\partial y'}\frac{\partial u}{\partial y'} + \mu(4a_x\frac{\partial z_b}{\partial x'} + a_y\frac{\partial z_b}{\partial y'} - \frac{1}{H})\frac{\partial u}{\partial \sigma}$$
$$= \beta^2 v||\mathbf{n_b}|| - 2\mu\frac{\partial z_b}{\partial x'}\frac{\partial v}{\partial y'} - \mu\frac{\partial z_b}{\partial y'}\frac{\partial v}{\partial x'} - \mu(2a_y\frac{\partial z_b}{\partial x'} + a_x\frac{\partial z_b}{\partial y'})\frac{\partial v}{\partial \sigma}. \quad (3.17)$$

Finally, the lateral ice shelf boundary condition becomes

$$4\mu n_x\frac{\partial u}{\partial x'} + \mu n_y\frac{\partial u}{\partial y'} + \mu(4a_x n_x + a_y n_y)\frac{\partial u}{\partial \sigma}$$
$$= n_x\frac{1}{2}\rho_i g H(1 - \frac{\rho_i}{\rho_w}) - 2\mu n_x\frac{\partial v}{\partial y'} - \mu n_y\frac{\partial v}{\partial x'} - \mu(2a_y n_x + a_x n_y)\frac{\partial v}{\partial \sigma}. \quad (3.18)$$

Note that no transformation of the source term is needed, as we had previously removed the vertical dependence.

3.2 Nonlinear Iteration

The equations that govern the velocity field in an ice sheet are nonlinear, and therefore require an iterative solution. The nonlinear system of equations arising from the stress balance is solved by computing the viscosity using the velocity values from the previous iteration. We use velocities computed using the shallow ice approximation as an initial guess. A separate linear system is then set up for the U and V components of velocity, each solving one of the equations that arises from the stress balance. While solving for the U component, any derivatives of V are computed numerically from the values from the previous iteration. The same is done when computing V. Pattyn found this to be more efficient than building a linear system that directly couples U and V, as the unused strategy builds a matrix with four times as many

elements and does not appear to speed convergence [Pattyn, 2003]. The pseudocode for this high-level iteration is

1. $u, v \leftarrow$ shallow_ice_guess()
2. $c \leftarrow 0$
3. Repeat until error $<$ tolerance:
4. $\quad \mu \leftarrow$ compute_viscosity(u, v)
5. $\quad u^\star \leftarrow$ compute_u_component(v, μ)
6. $\quad v^\star \leftarrow$ compute_v_component(u, μ)
7. $\quad u, v, c, error \leftarrow$ unstable_manifold_correct(u, $u^\star$, v, $v^\star$, c)

The nonlinear iteration method used is Picard iteration (also known as successive substitution) with unstable manifold correction, an accelerator method designed specifically for problems that arise in ice sheet modeling [Hindmarsh and Payne, 1996]. The method considers Picard iteration to be deriving successive velocities $\mathbf{u}_i, \mathbf{u}_{i+1}, ...$ by computing and applying a series of correction vectors $\mathbf{c}_i, \mathbf{c}_{i+1}, ...$ such that $\mathbf{u}_{i+1} = \mathbf{u}_i + \mathbf{c}_i$. Unstable manifold correction relaxes the partial solutions by scaling the correction vector such that

$$\mathbf{u}_{i+1} = \mathbf{u}_i + \alpha \mathbf{c}_i, \alpha = \frac{\|\mathbf{c}_{i-1}\|}{\|\mathbf{c}_i - \mathbf{c}_{i-1}\|}. \tag{3.19}$$

This correction is applied if the angle between successive correction vectors is less than $\frac{5\pi}{6}$, a threshold identified by **?** to work well. Iteration ends when the residual, computed as

$$\frac{\|\mathbf{c}_i\|}{\|\mathbf{u}_{i+1}\|}, \tag{3.20}$$

is beneath an error tolerance. I have found that tolerances on the order of 10^{-4} work well, and are a good compromise between the accuracy of the solution and the speed at which it is computed. Some problems are unstable enough that they will never reach that tolerance, however. In this case, increasing the tolerance by a small percentage every ten iterations can ensure good convergence where possible and a compromise where not. This is turned off by default in the model.

I have found no situations in which the use of unstable manifold correction slows convergence compared to pure Picard iteration, and it is often much faster than when using Picard iteration. However, this method still tends to converge somewhat slowly, particularly in cases where a nonuniform basal boundary condition is specified. Its main benefit is that it is easy to implement and it is unknown whether more contemporary methods such as Newton-Krylov iteration would perform better. It is a potential topic of future research to characterize this method and compare its performance to that of other, better-studied iteration methods.

3.3 Finite Difference Approximation

In order to obtain a numerical solution for the boundary value problem described in Chapter 2 the model domain is discretized using a regularly spaced grid in the horizontal dimensions and an irregularly spaced grid in the vertical dimensions. This gives us a three-dimensional grid of points. Rather than computing a smooth analytical solution for the equations in the entire domain, we compute an approximation of the value of the solution at each of these grid nodes.

As an improvement to Pattyn's original model, rather than including all grid nodes in the domain in the computation, I only solve for those grid nodes containing ice. In many cases, particularly when solving on real-world domains, this can greatly reduce

the size of the computation.

The vertically rescaled equations are solved for each point on this grid using a fully implicit scheme. Here I will briefly describe the implicit technique used to solve boundary value problems; a more complete discussion can be found in Press et al. [1992]. As usual for the finite difference method, we obtain a discretized version of the velocity equations by replacing the partial differential operators with approximations of the derivatives using neighboring grid nodes. The nonlinearities that arise from the viscosity are handled through an iteration technique described in

3.4

??.

Using these approximations of the momentum balance equations, I then construct a system of linear equations with a number of variables equal to the number of computational nodes. If, for example, I use ten nodes in each horizontal dimension and five vertical layers, it results in a linear system of 500 variables. If we consider the matrix that represents this linear system with each row corresponding to a grid node, we consider the grid nodes that enter into the finite difference approximation of the equations at that node and enter coefficients into the repsective columns.

Because there are two equations for two variables, one equation is used to solve for each velocity component while holding the other velocity component constant. Derivatives of the component that is not being solved for are computed numerically and entered into the right-hand side along with any source term.

In all cases, the finite difference approximations used are 2nd-order accurate. On the interior of the domain, centered differences are always applied. On the regluar grid in the horizontal dimensions, the approximation is

$$\frac{\partial f}{\partial x} \approx \frac{f_{i+1} - f_{i-1}}{2\Delta x} \tag{3.21}$$

$$\frac{\partial^2 f}{\partial x^2} \approx \frac{f_{i+1} + f_{i-1} - 2f_i}{(\Delta x)^2}. \tag{3.22}$$

At the lateral edge of an ice sheet, horizontal derivatives are still centered, with a Dirichlet boundary of 0 velocity applied at one point past the edge of the ice. This works in cases where it can be assumed that the ice sheet thickness gradually becomes zero (i.e. there is no ice cliff). A more accurate discretization could be obtained by upwinding at these points, as described by Fuyuki et al. [2007], but this has not yet been implemented and should be considered a subject for future work. An alternate method that has been used by Stephen Price (from personal conversations) is to specify a zero-value Dirichlet condition at the land margin instead of one cell away, assuming that the important features in the velocity occur in the interior, at grounding lines, and at calving fronts.

The calving front of an ice shelf must be handled differently, as it is a sheer cliff of ice rather than a gradual drop-off and cannot be assumed to have no flux! This boundary condition is applied to all grid cells that are themselves floating and that are either directly or diagonally adjacent to open ocean (if we only include cells that are directly adjacent to ocean, the $\frac{\partial^2 v}{\partial x \partial y}$ terms difference into cells with no ice). In these cases, centered derivatives are applied where possible, and upwinded derivatives are applied where using a centered difference would reference a point that is off the ice shelf.

In software, this is implemented using a mask field that assigns integer values to different kinds of grid points such as open ocean, floating ice, grounded ice, and transition zones such as the grounding line and calving front, as demonstrated in

Figure 3.2. In addition to determining the type of computation, the mask field is important in the case of the calving front to determine both the direction of the vector that is normal to the shelf front and the type of derivatives used at that computational node. This is demonstrated in Figure 3.3.

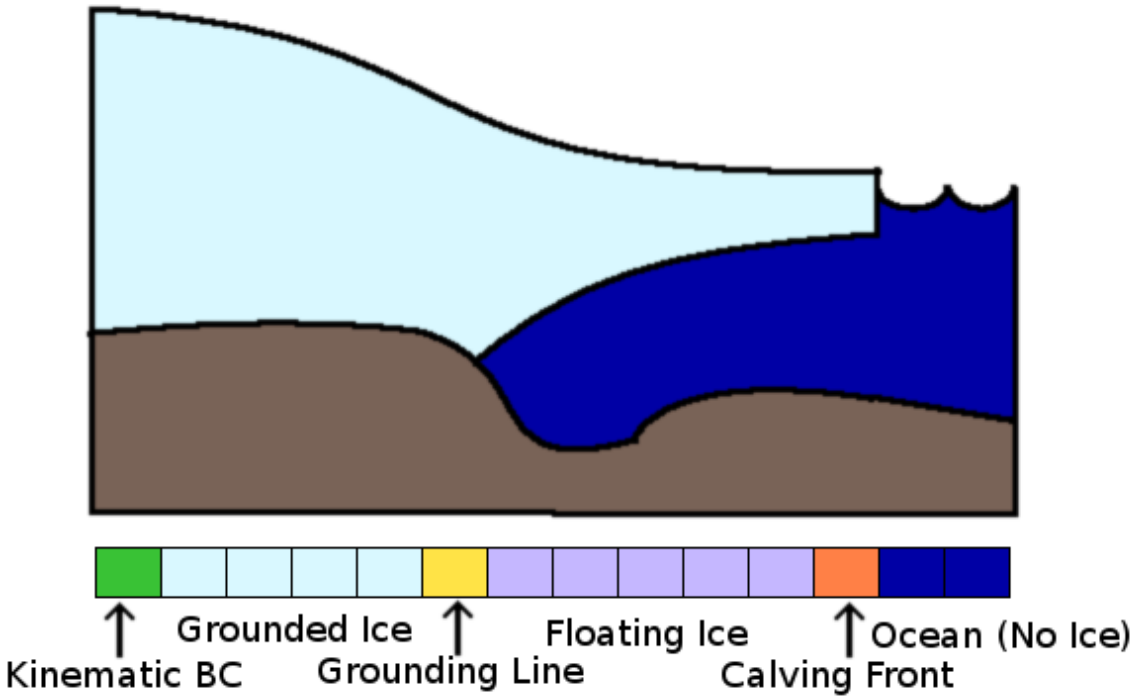

Figure 3.2 A mask field is constructed to identify what kind of grid point a computational node is. Here a flowline stream-shelf system and the corresponding one-dimensional mask are shown.

Upwinded differences on the calving front are applied using the standard second-order upwinded and downwinded finite difference discretizations

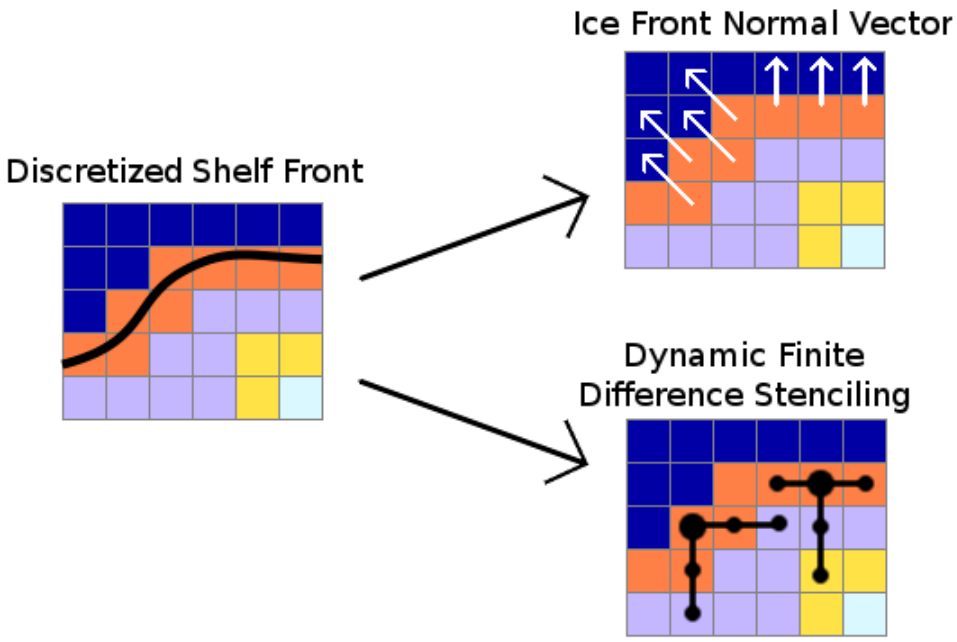

Figure 3.3 The mask field, determined by the geometry at the calving front of an ice shelf, is used to construct both the shelf front normal vectors and to dynamically determine where upwinding of derivatives is needed to remain on the shelf.

$$\frac{\partial f}{\partial x} \approx \frac{3f_i - 4f_{i-1} + f_{i-2}}{2\Delta x} \tag{3.23}$$

$$\frac{\partial f}{\partial x} \approx \frac{-3f_i + 4f_{i+1} - f_{i+2}}{2\Delta x}. \tag{3.24}$$

In the vertical dimension, derivatives must be handled differently than their horizontal counterparts because the vertical grid spacing may be nonuniform. We define the vertical grid as a list σ_i that contains the value of the rescaled coordinate at each vertical layer $1 \leq i \leq n$, subject to the restrictions $\sigma_1 = 0$, $\sigma_n = 1$, and $\sigma_i < \sigma_{i+1}$. Second-order accurate derivative approximations with nonuniform grid spacing are

$$\begin{aligned}\frac{\partial f}{\partial \sigma} \approx\ & f_{k-1}\frac{\sigma_i - \sigma_{i+1}}{(\sigma_i - \sigma_{i-1}) \cdot (\sigma_{i+1} - \sigma_{i-1})} + f_k \frac{\sigma_{i+1} + \sigma_{i-1} - 2\sigma_i}{(\sigma_i - \sigma_{i-1}) \cdot (\sigma_{i+1} - \sigma_i)} \\ & + f_{k+1}\frac{\sigma_i - \sigma_{i-1}}{(\sigma_{i+1} - \sigma_i) \cdot (\sigma_{i+1} - \sigma_{i-1})}\end{aligned} \tag{3.25}$$

$$\begin{aligned}\frac{\partial^2 f}{\partial \sigma^2} \approx\ & f_{k-1}\frac{2}{(\sigma_i - \sigma_{i-1}) \cdot (\sigma_{i+1} - \sigma_{i-1})} + f_k \frac{2}{(\sigma_{i+1} - \sigma_i) \cdot (\sigma_i - \sigma_{i-1})} \\ & + f_{k+1}\frac{2}{(\sigma_{i+1} - \sigma_i) \cdot (\sigma_{i+1} - \sigma_{i-1})}.\end{aligned} \tag{3.26}$$

At the upper and lower ice surfaces, upwinded and downwinded versions of the derivatives are used, as there are no computational nodes in the bedrock or in the atmosphere. These are

$$\frac{\partial f}{\partial \sigma} \approx f_{k-2}\frac{\sigma_i - \sigma_{i-1}}{(\sigma_{i-1} - \sigma_{i-2}) \cdot (\sigma_i - \sigma_{i-2})} + f_{k-1}\frac{\sigma_i - \sigma_{i-2}}{(\sigma_i - \sigma_{i-1}) \cdot (\sigma_{i-1} - \sigma_{i-2})} \tag{3.27}$$
$$+ f_k \frac{2\sigma_i - \sigma_{i-1} - \sigma_{i-2}}{(\sigma_i - \sigma_{i-1}) \cdot (\sigma_i - \sigma_{i-1})}$$
$$\frac{\partial f}{\partial \sigma} \approx f_{k+2}\frac{\sigma_{i+1} - \sigma_i}{(\sigma_{i+2} - \sigma_{i+1}) \cdot (\sigma_{i+2} - \sigma_i)} + f_{k+1}\frac{\sigma_{i+2} - \sigma_i}{(\sigma_{i+2} - \sigma_{i+1}) \cdot (\sigma_{i+1} - \sigma_i)} \tag{3.28}$$
$$+ f_k \frac{2\sigma_i - \sigma_{i+1} - \sigma_{i+2}}{(\sigma_{i+2} - \sigma_i) \cdot (\sigma_{i+1} - \sigma_i)}.$$

3.5 Solving the Linear System

Holding μ and the across-flow velocity component constant, I build a sparse linear system as above. A separate matrix is built and solved for each velocity equation to solve for the two components of velocity. Again, this linear system represents an approximation of the partial differential equations describing the momentum balance, discretized onto a finite grid. The last step in computing velocities, then, is to solve the linear system.

Because the computational time required to solve the higher-order system is dominated by finding the solution to the system of linear equations that arises from the numerical discretization, an efficient method of storing and solving a linear system is required. A simple "dense-matrix" storage format for a linear system consists of a two-dimensional array. This incurs large storage and computational inefficiencies, however, as most of the entries in the array are zeroes. An $N \times N$ linear system requires $\mathcal{O}(N^2)$ storage, most of which is wasted [Press et al., 1992].

I therefore store the linear system arising from the discretized equations in a triad-format sparse matrix, which consists of three arrays storing the row, column, and value of each non-zero entry in the matrix. If the matrix has m nonzero elements,

sparse matrices require only $\mathcal{O}(m)$ memory. If the number of nonzero entries m is bounded above by a linear function of the number of variables N (as is the case in systems arising from finite difference approximations), then the memory required is $\mathcal{O}(N)$ [Press et al., 1992]. Therefore, for reasonably large computational domains, the sparse storage format is much more efficient than the full matrix storage [Press et al., 1992].

The option now exists in our model to use a variety of sparse linear solvers; this will be returned to in Chapter 4. In general, I have found that the Biconjugate Gradient (BiCG) method described by Press et al. [1992] and implemented in the SLAP library by Seager [1989] is an adequate solver when combined with an incomplete LU preconditioner. This is an iterative method for solving sparse linear systems that frequently outperforms direct methods such as the unsymmetric multifrontal method implemented in UMFPACK by Davis [2004].

However, the biconjugate gradient method does not always reach a solution: in particular, if the solution vector is very close to zero, the method can stagnate before reaching convergence. In this case, I have found it to be effective to bootstrap the nonlinear solve by using the direct solver in UMFPACK to solve the first few iterations, then use the much faster biconjugate gradient method to continue with the solution. Additionally, during model development, use of a direct solver can be a good debugging tool, as it can provide insight into code errors that lead to instabilities that prevent the BiCG solver from converging. Finally, certain problems are somewhat sensitive to errors introduced by partially converged SLAP solutions, and either a much lower error tolerance or a direct solver must be used throughout the computation. I have found, however, no case where the BiCG solver converges but where UMFPACK nevertheless outperforms it.

CHAPTER 4 SOFTWARE INTEGRATION

The integration of Pattyn's model with Glimmer presented numerous software engineering challenges. The model was written as a stand-alone Fortran program. Pattyn wrote his code in an antiquated version of the language: Fortran-77 instead of the Fortran-90 language that CISM is written in. Parameters such as grid size and spacing, physical constants, and numerical tolerances were stored as global variables at the start this single file of source code. Rather than reading in data for experiments, the model had some simple experiments hard-coded at the beginning of the executable program. This contrasts with Glimmer's method of model setup, which is to read model parameters and options from a text-based configuration file and input data such as the model geometry from a NetCDF file; nothing is hard-coded into the model its self.

The two models additionally made different assumptions regarding the nature of the numerical field data, and these differences presented deeper concerns. Pattyn's model used a different dimension ordering than Glimmer: whereas Glimmer used the NetCDF standard of addressing two-dimensional arrays as (x, y) and three-dimensional arrays as (z, x, y), Pattyn's model addressed two-dimensional arrays as (y, x) and three-dimensional arrays as (y, x, z). Additionally, Pattyn's diagnostic model does not share Glimmer's concept of multiple grids. In Glimmer, velocities and associated quantities such as diffusivities are computed at the centroids of the grid spaces, or on what is referred to as the staggered grid or "velocity grid" [Hag-

dorn et al., 2006]. Values of a function specified on one grid can be approximated by averaging surrounding values from the other: $p_{i+1/2,j+1/2}$ is the average of the points $p_{i,j}, p_{i,j+1}, p_{i+1,j}, p_{i+1,j+1}$, shown in Figure 4.1. In contrast, Pattyn's diagnostic model co-located the ice geometry values and the velocity values (although Pattyn does place the velocity values onto a staggered grid in an more ad-hoc way while solving the ice equation [Pattyn, 2003]).

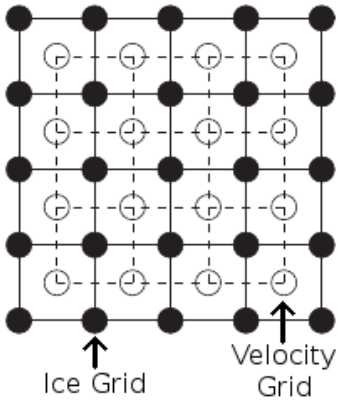

Figure 4.1 Points on the staggered grid are located at the centroids of the nonstaggered grid, and can be thought of as the averages of the nonstaggered points. From the Glimmer documentation [Hagdorn et al., 2006].

Pattyn's model required infrastructure that was only partially present in Glimmer. Certain model components, particularly those related to thermomechanical coupling, were missing from the model; the required fields were present, but were filled in with constant values. An ice transport scheme was present, but was based on a simple model and was known not to conserve mass and required very small time steps [Pattyn, 2003]. The sparse matrix solver used by Pattyn (namely, a diagonal-preconditioned biconjugate gradient method copied from Press et al. [1992]) was ill-suited to the problem and not compatible with Glimmer's use of the GPL; for these

reasons, and because of the desire to use direct and parallel solvers with Glimmer, it was necessary to build components to allow a more diverse selection of sparse matrix solvers. Finally, many of the functions required by Pattyn's model, such as numeric differentiation, were also present in Glimmer in some form but were not generic enough to be used outside of the modules for which they were originally intended. The code duplication caused by introducing Pattyn's redundant components was undesirable, and these components reduced the cohesion of Pattyn's model by introducing concerns not directly related to the computation of higher-order velocities. These components required a re-engineering effort to ensure that these functions were available to both models without these drawbacks.

Interestingly, during my work on Pattyn's higher-order component, an effort to integrate a similar model was being performed at Los Alamos National Laboratories. Though this required additional attention to make sure that both systems were integrated cleanly and uniformly, it also helped both parties solve common problems that resulted in the integration more quickly and effectively and drove the development of the software's architecture with a greater eye towards extendibility than would have otherwise been developed.

In this chapter, I will describe the software development process and engineering decisions that addressed these challenges and opportunities.

4.1 Code Structure

Fortran-90 does not support true object oriented programming: the closest it comes is the addition of C-style data structures, but this lacks even the basic ability to declare certain members as private. Although the Fortran 2003 standard does support the object-oriented paradigm, few compilers implement the entire standard, so these

constructs were not used. On the other hand, the problem at hand, the integration of several components that act as alternatives to one another is a problem that lends itself very well to object-oriented design patterns. Therefore, a pseudo object-oriented approach was used to design the additions to Glimmer that supported the new higher-order physics. This means that concepts and best practices from object oriented programming were used during the design, if not the implementation, of the new modules.

In general, because the additions to Glimmer represented different physical approximations or numerical algorithms used to solve the same set of equations, I employed the strategy pattern described by Gamma et al. [1995] as a design pattern that "defines a family of algorithms, encapsulates each one, and makes them interchangeable. Strategy lets the algorithm vary independently from clients that use it" [Gamma et al., 1995]. In other words, the strategy pattern defines a standard interface that client code can use to solve a problem and creates multiple alternate algorithms that solve the problem and conform to that interface. The standard interface is specified in a base class from which classes implementing algorithms derive, thus allowing client code to switch between algorithms transparently.

This design pattern was employed in three places in the code. The first was the selection of an ice transport scheme between the old diffusive scheme in Glimmer and the new incremental remapping scheme described by Lipscomb and Hunke [2004] and integrated by LANL. The second was the decision between two alternate sets of higher-order velocity solvers: those from Pattyn [2003] and those from an unpublished model from Payne and Price. Note here the natural partitioning of the dynamical core into the diagnostic portion (solving for a velocity field) and the prognostic portion (using the solved velocities to evolve the ice geometry). Finally, I employed the strategy pattern in selecting between

algorithms to solve the sparse linear system arising from the elliptical Stokes approximation (in this case, not all options are available depending on compiler options). A "conceptual class diagram", showing how this pattern would be implemented were object-oriented constructs available, is shown in figure 4.2.

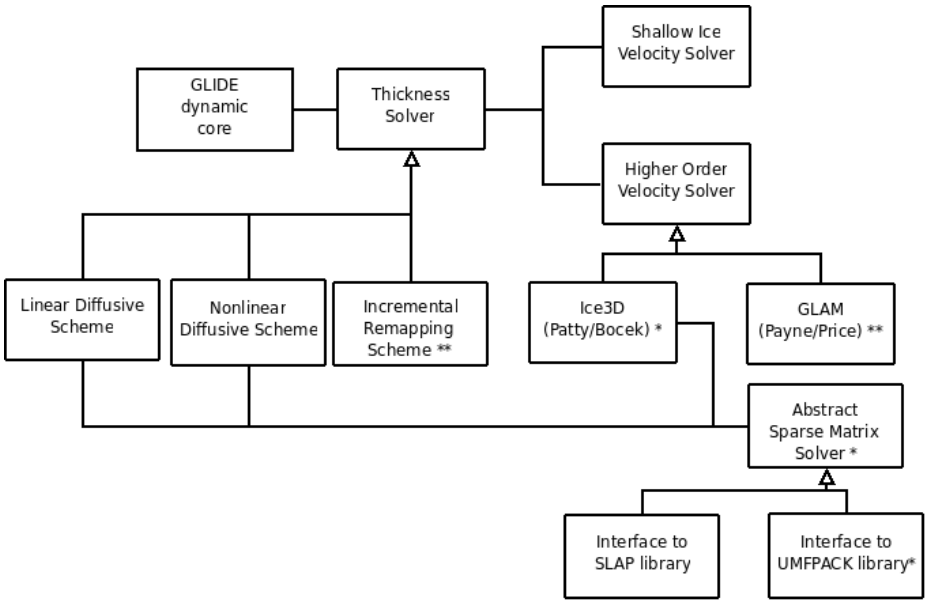

Figure 4.2 The conceptual, object-oriented static code structure of the additions to Glimmer. Conventions of a class diagram are followed, though names are high-level and not representative of the resulting software. (* Added as part of this thesis) (** Added by parallel effort at LANL)

I implemented the strategy pattern in the non-object-oriented Fortran-90 language as follows. First, I defined a data structure that held the data members required by each strategy; in many cases, there was sufficient overlap so that there was little memory in each case that was allocated but unused. This is to be expected, as the strategy pattern encapsulates different algorithms for operating on data structures rather than different methods of storing the data. Second, I defined a common interface that code requiring a diagnostic, prognostic, or sparse matrix solver would use

to call out to the chosen strategy. Third, I created a high-level module that took the place of the abstract base class, and an additional module for each different strategy implementation. Each module implemented a function with the defined signature as its entry point. Code that needed the algorithm would simply call the function in the abstract base class; the abstract version of the call was little more than a switch on a flag read at run time from the supplied configuration file. Thus, like the strategy pattern, the switching between algorithms is mostly transparent to the calling code, which is burdened only with the need to keep track of an additional flag specifying which strategy should be used. This benefit was made possible by the common interface. Figure 4.3 shows the F90 implementation of the conceptual static structure in figure 4.2.

Notice that the shallow ice velocity solver is considered a separate concern from the higher-order velocity solver. Indeed, they occupy separate derived types (`glide_velocity` versus `glide_velocity_hom`), even though these types need to contain several of the same fields (such as u, v, and w components of velocity, fluxes, etc.). Indeed, two data structures were created instead of one because the higher-order physics require many more data fields – a derived class would have been preferable, but F90 does not support this construct. Because the SIA method and the higher-order methods are all methods of computing the velocity, however, an object-oriented approach would require that all of these methods be implemented within derived classes based on a single velocity solver base class. It is therefore worth discussing the reasons why this was not done.

There are essentially four reasons for this, most of them deriving from the fact that a higher-order velocity solution takes several orders of magnitude longer to compute than a shallow ice velocity solution. First, during early stages of development, it was desired to keep higher-order velocity separate so that it did not break production code

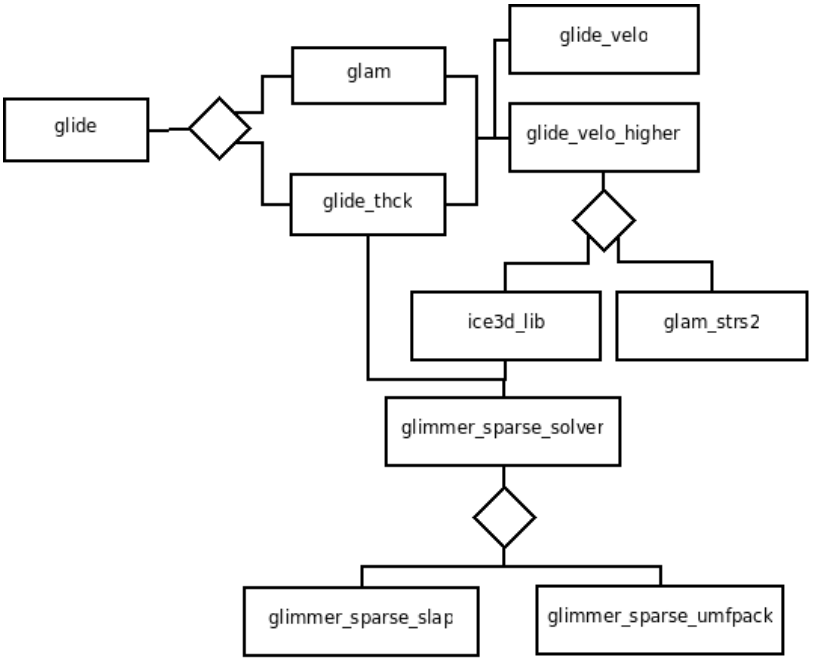

Figure 4.3 The static code structure shown in figure 4.2, translated to the idioms available in Fortran-90. Diamonds are analogous to class inheritance, as they represent instances where one module will call only one other module in the relation at runtime, depending on the configuration of the run.

(intelligent use of branching in our version control system could have accomplished the same thing, but this was not done at the time). Second, for non-technical reasons we desired the ability to create a version of CISM with the higher-order code stripped out. Third, we thought it would be useful to be able to run higher-order computations diagnostically alongside an ice sheet that was evolving according to shallow ice physics. Finally, we wanted to support emerging schemes that used a combination of shallow ice and shallow shelf physics, such as the hybrid SIA-SSA model from Bueler and Brown [2009]. All of these requirements pointed to a methodology of keeping shallow ice operations "unpolluted" with higher-order operations.

Not shown on the module diagram are a number of modules added to provide lower-level numerical routines to both Glimmer and Pattyn's model. An example of this is numeric differentiation, which was previously handled by routines in `ice3d` and in `glide_thck`. This reduced the software cohesion of these models, as numeric differentiation is not *directly* related to either thickness evolution or higher-order solving. If other modules needed access to these routines, they would have needed to import an inappropriate module, increasing the amount of coupling in the system, or write their own, increasing the amount of code duplication. Therefore, the numeric differentiation routines were re-written to be more general, moved into a `glide_derivs` module whose only purpose was to provide these routines to both `ice3d` and `glide_thck`, and removed from the modules in which they originally resided. The same was done to algorithms for staggering, handling masks, and handling periodic boundary conditions, among other examples. The result is that not only is the integration cleaner, but also code that has nothing to do with the higher-order integration has been improved and, in general, CISM is a more welcoming framework for other integration efforts.

4.2 Engineering Process

I performed the actual integration of Pattyn's Fortran-77 model with the Fortran-90 Glimmer code in a multiphase, iterative prototyping process. In general, this process consisted of working first on integrating components and upgrading physics or numerics using an evolutionary, agile approach first, resulting in prototype code that I then refined using more structured software engineering techniques.

4.2.1 Preparation of Standalone Code

A good deal of the integration effort was performed before Pattyn's model was even placed in CISM's build environment. At this stage, I prepared A Fortran-90 version of the model suitable for integrating with Glimmer. I first wrote and used an automatic preprocessor to convert the F77 fixed-form syntax into F90 free-form syntax by changing the comment style and line continuation style. The script did not attempt to apply any deeper syntactic updates, as it was determined that doing so would take more time to develop than applying the updates manually. Next, Pattyn's model was updated to use Fortran-90 idioms rather than those of Fortran-77. During this step I adopted best practices that emerged with the newer F90 syntax. These changes included changing F77-style DO...CONTINUE loops to F90-style DO...END DO loops, using implicit none, using assumed array shapes rather than explicit array shapes, and declaring intents. This was an important step in making Pattyn's code more familiar, and thus more readable and maintainable, to future CISM developers. It should theoretically have had no impact on model output; in practice, I saw differences on the order of $10^{-6}\%$ which I can only attribute to differences in the way the F77 and F90 compilers used handled floating-point precision.

Finally, I imported a subset of Glimmer's modules into the project and updated

Pattyn's model to use them. Chiefly, Pattyn's ad-hoc derivative and periodic boundary condition routines were replaced by routines in a new Glimmer module developed for the integration effort. Other minor changes included the use of Glimmer's physical constants and precision types. At this stage, I also gave the modified model a staggered version of the geometry to verify that the model produced reasonable results, as I knew that this would be required of it later in the integration process.

4.2.2 Initial Diagnostic Integration

During this stage, I integrated Pattyn's diagnostic model with Glimmer using any means necessary. As a result, the code at this phase was highly prototypical. The main effort during this stage was the creation of a facade in glide_veo_higher to address the data incompatibility concerns. A facade is another object-oriented design pattern described by Gamma et al. [1995] that "provides a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use" [Gamma et al., 1995]. In the case of the higher-order wrapper, the complex subsystem was the series of calls needed to compute higher-order velocities, including the data transpositions needed. The facade implementation created temporary versions of variables that transposed and staggered the data, making it suitable for passing to Pattyn's functions. The facade code also translated the results of Pattyn's computations back to CISM's native format. Any inefficiency incurred by these operations was overshadowed by the length of the higher-order computation itself. Some of these operations remained in the final version of the integration, others were rendered obsolete by additional engineering efforts. Wrapping these operations in a high-level facade avoided polluting client code with the details of calling Pattyn's model.

4.2.3 Physics and Numerics Refinement

During this phase of development, I refined the initially rough integration by putting into place the rest of the infrastructure developed for the integration and by implementing additional physical requirements. I put into place a first version of the modules and logic that allowed switching between sparse matrix solver packages, though due to a poor understanding of the requirements of this subsystem the solver algorithm could only be switched at compile time. The other main effort at this stage was the addition of ice shelf physics, which Pattyn's model lacked at the time of integration. Software design changes required by the updates were still done in a more prototypical fashion.

4.2.4 Software Design Refinement

During this phase, I refactored the prototype code from the previous phases, greatly improving the overall software design. The compile-time sparse solver option was changed to a runtime option, improving flexibility and allowing for the use of a direct solver as a fall-back if an iterative solve failed. Pattyn's code was changed to run natively using CISM's coordinate system, obviating the need to create transposed versions of fields used by Pattyn's model. Though this was a major effort, the transposed coordinate system was a source of many subtle bugs during the integration, and future maintainers used to CISM's native coordinate system would likely be confused when reading Pattyn's code. In addition to greatly improving the models maintainability, the change allowed even more general-purpose routines in the higher-order model that were sensitive to coordinate ordering to be pulled into their own modules. Like the similar treatment of numeric differentiation, this avoided code duplication and clutter while improving the cohesion in Pattyn's model. Finally, during this phase LANL's

Payne/Price model integration and the Pattyn model integration were redesigned and standardized to use the same strategy-class-like design previously described.

4.2.5 Initial Prognostic Integration

All of the work described above was in support of the diagnostic computation: computing a field of ice velocities within the ice sheet. CISM lacked, however, the ability to use higher-order diagnostic velocities to solve the *prognostic* equation, which uses those velocities to compute how the thickness and extent of an ice sheet evolves over time. Glimmer contained the ability to evolve an ice sheet based on computing shallow ice velocities and assuming that ice primarily diffused across a continent. This diffusive scheme, however, is not suitable for use with higher-order velocities. To ameliorate this, the model was integrated with an incremental remapping scheme, described by Dukowicz and Baumgardner [2000] and Lipscomb and Hunke [2004] and developed at LANL. This allowed the higher-order velocities to evolve the geometry of the ice. Again, this early engineering effort was done by any means necessary, resulting in highly prototypical code. Once again, I employed the strategy pattern to allow transparent switching between the new prognostic solver and Glimmer's old diffusive scheme. At the time of this writing, this integration effort is still under testing. After it is complete, another design refinement phase will be carried out.

4.3 Testing Process

Because both the new higher-order model and the original shallow ice model are relatively complex to maintain, a suite of regression tests was created and run throughout the integration process. The benefits of regression testing are well-understood in software engineering: as software projects grow in complexity, it is difficult to ac-

curately assess the ripple effect of changes to software components across the entire project. It is therefore desirable to develop a suite of tests that can be run quickly to build confidence that code changes have not caused "regression", or a reduction in working functionality. These tests are run frequently so that if a regression does occur, it is easy to pinpoint the small set of changes that may have caused it. (A discussion of regression testing can be found in most any software engineering textbook, e.g. Pressman and Ince [2005].)

I chiefly tested CISM using the EISMINT-2 tests, described by Huybrechts et al. [1996], and the ISMIP-HOM tests, described by Pattyn et al. [2008] and detailed in the next chapter. I opted to use a black-box testing approach rather than the white-box unit test approach because the components in scientific code tend to be tightly coupled and therefore difficult to test in isolation. In addition, the existing Glimmer code did not have requirements defined at a level of granularity that made unit testing feasible.

Before the integration began, I used Pattyn's original model to run ISMIP-HOM A and C. During the pre-integration work, the updated F90 version of the model was periodically used to run the same tests, with output compared numerically to the original. Due to the complexity of numerical calculations and the opportunity for roundoff errors to slightly change results, identical results were not expected. Rather, I accepted the test results if they were within some predefined error tolerance when comparing the original results and the most recent results from the updated model.

While integrating the higher-order model with CISM, I switched to a subset of the EISMINT-2 tests, which I ran using the shallow ice computations in order to build confidence that the integration process introduced no regression in the existing capabilities of the model. Again, I compared the results to those from an unmodified version of Glimmer to determine whether to accept the new version of the model.

Once the integration was completed to the point that I could receive output from the higher-order computation running within CISM, I constructed a more full-featured ISMIP-HOM test script to verify that the integrated higher-order model worked correctly. The test script also helped check for regression introduced by new features added during concurrent development efforts. In this iteration of testing, model output was checked not only against that of previous versions of the same model but also against the output of other models running the same experiments. This intercomparison methodology is described in the next chapter. Both the test suite and the comparison were scripted in Python to allow an entire ISMIP-HOM test, or some subset thereof, to be run with no intervention; this same framework could be extended to support EISMINT testing as well.

Finally, despite the difficulties of unit testing that were previously described, we did write and perform unit tests on the subset of components developed as part of the integration that could be tested in isolation, such as matrix assembly and numeric differentiation.

CHAPTER 5 MODEL VERIFICATION

In Chapter 4, I discussed the use of well-known ice sheet experiments in a software engineering context: making sure that changes to software do not undermine existing features. In this chapter, I expand on that discussion and present a number of experiments that I used to build confidence in both Patty's original model and the correctness of its integration with CISM. During this effort, I ran a number of experiments that provided comparisons to both existing models and to exact solutions. The model was also validated through some comparison to observed data, though the opportunity to do this was limited.

Though very common in the ice sheet modelling community, the methodology of intercomparison comes under scrutiny often enough that it worth taking a moment to address concerns regarding it. Model intercomparison is one tactic used for to *verify* a new ice sheet model - that is, it is used to build confidence that the output of the model under scrutiny corresponds to ice sheet physics as we currently understand them. In addition to intercomparisons, a handful of exact solutions exist which we can also compare to. Ice sheet modelers freely admit, however, that such experiments cannot *validate* our models - that is, ensure that the underlying physics that they implement reflect physical data. Intercomparison is prevalent in the ice sheet modelling community because data to compare to is difficult to come by, is often incomplete, and usually requires models to solve the ice sheet equations over more difficult domains.

5.1 ISMIP-HOM Experiments

The first verification suite that I ran was a subset of the Ice Sheet Model Intercomparison Project for Higher Order Models (ISMIP-HOM) test suite described in Pattyn et al. [2008]. The goal of ISMIP-HOM is to facilitate basic verification of ice sheet models that implement higher-order physics by providing experiments that require the use of higher-order stresses to properly solve. A wealth of model output data in a standardized format has been published alongside the experiment descriptions [Pattyn et al., 2008], allowing modelers to check their work against that of others building similar models. Although these tests were previously performed by Pattyn using the standalone version of his model, my version of Pattyn's model represents a significant amount of software engineering. I therefore performed the tests again to verify that the model was correctly integrated into CISM, and that no regression occurred as a result of Pattyn's translation of his published model from C to F77 and our translation from F77 to F90.

5.1.1 Description of Experiments

The subset of ISMIP-HOM that I ran consisted of the first four experiments (two experiments are neglected due to relying on the presence of a prognostic solver, which we did not have in place at the time). Each experiment is described in terms of a rescaled horizontal coordinate system, with

$$\begin{aligned} \hat{x} &= \frac{x}{L} \\ \hat{y} &= \frac{y}{L}. \end{aligned} \tag{5.1}$$

This allows the experiments to be run on geometrically similar geometries for domain sizes of L = 5, 10, 20, 40, 80, and 160 km.

All experiments are run with periodic boundary conditions - that is, the domain is assumed to be a small portion of an infinite domain that is periodically similar. For these experiments, this has the advantage of avoiding the need to specify lateral boundary conditions of the ice sheet. The ISMIP-HOM experiment descriptions Pattyn and Payne [2006] recommend a simple and naive implementation of these boundary conditions: domains are specified with a layer of "ghost cells" on the edges. These ghost cells are not part of the computation, but rather represent the start of the next repetition of the domain. During each nonlinear iteration, the ghost cells are filled in by copying values from the computational domain. This method requires iteration, but as it is used in the context of a computation that is already iterative it is not seen as a disadvantage.

Note that this copy operation is valid even on along the axis perpendicular to the slope of the surface and bed. Although the elevations differ between the two edges, the copying does not cause a discontinuity because the elevations themselves do not enter into the ice sheet equations and the elevation gradients remain consistent.

While two of the experiments are specified as three-dimensional, two are specified as two-dimensional domains consisting of one horizontal and one vertical dimension so as to allow flowline models to participate in the intercomparison. For the two flowline experiments, the domain was extended the second horizontal dimension with no across-flow variation; a more complete discussion can be found in Section **??**.

The first two experiments exercise the handling of higher-order stresses that appear due to variations in the bed geometry. The ISMIP-HOM A experiment consists of a three-dimensional ice sheet with a uniformly sloping surface and a bed that varies sinusoidally in two dimensions [Pattyn et al., 2008],

$$z_s(x, y) = -x \cdot \tan \alpha \tag{5.2}$$

$$z_b(x, y) = z_s(x, y) - 1000 + 500 \sin(2\pi\hat{x}) \sin(2\pi\hat{y}) \tag{5.3}$$

where the surface slope $\alpha = 0.5°$.

ISMIP-HOM B consists of a sinusoidally varying bed in one dimension to facilitate participation by flowline models. Here z_s is as in ISMIP-HOM A, and the basal geometry is

$$z_b(x, y) = z_s(x, y) - 1000 + 500 \sin(2\pi\hat{x}). \tag{5.4}$$

In both experiments, velocity at the bed is specified as a zero-valued Dirichlet condition, corresponding to ice that is frozen to the bed.

The second two experiments exercise the handling of varying basal boundary conditions, a necessary component of modeling ice streams. In both experiments, the ice thickness is specified as a uniform 1 km with a uniform slope in both the surface and base elevation of the ice sheet:

$$z_s(x, y) = -x \cdot \tan \alpha \tag{5.5}$$

$$z_b(x, y) = z_s(x, y) - 1000 \tag{5.6}$$

Here, $\alpha = 0.1°$ rather than $0.5°$.

The basal boundary condition specified as a hard (linear) bed with a sinusoidally varying coefficient of friction (β^2). Analogously to the three- and two-dimensional experiments A and B, ISMIP-HOM C specifies that the sinusoidal variation occurs

in two dimensions while ISMIP-HOM D specifies that the sinusoidal variation occurs in only one, again to allow flowline models to participate.

For ISMIP-HOM C:

$$\beta^2 = 1000 + 1000 \sin(2\pi\hat{x}) \sin(2\pi\hat{y}) \tag{5.7}$$

For ISMIP-HOM D:

$$\beta^2 = 1000 + 1000 \sin(2\pi\hat{x}) \tag{5.8}$$

All experiments were run on a 40x40x40 grid, with an exception for flowline experiments discussed below. Given the hardware available this was the best compromise between computational tractability and model accuracy. All computations were done on CISM's staggered velocity grid rather than its ice grid. Vertical grid spacing in all cases was even; though Pattyn's model has the ability to use a nonuniform spacing in the vertical we did not take advantage of it for these runs.

5.1.1.1 Approximating Flowline Experiments

Special care must be taken when approximating two-dimensional flowline experiments such as ISMIP-HOM B and D with a three-dimensional higher-order model. This is done in theory by enabling periodic boundary conditions in the direction perpendicular to the flow. Because computational nodes in the perpendicular dimension theoretically contain redundant data, I reduced the number of these nodes.

Problems arise, however, as a result of the periodic boundary update. Rather than building the sparse matrix so as to reflect the presence of periodic boundary conditions, this model enforces these boundary conditions by performing a ghost cell update after velocity computations are completed. This copy operation can introduce

artificially large gradients at the boundaries of the computational domain. For most experiments, this is not a problem, as the gradients "quiet down" after a number of nonlinear iterations. When running a flowline experiment, however, there are no naturally occurring gradients in the direction perpendicular to flow. Therefore, rather than reducing in magnitude as the natural gradients begin to dominate, these artificial gradients produce oscillations at the boundaries that neither decrease nor increase in magnitude. The solution is simple: maintain the same number of grid cells in the perpendicular direction while increasing the size of the grid spacing, effectively reducing the magnitude of the artificial gradients. A properly chosen grid spacing will stabilize the iteration at no computational cost.

5.1.2 Results

Although there are some deviations, my integration of Pattyn's model compares somewhat favorably to the other models submitted in the ISMIP-HOM intercomparison. In general, our results tend to be biased towards high velocities compared to other first-order and full-stokes models. However, for most experiments we have results that are within one standard deviation of the first-order models submitted to the intercomparison.

For each experiment, we provide a visual of the flowline results of the experiment, after the style of results presented in Pattyn et al. [2008]. We also present a table of the maximum observed error, as well as whether the point at which that error was observed was within one standard deviation of the mean.

5.1.2.1 ISMIP-HOM A

Figure 5.1 presents the results of this experiment visually. Per the ISMIP-HOM experiment specification, the velocity plotted is a flowline extracted along the direction

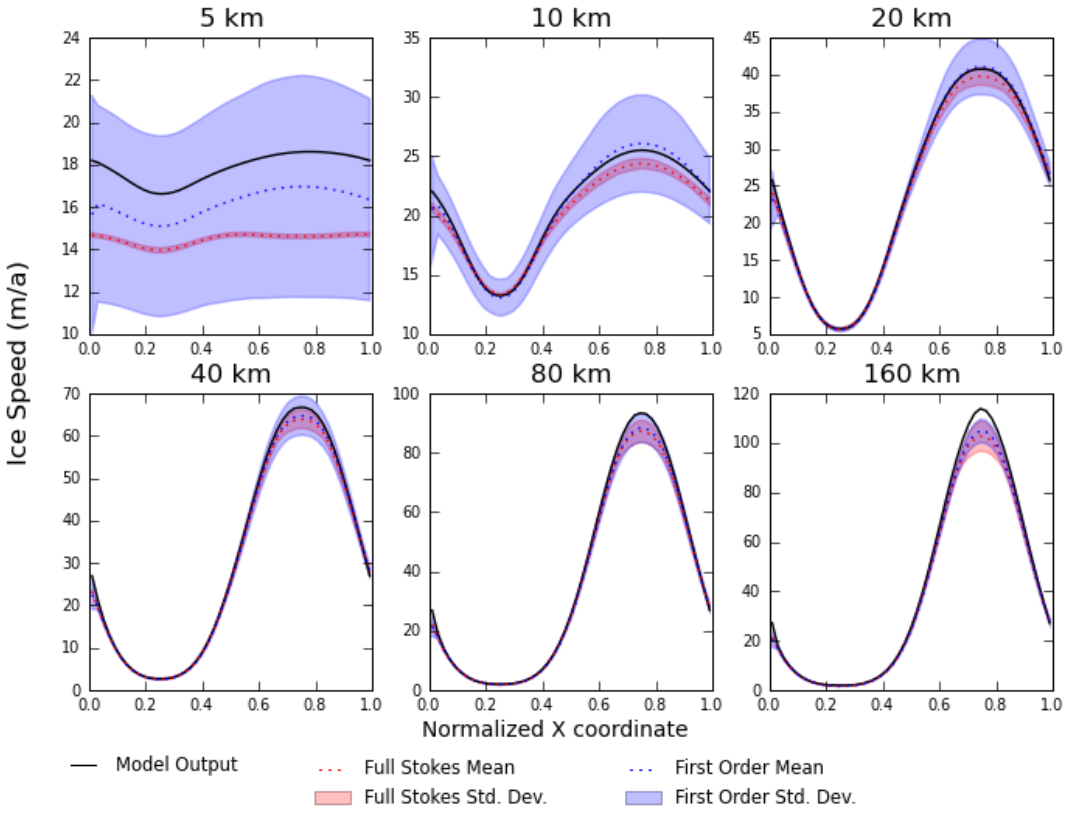

Figure 5.1 Results for ISMIP-HOM A

of the surface slope at $\hat{L} = .25$. A numerical comparison using MSE is presented in Table 5.1. Qualitatively, our model matches the mean outputs of first-order models very closely. The velocity profile in all cases is similar to that of the mean. However, the although the qualitative shape is the same in some cases the velocity magnitude is not correct. This is particularly true of domain lengths of 5 km, 80 km, and 160 km, where the velocity (particularly the maximum observed velocity) is higher than the mean of submitted models. In all cases except for the 160 km domain length, our model output is consistently within 1 standard deviation of the mean of first-order models submitted to the intercomparison.

5.1.2.2 ISMIP-HOM B

Similar to ISMIP-HOM A, our ISMIP-HOM B results are qualitatively similar. We reproduce several distinctive features observed in the mean of the model outputs, including the retrograde profile for L = 5 km observed between first order models and full Stokes models. We also reproduce the flattening of the velocity profile observed particularly when L = 10 km and L = 20 km. This is only observed in the flowline version of the experiment, and is probably due to the lack of lateral stresses in the flowline scenario compared with the three-dimensional scenario. Like the ISMIP-HOM A results, the ISMIP-HOM B results overestimate the maximum velocity at the longer domain lengths. However, this overestimation is less severe, with our model output still within one standard deviation of the first order model mean.

5.1.2.3 ISMIP-HOM C

Compared to experiments that determine the effect of variations in bed topography on velocity, experiments that determine the effect of variations in bed friction converge very slowly. Short domain lengths in particular can take upwards of 500 un-

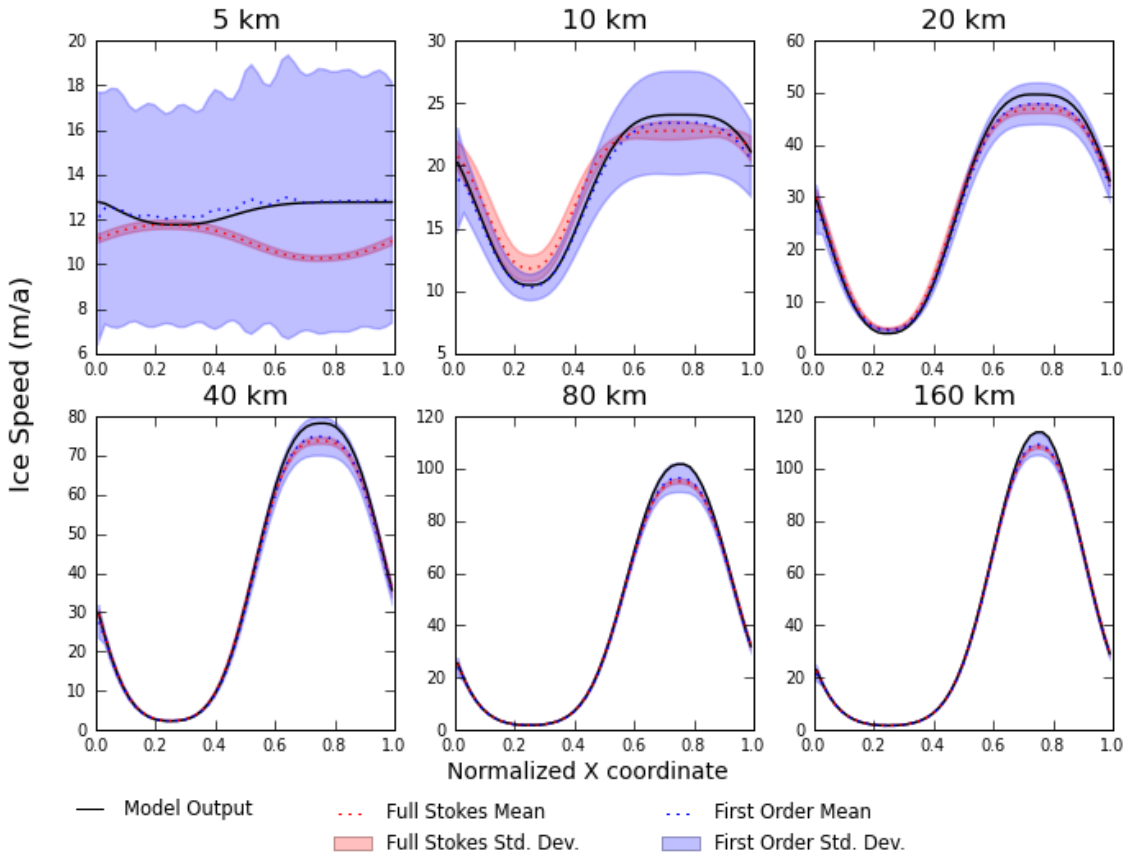

Figure 5.2 Results for ISMIP-HOM B

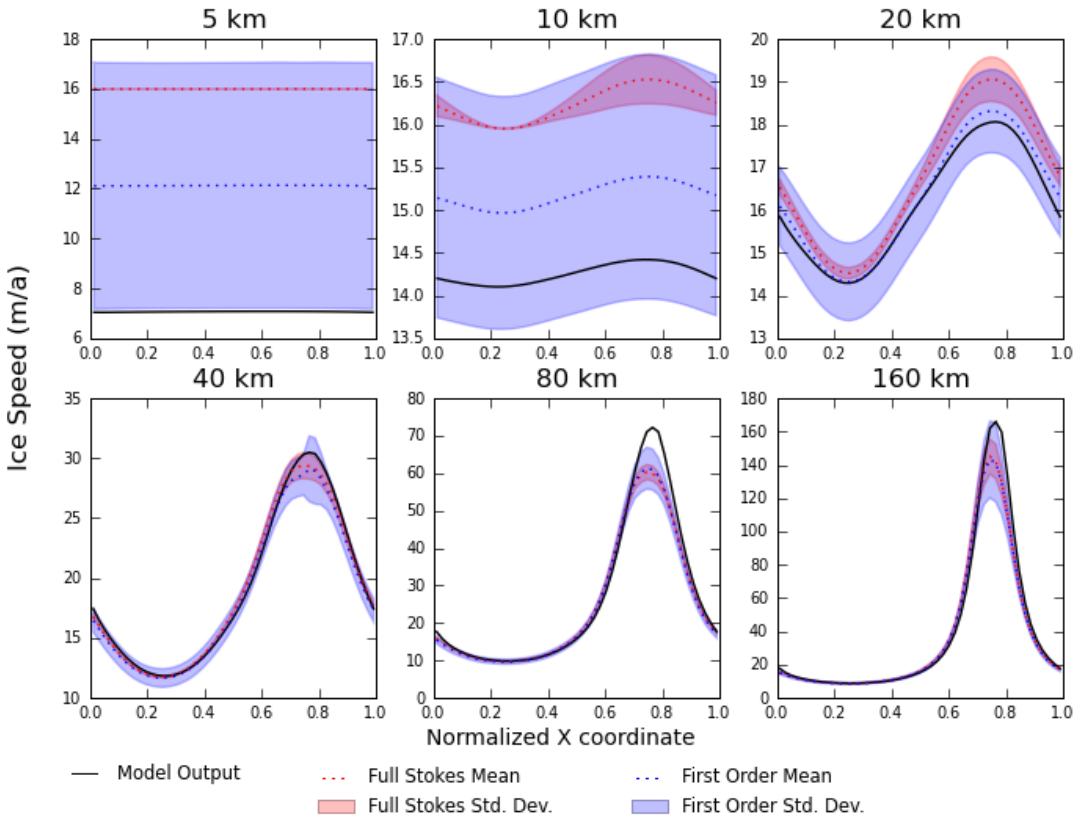

Figure 5.3 Results for ISMIP-HOM C

stable manifold iterations to converge, compared to 30-50 iterations needed for typical ISMIP-HOM A and B experiments. Additionally, the slow convergence means that the quality of the solution is much more dependent on the error tolerance used.

Our ISMIP-HOM C results accordingly are poor compared to our ISMIP-HOM A and B results. Although our velocity profiles, compared to those of models submitted to the intercomparison, are qualitatively similar, the deviations are much more severe. For domain sizes less than 20 km, the velocity is underestimated. The velocities are barely within one standard deviation when L = 10 km; when L = 5 km, the velocity is just outside one standard deviation. Conversely, for domain sizes greater than 20 km, the velocities are overestimated. This is analogous to our results for ISMIP-HOM A, though more exaggerated. In addition, the maximum velocity in our model output is offset slightly.

5.1.3 ISMIP-HOM D

At the time of this writing, I have not successfully run Pattyn's model on ISMIP-HOM D for domain lengths greater than 10 km. The solver quickly converges to a partial solution, but fails to reach the specified error tolerance as a result of gradients in the across-flow component that change in qualitative nature without reducing in magnitude as the problem iterates. The shorter domains produce reasonable (within one standard deviation) results, though a numerical comparison has not yet been made. I do not, however, see this failing of Pattyn's model as a problem. I have been equally unsuccessful with these computations using his standalone version of the model. As described earlier, running a flowline experiment with a three-dimensional model is tricky, as the required periodic boundary conditions produce artificial gradients that dominate the solution. I therefore postulate that I have simply not found across-flow grid parameters that work well for this experiment. It could be, however,

that the problem is a more serious error in the experimental setup, causing a well-posed two-dimensional problem to become ill-posed in three dimensions (this poor iterative behaviour has, in my experience, been indicative of an ill-posed problem). At any rate, as all of the practical applications of an ice model are three-dimensional, difficulty in running flowline experiments does not significantly reduce the model's utility.

5.1.4 Effects of Grid Selection

A characteristic of numerical models that is often discussed is whether the model's performance is dependent on grid refinement. The ISMIP-HOM experiments were run on a diverse selection of grids in order to determine this dependence. Intuitively, one would expect that increasing the grid resolution in any dimension would improve the accuracy of the solution. However, this is not necessarily the case!

To determine the effect of grid selection on how well the solution agrees with other models, I ran ISMIP-HOM A for a variety of grid size selections. I compared the model output to the mean output of non-full-Stokes models by computing the mean squared error (MSE). The results are summarized in Table 5.1.

The data here do not paint a clear picture. For some domain sizes, namely 5km, 80km, and 160km, increasing the horizontal grid spacing without changing the vertical grid spacing appears to *decrease* the accuracy of the model! The situation, however, is opposite for 20km, 20km, and 40km. In all cases, decreasing the number of vertical grid nodes from 40 to 20 led to a decrease in accuracy, but an increase from 40 to 60 only increased accuracy for domain sizes L $>$ 40 km, and an increase from 60 to 80 only increased accuracy for domain sizes L $>$ 80 km. These suggest that there is at least some dependence on aspect ratio rather than simply on grid resolution, but it is not a simple relationship and there are likely other factors involved.

	5 km	10 km	20 km	40 km	80 km	160 km
± 1 std. dev.	22.4	8.82	4.72	5.72	5.21	4.13
20x20x40	0.314	2.52	2.72	1.24	0.453	0.627
40x40x40	1.08	1.10	0.197	0.407	2.38	**7.48**
60x60x40	2.09	1.08	0.0693	0.777	4.81	**12.9**
40x40x20	**315**	**15.5**	**6.85**	**18.6**	**5.34**	**15.1**
40x40x60	4.78	2.08	0.469	0.0698	0.308	0.848
40x40x80	6.96	2.45	0.626	0.0787	0.0825	0.152

Table 5.1 Comparison of the MSE observed in ISMIP-HOM A on a variety of grids with the MSE expected if the model produced an output that was exactly one standard deviation away from the mean across all models in the intercomparison. Bolded entries are on average worse than one standard deviation from the mean.

The results for a similar experiment using ISMIP-HOM C are presented in Table 5.2. Unfortunately, the selection of grid sizes here is limited by the need to "spin up" an ISMIP-HOM C solve with a direct sparse solve in the first few iterations. Here the trend is somewhat more clear: For most domain sizes, refinement of the horizontal grid is beneficial (curiously, though, for L = 5 km a coarser horizontal grid works better). Both refinement and coarsening of the vertical grid while holding the horizontal grid size constant leads in most cases to a worse solution. This suggests again that there is some aspect ratio dependence.

5.2 Idealized ice shelf experiments

While ISMIP-HOM A-D test the components of the model required for diagnostics of land ice velocities, they fail to exercise the ability of the model to handle floating ice. As this was a major component added beyond the capability of Pattyn's original model, it is important to rigorously verify its performance. In order to verify the correctness of the ice shelf physics with regard to the mathematical model, we

	5 km	10 km	20 km	40 km	80 km	160 km
± 1 std. dev.	16.5	0.844	0.326	0.363	0.435	0.625
40x40x40	11.6	0.370	0.171	0.611	0.778	0.930
20x20x40	3.13	0.0526	0.287	0.542	1.21	3.14
60x60x40	16.5	0.381	0.00282	0.0620	0.450	0.795
40x40x20	11.8	0.383	0.275	0.425	0.820	1.01
40x40x60	11.5	0.381	0.179	0.571	0.779	0.929

Table 5.2 Comparison of the MSE observed in ISMIP-HOM C on a variety of grids with the MSE expected if the model produced an output that was exactly one standard deviation away from the mean across all models in the intercomparison. Bolded entries are on average worse than one standard deviation from the mean.

compared the CISM model output to an exact solution for the 1D velocity profile of a flowline through an unconfined ice shelf, developed by Weertman [1957]. The analytical solution for strain rate is

$$\dot{\epsilon}_{xx} = \frac{d\bar{v}}{dx} = A\left(\frac{\rho_i g H}{4}\left(1 - \frac{\rho_i}{\rho_w}\right)\right)^n. \tag{5.9}$$

The domain is specified such that one end is the grounding line and the other end is the calving front. A Dirichlet boundary condition specifies the flux exactly at the grounding line. Because the entirety of the domain is an ice shelf, note that the assumption of plug flow means that there is no vertical dependence in the solution; this is emphasized with the use of average velocity in the strain rate definition.

5.2.1 Constant thickness

As an initial experiment, we held the thickness to be constant $H = 1000$ meters. The ice shelf was assumed to be isothermal, using the EISMINT-Ross value of $A = 4.6 \times 10^{-18}$. In these cases, there is no horizontal dependence in either A or H.

Therefore, Weertman's solution can be easily integrated, giving

$$\bar{v}(x) = A\left(\frac{\rho_i g H}{4}\left(1 - \frac{\rho_i}{\rho_w}\right)\right)^n x + \bar{v}_0. \tag{5.10}$$

Using the values of the constants specified above, specifying zero flux at the grounding line ($v_0 = 0$), and with x in meters, this becomes

$$\bar{v}(x) = (0.077098679482272892\ s^{-1})x. \tag{5.11}$$

As is the case of ISMIP-HOM B and D, a flowline experiment was simulated by reducing the size of the domain perpendicular to the direction of flow and enabling periodic boundary conditions on the lateral edges of the domain. Although this problem converges slowly, our model output matches the analytical solution to .3026% just after the grounding line to .3047% at the shelf front. As these are the minimum and maximum observed error, our model matches the analytical flowline velocity very closely.

5.2.2 Van der Veen Ice Tongue

We additionally tested our model against the steady-state ice shelf geometry derived by Van der Veen [1986]. We assuming a nonzero input flux q_0, and use the dimensional rescalings

$$u = Uu'$$

$$x = Lx'$$

$$h = Zh'$$

$$2\left(\frac{U}{AL}\right)^{\frac{1}{n}} \frac{Z}{L} = \frac{\rho_i g Z^2}{2L}.\left(1 - \frac{\rho_i}{\rho_w}\right)$$

Van der Veen found the steady-state ice profile given $q_0, h_0 \neq 0$ to be

$$H'(x) = \left[1 + \frac{q_0'^{n+1}(H_0'^{-(n+1)} - 1)}{(q_0' + x)^{n+1}}\right]^{\frac{-1}{n+1}}. \tag{5.12}$$

For this experiment, rather than the EISMINT-Ross constants, we use the constants specified for the EISMINT-1 experiments [Huybrechts et al., 1996], shown in Table 5.3. Under these conditions, [MacAyeal, 1996b] computes the rescaling parameters as also shown in the table.

Symbol	Meaning	Value
a	Mean annual accumulation	$.3\ m\ a^{-1}$
q_0	Flux at domain boundary	$4 \times 10^5\ m^2\ a^{-1}$
H_0	Thickness at domain boundary	$10^3\ m$
A	Glen's flow law parameter	$3.155818 \times 10^{-25}\ Pa^{-3}\ s^{-1}$
Z	Thickness rescaling parameter	$205.7426\ m$
U	Velocity rescaling parameter	$400\ m\ a^{-1}$
L	Distance rescaling parameter	$274.32\ km$

Table 5.3 EISMINT-1 constants and rescaling parameters for ice tongue experiment

Again approximating a flowline experiment by enforcing periodic boundary conditions, we performed a diagnostic run on the steady-state geometry (note that, because

both the grounding line flux and the annual accumulation are not zero, the velocity for the steady state is not necessarily zero). Integrating the analytical solution as above reveals a mean percent error of 2.56% between the computed solution and the analytical solution. However, the majority of this error is not due to the way the model handles the ice shelf front. Rather, it arises from the fact that the initial gradient at the grounding line is quite steep and difficult to resolve. This is supported by the fact that, if we assume that the grounding line is moved one node forward and take the computed velocity there as the Dirichlet condition, the mean error is reduced to just .58%. It is additionally supported by the fact that the 2.56% error responds favorably to grid refinement. The vertically averaged velocities resulting from both the model, the correct analytical solution, and the analytical solution adjusted to ignore the initially high thickness gradient are summarized in Figure 5.4

5.3 Ross ice shelf experiment

5.3.1 Description of Experiment

The EISMINT-Ross ice shelf experiment is an intercomparison experiment for shallow shelf models and other models capable of modeling ice shelf physics. The experiment is based on velocity observations from the Ross Ice Shelf Geophysical and Glaciological Survey (RIGGS), whose results were first published by Thomas, R.H., D.R. MacAyeal, D.H. Eilers and Gaylord [1984]. Velocity data from the RIGGS observation stations has been interpolated and aggregated with other observations of the Ross ice shelf by MacAyeal et al. [1996].

Because it is meant as an intercomparison solely of ice shelf models, EISMINT-Ross prescribes as the computation domain only those locations that are fully floating and located on the Ross ice shelf. Ice flux into the domain from ice streams are prescribed

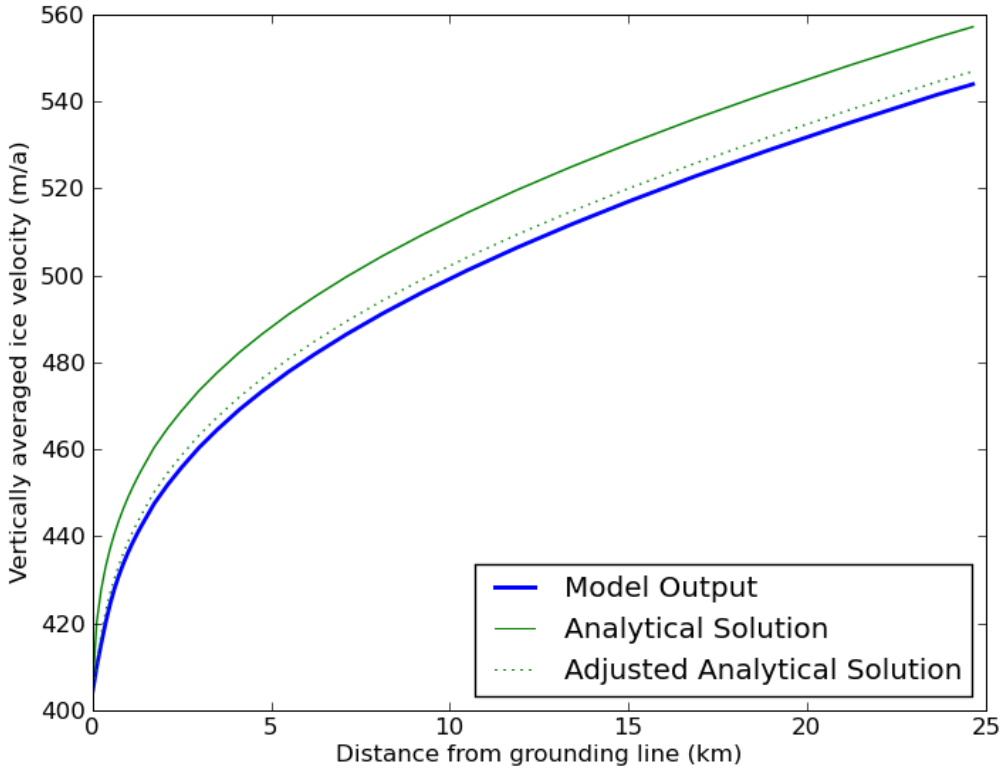

Figure 5.4 The model output given Van der Veen's steady-state ice tongue profile is compared to both the analytical solution and an analytical solution that is adjusted to account for the difficulty of resolving the high thickness gradient just after the grounding line.

as Dirichlet boundary conditions applied to the velocity computation. Other areas of the domain not on the Ross ice shelf (such as Roosevelt Island) are excluded from the domain by prescribing a zero-value Dirichlet boundary (thereby holding those velocities at zero during the computation). Although these simplifications are not needed for a higher-order model that can treat ice shelves holistically with ice streams and grounded regions within the shelf, they were maintained for the purpose of intercomparison. As an additional simplification, although ice temperatures have been determined through observation, the EISMINT-Ross intercomparison is prescribed for an isothermal ice shelf; this is because models using polythermal ice have so far been unsuccessful in capturing the correct behavior [MacAyeal et al., 1996].

5.3.2 Results

Evaluating the EISMINT-Ross experimental results involves two layers of intercomparison. First, our experimental results are compared to the observations from the RIGGS stations. The Chi-squared error is computed. Figure 5.5 shows the map of our computed velocities; Figure 5.6 shows how our model compares to the observations. The second layer of intercomparison is a comparison maximum velocity observed in our model and the chi-squared error to other results published by MacAyeal et al. [1996]. Table 5.4 shows that we compare favorably to other models attempting the same problem, both in terms of the observed chi-squared error and the maximum observed velocity. Only one published result, Bremerhaven1, outperforms our model. This does not mean that we are performing correctly! Rather, it means that any mistakes that we made with respect to our ice shelf physics are systemic and are present to some degree in other participating models. [MacAyeal et al., 1996] believes that this is due to a combination of using isothermal rather than polythermal ice and specifying an incorrect basal boundary condition. Modeling flow over Roosevelt Island

rather than specifying zero flux at the grounding line may also produce more realistic results; this was not done originally as it was impossible to handle this case in SSA models.

Model	X^2	Max vel. (m/a)
Bremerhaven1	3605	1379
Bremerhaven2	12518	1663
Chicago1	5114	1497
Chicago2	5125	1497
Grenoble	5237	1508
Missoula	**4962**	**1495**

Table 5.4 Error from my model's Ross diagnostic run compared to those of other models submitted to the intercomparison project [MacAyeal et al., 1996]

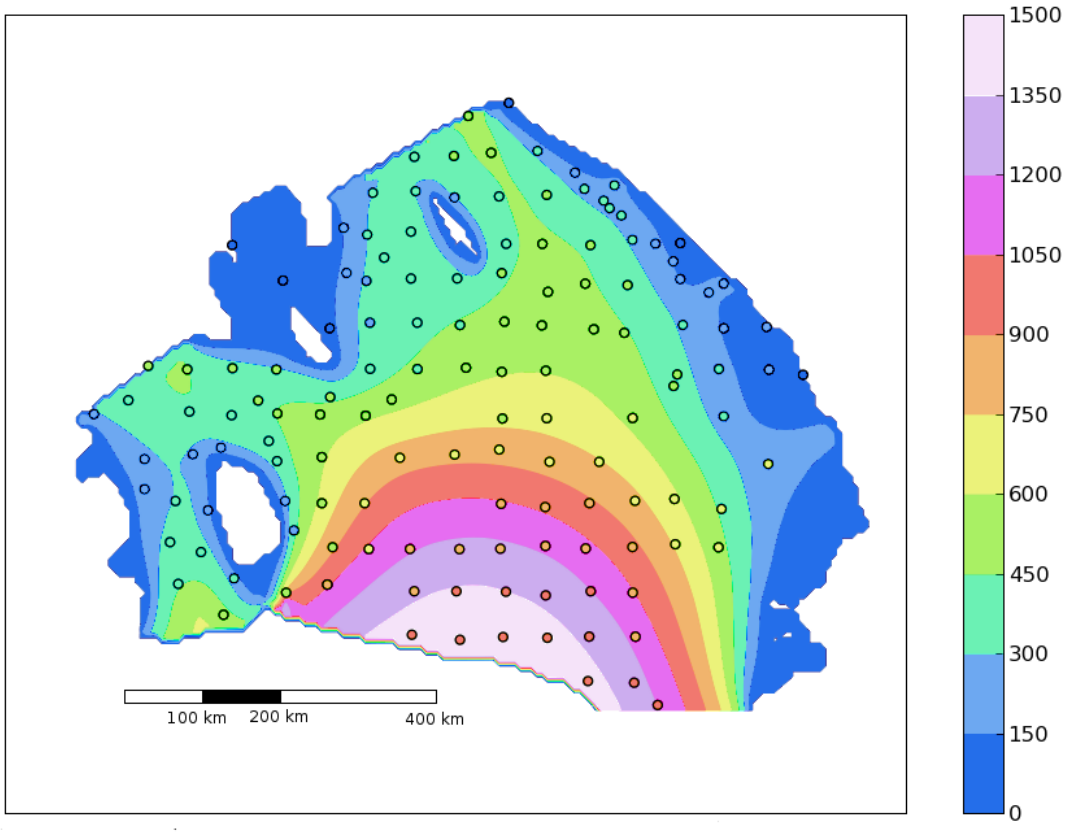

Figure 5.5 Map of velocities computed for EISMINT-Ross. Discrete data points represent observed velocities at RIGGS stations.

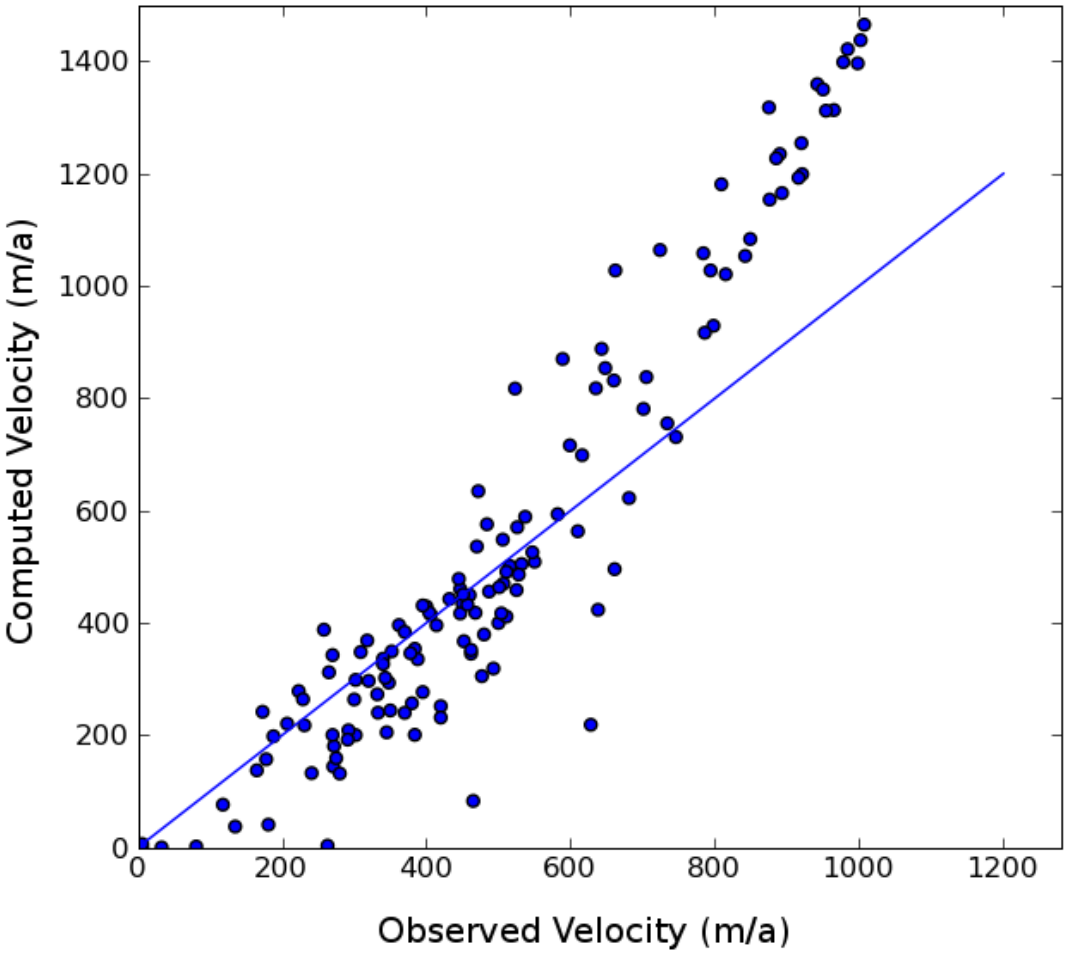

Figure 5.6 Computed velocity versus velocity observed at RIGGS station at that location for EISMINT-Ross.

CHAPTER 6 CONCLUSION

Although this thesis represents over a year of software development effort, this project was by no means isolated. As such, I would like to conclude by looking not only behind at the work that I did and what we can take from it, but also ahead at how this past year's effort impacts the Community Ice Sheet Model and where the project might go from here.

6.1 Lessons Learned

The integration of higher-order physics into CISM provided numerous challenges and opportunities for learning. There are several lessons we can take from the software engineering effort. First, two "best practices" that were particularly reaffirmed are the importance of frequent regression testing and the related importance of version management. Because Glimmer is such a highly coupled model, the code base was quite fragile and prone to breaking when changing seemingly unrelated parts of the code. Testing using both EISMINT and ISMIP-HOM proved essential in catching these errors, and the presence of a version control system (namely Subversion) made it possible to determine quickly the change that lead to the regression, especially when tests were not run as often as would be ideal and a binary search through the revision archive was needed to find the responsible revision! The quality of the code benefited greatly from this methodology.

The rapid prototyping approach used during the development of the model had both benefits and drawbacks. A fast integration allowed us to test often against intercomparison experiments and to find problems with the integration early. Debugging any code is a lengthy, arduous process, and the nature of scientific code means that this was doubly true in this instance. Early feedback was therefore essential. However, in some cases the rapid prototype itself created bugs. In particular, a plethora of errors arose from the coordinate transposition of Pattyn's model with respect to CISM's system, and many of these errors were a result of the ad-hoc way in which the transposition was initially handled. It is my opinion that re-engineering Pattyn's code to natively use Glimmer's coordinate system would have been a large up-front investment but would have saved time overall. Additionally, the danger with any prototype code is that the prototype gains a sort of inertia, and refactoring must be done consciously and with an understanding that it is worth the effort in the long run. This rings just as true in the case of CISM as in any other development effort.

I had mentioned that debugging scientific code provides a particular set of challenges compared to debugging other types of programs; this arises from the fact that this kind of scientific program in particular involves the processing of large fields rather than scalar quantities. A visual debugging process is therefore key – writing snapshots of the fields involved to text or NetCDF files and using an appropriate visualization tool (Matlab or ncview) revealed patterns that numerical inspection of the data could not. Choosing an appropriate level of granularity was also key; though insights sometimes arose from directly inspecting the sparse matrix that was assembled, more often than not the exercise was a waste of time.

In addition, the fact that a sparse matrix solve is involved means that the errors are particularly hard to diagnose. The solve is a large "black box" at the very core of the higher-order scheme that transforms inputs to outputs in such a way that it is

nearly impossible to backtrack from a bad output and determine what specific input error caused the error on the output side. In many cases, I erroneously responded to this by "randomly thrashing" - I would change signs or constants in the code to see how the model responded. This was rarely helpful.

A more useful exercise was to *simplify* the model rather than *change* it randomly. In one case where ice shelf physics became unstable when the shelf front was at a 45° angle with respect to the numerical grid, a month-long debugging effort was ended by changing the model such that viscosity was constant. Combined with the insight of which terms in the model disappeared when viscosity derivatives were zero, I was able to find that the numerical discretization of $\frac{\partial^2 \mu}{\partial xy}$ was reaching into areas with no ice and that a grid cell needed to be a shelf front if it was diagonally adjacent to an ocean cell! A few days of more directed searching therefore found an error that a month of less directed probing did not.

6.2 Towards a True Community Model

Understanding the response of ice sheets to climate forcing is an essential part of understanding the impact climate change will have in the 21st century. We have seen that the previous generation of ice sheet models, of which Glimmer was a member, is unable to properly capture the behavior of some of the more important glaciological systems. The work in this thesis represents an important first step towards readying the Community Ice Sheet Model to tackle the problems currently facing the ice sheet modeling community. To our knowledge, no other program combines higher-order physics with the infrastructure available in CISM, putting it on the leading edge of ice models.

As I have discussed before, though, the work in this thesis is but one arm of a major

effort to prepare CISM to meet these challenge. Work at Los Alamos to integrate similar physics has also progressed over the same time period. The fact that two models were integrated forced additional design work to make sure that the models could co-exist. But with a framework already in development for multiple models to exist in the system, with a little extra attention towards good object-oriented practices we made sure that the framework can scale to allow much more than two methods of computation to exist in CISM. This emphasizes a single fact: that CISM is a community effort. It now exists as not only a usable next-generation ice sheet model, but also as a better platform for experimentation. Trying different methods of solving the ice equations within this framework should be encouraged.

There is more work that needs to be done, however, before CISM is able to answer the scientific questions that we have set out to answer. One issue that needs to be tackled is the model's ability to integrate with prognostic solvers, evolving both the geometry and the temperature distribution of the ice sheet. Work has been done on this as part of this thesis, but additional development and testing is needed.

A larger issue is that of scalability. The problem size that Pattyn's model is able to solve is limited by the capabilities of a single processor. Continent-scale diagnostic solves are just at the limit of what can be accomplished, and prognostic solves using higher-order physics are slow enough that they are not generally feasible. The situation will not improve with more processing power, as newer microchip technology is used to fabricate chips with more processors, not single processors with more power. Therefore, parallelism is essential to the scalability of this model.

This scalability can be partially accomplished by parallelizing the sparse matrix solve, as that is the main bottleneck in the higher-order solve. With the addition of the sparse matrix subsystem as part of this thesis, it is much easier to integrate a different sparse matrix solving library that is set up for parallel computation. At another

university, an effort is underway to integrate Pardiso ("Parallel Direct Solver") in order to ameliorate this bottleneck.

However, this parallelism does not use the Message Passing Interface (MPI) and is therefore limited to multiple cores on a single processor. It will also reach diminishing returns as the sparse solve is parallelized to the point that it is no longer the main bottleneck. In order for a truly scalable solver to be developed, techniques such as MPI and domain decomposition need to be applied at a higher level in the model. Work is underway at LANL to parallelize the entire dynamic core, not just the higher-order core, in just this way. This has the additional advantage that parallel processing is available for almost no additional effort to model developers, again contributing to the community aspect of the model. It will, however, take a significant development effort until this is ready.

Finally, the fastest and most complete model in the world is useless if it does not support a scientific effort. Datasets are being developed in parallel to the model development. Inverse modeling techniques need to be developed to determine the initial temperature conditions of the ice sheet, as well as to specify the basal boundary condition. Finally, we return to the original purpose of building these models: to determine the impact of climate change on the Earth's glaciers, and to determine the impact of changes to those glaciers on the Earth's sea level. The specification and execution of these experiments is the purpose of the newly created SeaRISE and Ice2Sea efforts. The presence of a community ice sheet model as a foundation that these efforts can build on for experiment development and intercomparison is an important step forward in answering these questions.

APPENDIX A GLIDE CONFIGURATION WITH HIGHER-ORDER OPTIONS

A.1 Introduction

In this appendix, I present the format of the Glide configuration file. The majority of this appendix originally appeared in the Glimmer documentation [Hagdorn et al., 2006], and has been expanded with the higher-order options. New sections, options, and settings have been marked with an asterisk. The major changes are the addition of a higher-order options section, the addition of a new way of specifying one of several built-in vertical layers, and the ability to choose incremental remapping as an evolution scheme.

A.2 Documentation

The format of the configuration files is similar to Windows `.ini` files and contains sections. Each section contains key, values pairs.

- Empty lines, or lines starting with a `#`, `;` or `!` are ignored.
- A new section starts with the the section name enclose with square brackets, e.g. `[grid]`.
- Keys are separated from their associated values by a `=` or `:`.

Sections and keys are case sensitive and may contain white space. However, the configuration parser is very simple and thus the number of spaces within a key or section name also matters. Sensible defaults are used when a specific key is not found.

`[grid]`	
Define model grid. Maybe we should make this optional and read grid specifications from input netCDF file (if present). Certainly, the input netCDF files should be checked (but presently are not) if grid specifications are compatible.	
`ewn`	(integer) number of nodes in x–direction
`nsn`	(integer) number of nodes in y–direction
`upn`	(integer) number of nodes in z–direction
`dew`	(real) node spacing in x–direction (m)
`dns`	(real) node spacing in y–direction (m)
`sigma_file`	(string) Name of file containing σ coordinates. Alternatively, the sigma levels may be specified using the `[sigma]` section decribed below. If no sigma coordinates are specified explicitly, they are calculated based on the value of `sigma_builtin`
`sigma_builtin` *	If sigma coordinates are not specified in this configuration file or usin `sigma_file` option, this specifies how to compute the sigma coordinates. **0** Use Glimmer's default spacing $\sigma_i = \frac{1-(x_i+1)^{-n}}{1-2^{-n}}$ with $x_i = \frac{\sigma_i - 1}{\sigma_n - 1}, n = 2.$ 1 Use evenly spaced layers 2 Use the spacing defined for Pattyn's model

continued on next page

continued from previous page

`[sigma]`	
Define the sigma levels used in the vertical discretization. This is an alternative to using a separate file (specified in section `[grid]` above). If neither is used, the levels are calculated as described above. **This does not work in version 1.0.0 — a bugfix will be incorporated into v.1.0.2.**	
`sigma_levels`	(real) list of sigma levels, in ascending order, separated by spaces. These run between 0.0 and 1.0

`[time]`	
Configure time steps, etc. Update intervals should probably become absolute values rather than related to the main time step when we introduce variable time steps.	
`tstart`	(real) Start time of the model in years
`tend`	(real) End time of the model in years
`dt`	(real) size of time step in years
`ntem`	(real) time step multiplier setting the ice temperature update interval
`nvel`	(real) time step multiplier setting the velocity update interval

`[options]`	
Parameters set in this section determine how various components of the ice sheet model are treated. Defaults are indicated in bold.	
`ioparams`	(string) name of file containing netCDF I/O configuration. The main configuration file is searched for I/O related sections if no file name is given (default).
`temperature`	0 isothermal **1** full
`flow_law`	**0** Patterson and Budd 1 Patterson and Budd (temp=-10degC) 2 constant value, taken from default_flwa *
`basal_water`	0 local water balance 1 local water balance + const flux **2** none
`marine_margin`	0 ignore marine margin **1** Set thickness to zero if floating 2 Set thickness to zero if relaxed bedrock is below a given depth 3 Lose fraction of ice when edge cell 4 Set thickness to zero if present-day bedrock is below a given depth

continued on next page

continued from previous page

`slip_coeff`	**0** zero
	1 set to a non–zero constant everywhere
	2 set constant where the ice base is melting
	0 $\propto$ basal water
`evolution`	**0** pseudo-diffusion
	1 ADI scheme
	2 diffusion
	3 * Higher-order incremental remapping
`vertical_integration`	**0** standard
	1 obey upper BC
`topo_is_relaxed`	**0** relaxed topography is read from a separate variable
	1 first time slice of input topography is assumed to be relaxed
	2 first time slice of input topography is assumed to be in isostatic equilibrium with ice thickness.
`periodic_ew`	**0** switched off
	1 periodic lateral EW boundary conditions (i.e. run model on torus)
`periodic_ns`	**0** switched off
	1 periodic lateral NS boundary conditions (i.e. run model on torus)
`hotstart`	Hotstart the model if set to 1. This option only affects the way the initial temperature and flow factor distribution is calculated.

`[ho_options] *`

Parameters set in this section determine how various components of the higher-order extensions to the ice sheet model are treated. Defaults are indicated in bold. To enable higher-order computation, set diagnostic_scheme to something other than 0. The computed velocities will only be used prognistically, however, if an evolution scheme that can make use of them has been enabled (currently only incremental remapping)

`diagnostic_scheme`	**0** No higher-order diagnostics
	1 Pattyn/Bocek diagnostic, computed on the ice grid
	2 Pattyn/Bocek diagnostic, computed on the velocity grid
	3 Payne/Price diagnostic

continued on next page

continued from previous page

`basal_stress_input`	**0**	Ice glued to the bed (beta field is all NaN)
	1	Beta field is 1/soft
	2	Beta field is 1/btrc
	3	Beta field is read from input NetCDF independent of shallow-ice sliding law
	4	Use slip ratio, described in ISMIP-HOM F [Pattyn and Payne, 2006]
`basal_stress_type`	**0**	Linear bed
	1	Plastic bed
`which_ho_efvs`	**0**	Use full nonlinear viscosity
	1	Apply a linear viscosity
`which_ho_source`	**0**	Use vertically averaged formulation of the shelf front source term
	1	Use a vertically explicit formulation of the shelf front source term (currently not working)
	2	Turn off the ice shelf front and treat those locations as a land margin instead
`guess_specified`	**0**	Use a model-defined initial guess (SIA for Pattyn/Bocek, zero for Payne/Price)
	1	Read the initial velocity guess from uvelhom and vvelhom
`include_thin_ice`	0	Do not include ice below the ice dynamics limit in the higher-order diagnostic
	1	Compute higher-order diagnostic for all ice, even ice below the ice dynamics limit
`which_ho_sparse`	**0**	Solve sparse linear system with LU-preconditioned biconjugate gradient method
	1	Solve sparse linear system with LU-preconditioned GMRES method
	2	Solve sparse linear system with UMFPACK (not always available)

continued on next page

continued from previous page	
`which_ho_sparse_fallback`	Specifies a sparse solver package to use if the package specified in `which_ho_sparse` fails. The options are the same, though setting to -1 disables the fallback (this is the default).

`[parameters]`	
Set various parameters.	
`log_level`	(integer) set to a value between 0, no messages, and 6, all messages are displayed to stdout. By default messages are only logged to file.
`ice_limit`	(real) below this limit ice is only accumulated; ice dynamics are switched on once the ice thickness is above this value.
`marine_limit`	(real) all ice is assumed lost once water depths reach this value (for `marine_margin`=2 or 4 in `[options]` above). Note, water depth is negative.
`calving_fraction`	(real) fraction of ice lost due to calving.
`geothermal`	(real) constant geothermal heat flux.
`flow_factor`	(real) the flow law is enhanced with this factor
`hydro_time`	(real) basal hydrology time constant
`isos_time`	(real) isostasy time constant
`basal_tract_const`	constant basal traction parameter. You can load a nc file with a variable called `soft` if you want a specially variying bed softness parameter.
`basal_tract`	(real(5)) basal traction factors. Basal traction is set to $B = \tanh(W)$ where the parameters (1) width of the tanh curve (2) W at midpoint of tanh curve [m] (3) B minimum [$\mathrm{ma}^{-1}\mathrm{Pa}^{-1}$] (4) B maximum [$\mathrm{ma}^{-1}\mathrm{Pa}^{-1}$] (5) multiplier for marine sediments
`default_flwa` *	Flow law parameter A to use in isothermal experiments (flow_law set to 2). Default value is 10^{-16}.

`[isostasy]`	
Isostatic adjustment is only enabled if this section is present in the configuration file. The options described control isostasy model.	
`lithosphere`	**0** local lithosphere, equilibrium bedrock depression is found using Archimedes' principle 1 elastic lithosphere, flexural rigidity is taken into account

continued on next page

continued from previous page

`asthenosphere`	**0** fluid mantle, isostatic adjustment happens instantaneously 1 relaxing mantle, mantle is approximated by a half-space
`relaxed_tau`	characteristic time constant of relaxing mantle (default: 4000.a)
`update`	lithosphere update period (default: 500.a)

`[projection]`

Specify map projection. The reader is referred to Snyder J.P. (1987) *Map Projections - a working manual.* USGS Professional Paper 1395.

`type`	This is a string that specifies the projection type (`LAEA`, `AEA`, `LCC` or `STERE`).
`centre_longitude`	Central longitude in degrees east
`centre_latitude`	Central latitude in degrees north
`false_easting`	False easting in meters
`false_northing`	False northing in meters
`standard_parallel`	Location of standard parallel(s) in degrees north. Up to two standard parallels may be specified (depending on the projection).
`scale_factor`	non-dimensional. Only relevant for the Stereographic projection.

`[elastic lithosphere]`

Set up parameters of the elastic lithosphere.

`flexural_rigidity`	flexural rigidity of the lithosphere (default: 0.24e25)

`[GTHF]`

Switch on lithospheric temperature and geothermal heat calculation.

`num_dim`	can be either `1` for 1D calculations or 3 for 3D calculations.
`nlayer`	number of vertical layers (default: 20).
`surft`	initial surface temperature (default 2°C).
`rock_base`	depth below sea-level at which geothermal heat gradient is applied (default: -5000m).
`numt`	number time steps for spinning up GTHF calculations (default: 0).
`rho`	The density of lithosphere (default: 3300kg m^{-3}).
`shc`	specific heat capcity of lithosphere (default: 1000J kg^{-1} K^{-1}).

continued on next page

continued from previous page	
`con`	thermal conductivity of lithosphere (3.3 W m^{-1} K^{-1}).

NetCDF I/O can be configured in the main configuration file or in a separate file (see `ioparams` in the `[options]` section). Any number of input and output files can be specified. Input files are processed in the same order they occur in the configuration file, thus potentially overwriting priviously loaded fields.

`[CF default]`	
This section contains metadata describing the experiment. Any of these parameters can be modified in the `[output]` section. The model automatically attaches a time stamp and the model version to the netCDF output file.	
`title`	Title of the experiment
`institution`	Institution at which the experiment was run
`references`	References that might be useful
`comment`	A comment, further describing the experiment

`[CF input]`	
Any number of input files can be specified. They are processed in the order they occur in the configuration file, potentially overriding previously loaded variables.	
`name`	The name of the netCDF file to be read. Typically netCDF files end with `.nc`.
`time`	The time slice to be read from the netCDF file. The first time slice is read by default.

`[CF output]`	
This section of the netCDF parameter file controls how often selected variables are written to file.	
`name`	The name of the output netCDF file. Typically netCDF files end with `.nc`.
`start`	Start writing to file when this time is reached (default: first time slice).
`stop`	Stop writin to file when this time is reached (default: last time slice).
`frequency`	The time interval in years, determining how often selected variables are written to file.
`variables`	List of variables to be written to file. See Appendix B for a list of known variables. Names should be separated by at least one space. The variable names are case sensitive. Variable `hot` selects all variables necessary for a hotstart.

continued on next page

continued from previous page	

APPENDIX B GLIDE NETCDF VARIABLES

B.1 Introduction

In this appendix, I present the NetCDF variables that Glide reads and writes. This originally appeared in the Glimmer documentation [Hagdorn et al., 2006], and has been expanded with the higher-order options. A $*$ denotes variables that are used as input, all others are output only. A †denotes variables that have been added for this thesis.

B.2 Documentation

Name	Description	Units
`level`	sigma layers CF name: `land_ice_sigma_coordinate`	1
`lithoz`	vertical coordinate of lithosphere layer	meter
`x0`	Cartisian x-coordinate, velocity grid	meter
`x1`*	Cartisian x-coordinate	meter
`y0`	Cartisian y-coordinate, velocity grid	meter
`y1`*	Cartisian y-coordinate	meter

continued on next page

continued from previous page

Name	Description	Units
`acab`	accumulation, ablation rate CF name: `land_ice_surface_specific_mass_balance`	meter/year
`acab_tavg`	accumulation, ablation rate (time average) CF name: `land_ice_surface_specific_mass_balance`	meter/year
`age`*	ice age CF name: `land_ice_age`	year
`artm`	annual mean air temperature CF name: `surface_temperature`	degree_Celsius
`beta`*†	higher-order bed stress coefficient	unknown
`bheatflx`*	basal heat flux	watt/meter2
`bmlt`*	basal melt rate CF name: `land_ice_basal_melt_rate`	meter/year
`bmlt_tavg`	basal melt rate (time average) CF name: `land_ice_basal_melt_rate`	meter/year
`btemp`	basal ice temperature CF name: `land_ice_temperature`	degree_Celsius
`btrc`	basal slip coefficient	meter/pascal/year
`bwat`*	basal water depth	meter
`calving`	ice margin calving	meter
`diffu`	apparent diffusivity	meter2/year
`dusrfdtm`	rate of upper ice surface elevation change	meter/year
`eus`	global average sea level	meter

continued on next page

continued from previous page		
Name	Description	Units
	CF name: `global_average_sea_level_change`	
`flwa`*	Pre-exponential flow law parameter	pascal/year
`iarea`	area covered by ice	km2
`ivol`	ice volume	km3
`kinbcmask`*†	Mask of locations where uvelhom, vvelhom value should be held as Dirichlet boundaries	1
`lat`*	latitude CF name: `latitude`	degreeN
`litho_temp`*	lithosphere temperature	degree_Celsius
`lon`*	longitude CF name: `longitude`	degreeE
`lsurf`	ice lower surface elevation	meter
`relx`*	relaxed bedrock topography	meter
`slc`	isostatic adjustment CF name: `bedrock_altitude_change_due_to_isostatic_adjustment`	meter
`soft`*	bed softness parameter	meter/pascal/year
`tau_xz`†	X component vertical shear stress	kPa
`tau_yz` †	Y component vertical shear stress	kPa
`taux`	basal shear stress in x direction	kilopascal
`tauy`	basal shear stress in y direction	kilopascal
`temp`*	ice temperature CF name: `land_ice_temperature`	degree_Celsius
		continued on next page

continued from previous page

Name	Description	Units
`thk`*	ice thickness CF name: `land_ice_thickness`	meter
`thkmask`	mask	1
`topg`*	bedrock topography CF name: `bedrock_altitude`	meter
`ubas`*	basal slip velocity in x direction CF name: `land_ice_basal_x_velocity`	meter/year
`ubas_tavg`	basal slip velocity in x direction (time average) CF name: `land_ice_basal_x_velocity`	meter/year
`uflx`	flux in x direction	meter2/year
`usurf`*	ice upper surface elevation CF name: `surface_altitude`	meter
`uvel`	ice velocity in x direction CF name: `land_ice_x_velocity`	meter/year
`uvelhom`*†	ice velocity in x direction according to higher order model CF name: `land_ice__x_velocity`	meter/year
`vbas`*	basal slip velocity in y direction CF name: `land_ice_basal_y_velocity`	meter/year
`vbas_tavg`	basal slip velocity in y direction (time average)	meter/year

continued on next page

continued from previous page		
Name	Description	Units
	CF name: `land_ice_basal_y_velocity`	
`velnormhom` †	Ice velocity magnitude according to higher-order model	meter/year
`vflx`	flux in x direction	meter2/year
`vvel`	ice velocity in y direction CF name: `land_ice_y_velocity`	meter/year
`vvelhom`*†	ice velocity in y direction according to higher order model CF name: `land_ice_y_velocity`	meter/year
`wgrd`	Vertical grid velocity	meter/year
`wvel`	vertical ice velocity	meter/year

APPENDIX C SPARSE MATRIX DOCUMENTATION

C.1 User documentation

This section documents the usage of the sparse solver subsystem to solve a sparse linear system.

C.1.1 Modules

In order to make use of the sparse solver subsystem, first import the following modules:

1. `glimmer_sparse_type` – This module contains structure and function definitions for a sparse matrix type. This module concerns creating a sparse matrix in triad format, and is not dependent on the solver used

2. `glimmer_sparse` – This module provides the linear system solver framework that makes use of the sparse matrix type.

C.1.2 Setting up a linear system

The derived type `sparse_matrix_type` provides a high-level way to allocate a sparse matrix. If you have declared

```
type(sparse_matrix_type) :: matrix
```

call

```
new_sparse_type(n, m, matrix)
```

to initialize the data structure and allocate memory, where n is the order of the matrix and m is the expected number of nonzero entries. m need only be an initial guess; the data structure will grow dynamically if it is exceeded, and will handle under-use of the allocated memory properly.

To add a nonzero entry to the sparse matrix:

```
sparse_insert_val(matrix, i, j, a)
```

where i and j are the one-based row and column in the matrix in which to insert the value, and a is the value to insert. This will perform basic sanity checks: $0 < i \leq n$, $0 < j \leq n$, and $a \neq 0$. It will *not* check for double insertions, as this would incur a large inefficiency. Also, bear in mind that there is no check outside of the sparse matrix solver that you have placed at least one nonzero entry in every row and column - it is up to you to make sure that matrix is nonsingular.

A sparse matrix can be cleared without deallocating memory using

```
sparse_clear(matrix)
```

This is useful when creating a new matrix for the same kind of operation, such as solving the same set of equations for a different viscosity field. Once completely done with the memory it can be deallocated with

```
del_sparse_matrix(matrix)
```

As an example, the following code creates a 10x10 tridiagonal matrix suitable for solving a 1-D diffusion problem implicitly:

```
type{sparse_matrix_type} :: matrix
integer :: i
```

```
!Tridiagonal matrix will have 3 entries for every row.
call new sparse_type(10, 30, matrix)

!Impose Dirichlet boundaries
call sparse_insert_val(matrix, 1, 1, 1)
call sparse_insert_val(matrix, 10, 10, 1)

!Fill in the matrix for the interior
do i = 2, 9
    call sparse_insert_val(matrix, i, i-1, -1)
    call sparse_insert_val(matrix, i, i+1, -1)
    call sparse_insert_val(matrix, i, i, 2)
end do
```

C.1.3 Solving a linear system

A linear system solve is conceptually broken into six phases. Note that you will need to call all five, even if some solver packages may not need them. Although somewhat complicated, this was done so that operations that need not be done for every solve can be performed once and reused.

1. Solver initialization: Allocates memory used by the sparse solver that is independent of the matrix used.

2. Workspace allocation: This consists of allocating memory that can be done as soon as the size of the linear system is known. Once the workspace is allocated, it can remain allocated until the size of the linear system changes.

3. Matrix preprocessing: This consists of steps that need to be done for each matrix, but not necessarily for each solve, such as LU factorization.

4. Matrix solve

5. Matrix postprocessing: Deallocates all memory from the preprocessing step. Must be called before preprocessing another matrix.

6. Workspace deallocation: Deallocates all memory in the workspace allocation step.

There are also two sparse matrix data structures that need to be instantiated. These roughly correspond to the public and private interfaces of the sparse solver system. The public type is `sparse_solver_options`, and holds user-configurable options such as the error tolerance of an iterative method. The private type is `sparse_solver_workspace`, and holds temporary memory required by the solver. `Sparse_solver_workspace` should *always* be treated as an opaque type, and should never be accessed by client code.

I will present a simple example of setting up, running, and tearing down the solver. This example will not present any more complicated use cases of using the same allocated workspace across several solves, or of solving the same matrix with different right-hand side vectors. The module `ice3d_lib` is a good example of the former, though there are no examples of the latter in Glimmer/CISM currently.

If these two types are declared as

```
type(sparse_solver_options) :: opt
type(sparse_solver_workspace) :: wk
```

first step is initialize the sparse solver, including setting any default options, by calling

```
sparse_solver_default_options(method, opt)
```

where method is an integer indicating the sparse solver method. The possible values are listed at the beginning of `glimmer_sparse.F90`, and are currently

Option	Solver
0	Biconjugate Gradient Method (BiCG) with incomplete LU preconditioner from SLAP S
1	Method of Generalized Minimum Residuals (GMRES) with incomplete LU precondition
2	Unsymmetric Multifrontal Method direct solver from UMFPACK Davis [2004]

Once done, initialize the workspace and preprocess the matrix:

```
call sparse_allocate_workspace(matrix, opt, wk)
call sparse_solver_preprocess(matrix, opt, wk)
```

`Sparse_allocate_workspace` takes an optional fourth argument of the maximum number of nonzeros that the sparse matrix can have. This is to allow the workspace to be allocated *before* the matrix is set up, in case the same workspace is used to solve several systems. If left out, the workspace is allocated to support the current number of nonzeros in the provided sparse matrix structure.

Next, solve the sparse linear system

```
ierr = sparse_solve(matrix, rhs, answer, opt, wk, err, iter, verbose)
```

where:

- rhs is an array of length n that contains the right-hand side of the equation $Ax = b$
- answer is an array of length n. It should contain an initial guess of the solution before the call. After the call, it contains the solution.

- err is a an output variable that contains the error tolerance that the solver converged to (0 if a direct solver was used)
- iter is an output variable that contains the number of iterations that the solve took (1 if a direct solver was used)
- verbose is a boolean that specifies whether the solver should, if available, provide verbose output

This call returns an error flag that is zero if solve completed successfully, and nonzero if it did not. See section C.1.4 for more information.

Finally, clean up the memory used by the sparse matrix solver:

```
call sparse_solver_postprocess(matrix, opt, wk)
call sparse_destroy_workspace(matrix, opt, wk)
```

C.1.4 Handling errors

There are several ways to handle a sparse matrix error, indicated by a nonzero return value from `sparse_solve`. To print the error to stdout:

```
sparse_interpret_error(options, error_code)
```

More generally, the error code can be converted to a human-readable string:

```
sparse_interpret_error(options, error_code, error_string)
```

Finally, a subroutine is included for convenience to automatically log the error and stop the ice model:

```
subroutine handle_sparse_error(matrix, solver_options, error_code, __FIL
```

__FILE__ and __LINE__ need not be included, but if so the log file can point to the place in the code where the sparse solve failed, rather than to the handle_sparse_error routine its self.

BIBLIOGRAPHY

J. L. Bamber, R. L. Layberry, and S. P. Gogenini. A new ice thickness and bed data set for the Greenland ice sheet 1: Measurement, data reduction, and errors. *Journal of Geophysical Research*, 106(D24):33773–33780, 2001.

E. Bueler and J. Brown. The shallow shelf approximation as a "sliding law" in a thermomechanically coupled ice sheet model. *Journal of Geophysical Research*, 2009. In Press.

E. Bueler, J. Brown, and C. Lingle. PISM: a parallel ice sheet model. Available at http://www.pism-docs.org, 2008.

T.A. Davis. Algorithm 832: Umfpack, an unsymmetric-pattern multifrontal method. *ACM Transaction on Mathematical Software*, 30(2):196–199, June 2004.

J.B. Drake, P.W. Jones, and GR Carr. Overview of the Software Design of the CCSM. *INTERNATIONAL JOURNAL OF HIGH PERFORMANCE COMPUTING APPLICATIONS*, 19(3):177, 2005.

J.K. Dukowicz and J. R. Baumgardner. Incremental remapping as a transport/advection algorithm. *J. Comput. Phys.*, 160:318–335, 2000.

S. Fuyuki, A. Abe-Ouchi, and H. Blatter. An improved numerical scheme to compute horizontal gradients at the ice-sheet margin: its effect on the simulated ice thickness and temperature. *Annals of Glaciology*, 46:87, 2007.

E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Reusable Elements of Object-Oriented Programming*. Addison-Wesley, Boston, MA, 1995.

R. Greve. Application of a polythermal three-dimensional ice sheet model to the Greenland ice sheet: Response to steady-state and transient climate scenarios. *J. Climate*, 10(5):901–918, 1997.

M. Hagdorn, I. Rutt, T. Payne, and F. Hebeler. GLIMMER-The GENIE Land Ice Model with Multiply Enabled Regions-Documentation, 2006.

R. C. A. Hindmarsh. A numerical comparison of approximations to the Stokes equations used in ice sheet and glacier modeling. *J. Geophys. Res*, 109(F01012): doi:10.1029/2003JF000065, 2004.

R. C. A. Hindmarsh and A. J. Payne. Time-step limits for stable solutions of the ice-sheet equation. *Ann. Glaciol.*, 23:74–85, 1996.

R.C. Hindmarsh. The role of membrane-like stresses in determining the stability and sensitivity of the Antarctic ice sheets: back pressure and grounding line motion. *Philosophical Transactions A*, 364(1844):1733, 2006.

R. LeB. Hooke. *Principles of Glacier Mechanics*. Prentice-Hall, London, 1998.

K. Hutter. Theoretical glaciology. In *Material science of ice and the mechanics of glaciers and ice sheets*. D. Reidel Publishing Company/Tokyo, Dordrecht etc, 1983.

P. Huybrechts. J. deWolde, 1999: The dynamic response of the Greenland and Antarctic ice sheets to multiple-century climatic warming. *Journal of Climate*, 12(8): 2169–2188, 1999.

P. Huybrechts, T. Payne, and The EISMINT Intercomparison Group. The EISMINT benchmarks for testing ice–sheet models. *Ann. Glaciol.*, 23:1–12, 1996.

W. Lipscomb and E.C. Hunke. Modeling sea ice transport using incremental remapping. *Monthly Weather Review*, 132:1341–1354, 2004.

M. B. Lythe and D. G. Vaughan. BEDMAP: A new ice thickness and subglacial topographic model of Antarctica. *J. Geophys. Res.*, 106(B6):11335–11351, 2001.

D. R. MacAyeal. Large-scale ice flow over a viscous basal sediment: Theory and application to Ice Stream B, Antarctica. *Journal of Geophysical Research*, 94(B4): 4071–4087, 1989.

D. R. MacAyeal, V. Rommelaere, P. Huybrechts, C. L. Hulbe, J. Determann, and C. Ritz. An ice-shelf model test based on the Ross Ice Shelf, Antarctica. *Annals of Glaciology*, 23:46–51, 1996.

D.R. MacAyeal. EISMINT: Lessons in Ice-Sheet Modeling. *Department of Geophysical Sciences, University of Chicago*, 1996b.

L. W. Morland. Unconfined ice shelf flow. In C. J. Van der Veen and J. Oerlemans, editors, *Dynamics of the West Antarctic Ice Sheet*. Dordrecht, Reidel Publishing Company, 1987.

Michael Oppenheimer and Richard Alley. Report of the workshop on ice sheet modelling. Available at www.scar.org/researchgroups/physicalscience/icesheetrpt.pdf, 2007.

W. S. B. Paterson. *The Physics of Glaciers*. Oxford, 3rd edition, 1994.

F. Pattyn. A new three-dimensional higher-order thermomechanical ice-sheet model: basic sensitivity, ice-stream development and ice flow across subglacial lakes. *Journal of Geophysical Research (Solid Earth)*, 108(B8):2382, 2003. doi: 10.1029/2002JB002329.

F. Pattyn and T. Payne. ISMIPHOM: Ice sheet model intercomparison project. Available at http://homepages.ulb.ac.be/ fpattyn/ismip/ismiphom.pdf, 2006.

F. Pattyn et al. Benchmark experiments for higher-order and full-Stokes ice sheet models (ISMIP-HOM). *The Cryosphere*, 2:95–108, 2008.

NA Phillips. A coordinate system having some special advantages for numerical forecasting. *Journal of the Atmospheric Sciences*, 14(2):184–185, 1957.

D. Pollard and R.M. DeConto. A coupled ice-sheet/ice-shelf/sediment model applied to a marine margin flowline: forced and unforced variations. *SPECIAL PUBLICATION-INTERNATIONAL ASSOCIATION OF SEDIMENTOLOGISTS*, 39:37, 2007.

W. H. Press, B. P. Flannery, S. A. Teukolsky, and V. T. Vetterling. *Numerical Receipes in FORTRAN. The Art of Scientific Computing. Second Edition.* Cambridge University Press, Cambridge, 1992.

R.S. Pressman and D. Ince. *Software engineering: a practitioner's approach.* McGraw-Hill New York, 2005.

IC Rutt, M. Hagdorn, NRJ Hulton, and AJ Payne. The Glimmer community ice sheet model. *Journal of Geophysical Research-Earth Surface*, 114(F2):F02004, 2009.

C. Schoof. Variational methods for glacier flow over plastic till. *Journal of Fluid Mechanics*, 555:299–320, 2006.

M.K. Seager. A SLAP for the masses. *Parallel Supercomputing: Methods, Algorithms and Applications*, pages 135–155, 1989.

Thomas, R.H., D.R. MacAyeal, D.H. Eilers and D. R. Gaylord. Glaciological studies on the ross ice shelf, antarctica: 1973-1978. *Antarctic Research Series*, 42:21–53, 1984.

C. J. van der Veen. *Fundamentals of glacier dynamics*. A. A. Balkema, Rotterdam, 1999.

CJ Van der Veen. Numerical modelling of ice shelves and ice tongues. In *Annales geophysicae. Series B. Terrestrial and planetary physics*, volume 4, pages 45–53, 1986.

H.K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics*. Prentice Hall, Harlow, England, 1995.

Robert T. Watson, editor. *Climate Change 2001: Synthesis Report: Third Assesment Report of the Intergovermental Panel on Climate Change*. Cambridge University Press, Cambridge, England, 2001.

Robert T. Watson, editor. *Climate Change 2007: Synthesis Report: Fourth Assesment Report of the Intergovermental Panel on Climate Change*. Cambridge University Press, Cambridge, England, 2007.

J. Weertman. Deformation of floating ice shelves. *Journal of Glaciology*, 3(21):38–42, 1957.

M. Weis, R. Greve, and K. Hutter. Theory of shallow ice shelves. *Continuum Mechanics and Thermodynamics*, 11:15–50, 1999.