2009

Modeling Ice Streams

Aitbala Sargent

MODELING ICE STREAMS

by

Aitbala Sargent

Kandidat Nauk, Lomonosov Moscow State University,1986

M.A. in Economics, University of Maine, 1977

M.S. in Computer Science, University of Maine, 2006

A THESIS

submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

(in Computer Science)

May 2009

MODELING ICE STREAMS

By Aitbala Sargent

Thesis Advisor: Dr. James L. Fastook

An Abstract of the Thesis Presented

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

(in Computer Science)

May 2009

Modeling glacier and ice sheet flow is a computationally challenging problem. The most challenging part in simulating ice sheet flow is modeling the fastest moving part of ice sheets, ice streams. In the first part of the thesis, we have constructed two numerical models of isothermal ice stream flow, a three-dimensional full-Stokes ice-sheet/ice-stream/ice-shelf model and a modified MacAyeal-Morland ice-stream/ice-shelf model. In the second part of the thesis, we studied the possibility of using SuperLU-DIST multiprocessor software package for solving the systems of linear equations generated by the model.

The uniqueness of the modified MacAyeal-Morland model is in its inclusion of the basal shear friction in the derivation of the equations. In the original MacAyeal-Morland equations, the shear friction is not included in the fundamental formulation but instead is added as a small correction to the final equations. Inclusion of the basal friction in the derivation generates equations that contain a term that depends on the bed gradients; that is, it generates equations that show how the ice stream flow may depend on the bed topography. To validate the model, the European Ice Sheet Modeling Initiative 1 intercomparison test is conducted and the results are compared with the results generated by MacAyeal (1994).

The three-dimensional full-Stokes model includes all higher-order stress gradients in the force-balance equation. To validate the full-Stokes model, experiments demonstrating the importance of the inclusion of all higher order stresses in the model, such as simulation of the evolution of an ice stream within the ice sheet and simulation of iceberg profiles, are conducted.

The computational demands of the full-Stokes model do not allow us using it in large problem domains. To solve this problem, application of SuperLU-DIST multiprocessor software package has been examined. The software's performance characteristics have been explored and benchmarked on the matrices generated by the three-dimensional full-Stokes model. The performed tests indicate that for the big-size matrices computations may not be stable. However, we have shown that it is possible to improve stability of the algorithm by using a priori knowledge of the matrix and permuting rows prior to applying the algorithm.

ACKNOWLEDGMENTS

I would like to thank:

James Fastook for giving me an invaluable opportunity to work on extremely interesting projects over the past years, for his guidance, unfailing support and help;

Rodney Jacobs for designing and writing clear and efficient interfaces to UMFPACK and serial SuperLU packages that were used to solve systems of linear equations generated by our models;

L. W. Morland and Douglas MacAyeal for their work on the ice shelf and ice-stream modeling that served as an inspiration for the work in Chapter 3 of this work;

Christina Hulbe and Jesse Johnson for the interest they have taken and the suggestions they have provided during work on this dissertation;

Ted Scambos for giving a data set that made validation of a model from chapter 3 possible;

Terry Hughes and George Markowsky for offering opportunities and support; for genuine interest in my career;

my entire committee, James Fastook, Phillip Dickens, Terence Hughes, Peter Koons, George Markowsky, and Jesse Johnson for helpful and positive reviews;

my husband, Louis Sargent, and my daughter, Aiesha, for their understanding and endless support; and

the NSF and the CReSIS for financial support.

Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Ice streams are fast flowing parts within ice sheets. They are typically hundreds of kilometers long and tens of kilometers wide, and flow at velocities of up to several kilometers per year. While ice streams account for only about ten percent of the ice sheet volume, they are key to understanding ice sheet stability. Ice streams drain the majority of ice from ice sheets. In Antarctica, it is estimated that as much as 90% of ice sheet discharge is via ice streams. While the East Antarctic ice sheet is stable, there is a debate whether the West Antarctic ice sheet might decay in the future. If that happens, it could raise the sea level by about 5 m [5].

In order to answer the question what is the likely future of West Antarctica and Greenland, the flow behavior of ice sheets must be understood. Numerical modeling contributes to our understanding of the ice sheets, yet the numerical models of ice streams have substantial limitations varying from a need for better physics to overcoming the demand for larger computing power. The aim of this work is to contribute to modeling ice streams and to solving the computing problems associated with it.

In the introduction, I will review particular components of the multifaceted ice-flow system and the classical ways of modeling them. Since the work done in this thesis is an extension of the University of Maine Ice Sheet Model (UMISM), the introduction gives also an overview of UMISM.

1.1 Ice-flow System: Ice-sheets, Ice-streams, and Ice-shelves

Around 90% of the world's land ice is concentrated in the Antarctic ice sheets occupying over 13 million square kilometers and reaching thickness of a few kilometers. Ice flows as a highly viscous solid from the central parts of the continent, where the ice thickness is greatest, towards the margins, which are near the coast and feed into the floating ice shelves. The mass of ice is maintained by snowfall over the continent, while mass loss from the ice sheets occurs mostly through the calving of icebergs at its margins.

At the central part of the Antarctic continent, ice flow occurs essentially as a result of the ice sheet spreading under its own weight, thus, ice behaves as a highly viscous solid and flows by creep deformation. This types of flow is called *ice sheet flow*.

Some parts of Antarctic ice sheets, called *ice streams*, are found to move at much higher speeds than the surrounding parts of ice sheets. The observed high speeds cannot be explained by creep deformation alone. Researchers assume that there is a sliding movement of ice at the underlying bedrock. As the base of an ice sheet approaches the pressure melting point, a thin and possibly patchy film of water can form between the ice and its bed. Such a water film weakens the contact between ice and bed and may allow for a 'sliding' velocity. This type of flow is called *ice stream flow*.

The third type of flow is called *ice shelf flow*. An ice shelf is a large sheet of ice floating on the sea but attached to land or to a grounded ice sheet. Ice shelves range in thickness from about 50 to 600 meters. The place where the ice starts to float is called the *grounding line*. Ice shelves surround much of Antarctica.

To model particular components of these multifaceted ice flow systems, different types of models have been used: models of grounded ice sheet flow, models of the ice shelves, and models of ice streams. The latter ones are divided into two different groups, the first are based on the models of ice shelves, and the second are models which include higher-level stresses in the force-balance equation.

1.2 Numerical Modeling of Ice-sheets, Ice-streams, and Ice-shelves

Numerical modeling of glaciers and ice sheets generally involves a number of simplifications with respect to the physics of the ice mass.

Most ice-sheet models are based on the so-called *shallow-ice approximation* (SIA)[1] [37] which is valid for an ice mass with a small aspect ratio (ice thickness $\ll$ ice horizontal dimensions). SIA is used in the ice dynamics component of UMISM which is described below. In this approximation, longitudinal and transverse stress gradients are neglected. However, the SIA is not valid at all places in an ice sheet, such as at the ice divide or near the ice-sheet margins. Shallow ice approximation is also not valid to model ice-streams or ice-shelves, where inclusion of longitudinal stresses is especially important.

Almost all numerical models of ice-shelves solve stress-balance equations [49] that are derived with the assumption that vertical shear is negligible; that is, that the horizontal velocity does not vary with depth. In this approximation, the only stresses considered are the longitudinal stresses. Since the ice shelf is supported by water, the basal drag is not included in the fundamental formulation.

[1]Derivation of SIA model is given in B on page 137.

This assumption allowed Morland to integrate out the vertical dimension and reduce three-dimensional equations to two-dimensional equations.

Since ice-stream flow is transitional between ice sheet flow and ice shelf flow, both basal shear and longitudinal stresses are important to consider. For solving ice streams, a common approach, suggested by MacAyeal, is to treat the ice streams as barely-grounded ice shelves. MacAyeal modified Morland equations for ice-shelves by adding a term simulating the basal drag. He used a heuristic assumption that the basal drag is proportional to velocities. The resulting model is called MacAyeal-Morland model for ice streams. Although the model generates credible results, the fact that the term for basal drag was added to the equation after integrating the vertical dimension with the assumption of no basal drag makes the equations not self-consistent.

Both the SIA and the barely-grounded ice-shelf models involve a number of simplifications with respect to the physics of the ice mass. However, they are not valid at all places in an ice sheet. For example, near the margin of the ice sheet (at grounding lines, outlet glaciers, and ice streams) or at the transition zones between different types of ice flow, all stresses in the force balance become equally important. The only way to properly account for all stresses is to solve the full momentum equation with none of the limiting assumptions that go into either the shallow-ice or the barely-grounded ice shelf approximations. Examples of the models that take into account the higher-order stresses can be found in [53]. The models that take into account all stresses in the force balance are called the full-Stokes models. However, very few three-dimensional full-Stokes models exist. Among them are the models of Pattyn [52], Martin [47], Zwinger [62], and Price [55]. Since solving the 3-D full-Stokes equations demands huge computational power, most of the higher-order models make some simplifications to reduce complexity of the model. The most common approximation is to ”introduce the two horizontal velocity components as field variables. This leads to an elliptic system with two rather than four variables of the full system at points in three-dimensional space [51], [32], and the resulting linear systems are generally better conditioned than those resulting from the numerical analysis of the full system” [53].

Another common approximation is the scheme used by Blatter [6] and Pattyn [51]. In these approximation, higher-order longitudinal and transverse stresses are included in the force balance equation but the variational stresses are neglected $\left(\frac{\partial\sigma_{zx}}{\partial x}=0,\ \frac{\partial\sigma_{zy}}{\partial y}=0\right)$; that is, the conservation of momentum equation (see (2.15a)-(2.15c) on page 18) is reduced to the form:

$$\begin{aligned}
\frac{\partial\sigma_{xx}}{\partial x}+\frac{\partial\sigma_{xy}}{\partial y}+\frac{\partial\sigma_{xz}}{\partial z}&=\rho g_x,\\
\frac{\partial\sigma_{yx}}{\partial x}+\frac{\partial\sigma_{yy}}{\partial y}+\frac{\partial\sigma_{yz}}{\partial z}&=\rho g_y,\\
\frac{\partial\sigma_{zz}}{\partial z}&=\rho g_z.
\end{aligned}$$

This allows researchers to reduce three-dimensional problem to a computationally two-dimensional problem.

Truly three-dimensional thermomechanically coupled ice sheet models including all higher-order stress gradients, or full-Stokes models, are not widely used. The reason for this is in the complexity of the model description, the difficulty in obtaining a numerically stable result, and the high computational cost – a substantial increase in their complexity drastically affects the ability of a model to perform millennia-scale climate experiments.

Thus, to summarize above, the most challenging part in simulating ice sheet flow is modeling the fastest moving part of ice sheets, ice streams. Credible predictions of ice stream evolution require taking into account the higher-order stresses in the force balance equation. However, the three dimensional models of ice streams that include all higher-order stresses have huge demands on computer time. To solve this problem, the goal of this work was

1. to construct a model or models that can approximate ice stream flow, or different types of ice flow, including ice streams, and
2. to explore the efficiency of using parallel programming to overcome the high computational cost of solving big systems of linear equations generated by higher-order full-Stokes models.

1.3 The University of Maine Ice Sheet Model

The work done in this thesis is made possible by the prior efforts of Dr. James Fastook and his students. I have extended an existing ice sheet model by adding modules that simulate ice-streams. Below is an overview of the fundamental building blocks of the existing ice sheet model. This overview follows Johnson [41].

The UMISM has its origin in the 1980s. Early works on using finite element method to model ice sheet flow appear in Fastook & Schmidt [18], Fastook & Hughes [14], Fastook & Hughes [15], and Fastook [20]. Then, a flow band or one dimensional model for ice sheets was developed in Fastook [20, 21, 22], and Fastook & Hughes [16]. This model was programmed in FORTRAN and ran on an IBM mainframe. The data sets represented a problem domain with about 50 nodes.

In 1989, a two-dimensional, map-plane model for ice sheets was developed [Fastook & Chapman [11]]. The program used the finite element method to solve the continuity equation for ice deformation. Portions of the code written for this model are still in the ice sheet model that is run today. When it was created it ran on an IBM 360 mainframe. Runs typically included 150 nodes and the output was directed to a Tektronix graphics terminal.

Starting 1992, the model included calculations of internal temperatures from which mechanical properties could be derived [Fastook [23, 24, 25]]. The model was applied to glaciological problems in [13], [19, 26, 12], and [27]. Climatology for the model was developed with Fastook & Prentice [17].

Then, Johnson and Fastook modified the model to include an accounting of basal melt water. Inclusion of the basal water component allowed the model to identify all major lakes in Antarctica. Results were published in [41] and [29, 28]. This major improvement allowed the model itself to specify where and when the sliding would occur.

The model is also used to reconstruct ice-sheet evolution on the flanks of the large Tharsis Montes volcanos on Mars [30].

One of the most important applications of the ice sheet model was participation in EISMINT (European Ice Sheet Modeling INiTiavite). This initiative established a baseline for results from ice sheet models to assure that they produce similar results. The initiative considered applications to the Greenland and Antarctic ice sheets, thermo-mechanical coupling, grounding line treatments, and ice shelf models. The results appear in [38]. Involvement in the EISMINT experiment establishes credibility for a model's output and allows it to be used for new applications.

The model in its current form runs on both Linux, Mac OSX, and SGI platforms. The source code is (mostly) ANSI FORTRAN. The screen output uses Open GL, an industry standard for graphics. There are a number of data filters for postscript output of maps. We are currently able to model upward of 70,000 nodes in our map plane model for ice sheets.

The model predicts the ice thickness, velocity, and temperature of glaciers as functions of position and time. Inputs to the model are climate conditions, temperatures and precipitation rates, bed conditions, and elevations and sliding characteristics. Figure 1.1 shows the major components of the model and the data flow among them.

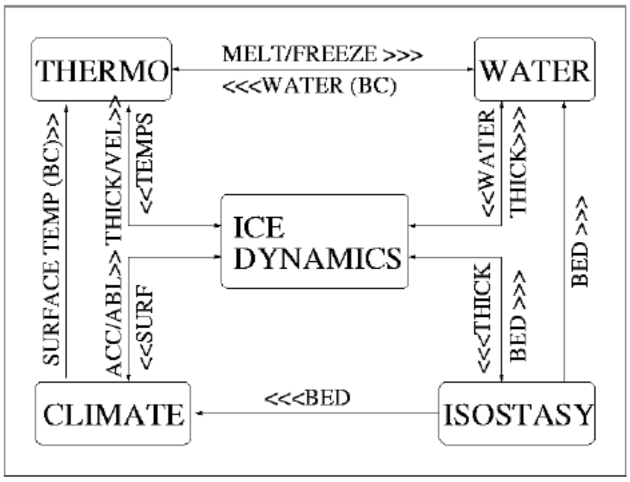

Figure 1.1. Components of UMISM. Provided by James Fastook.

The ice dynamics component is at the core of the model. It predicts ice thickness and ice velocity using accumulation - ablation rates generated by the climate module, ice temperatures generated by the thermodynamics module, the presence or absence of water at the bed generated by the basal water module, and bed elevation generated by the isostasy module. Ice temperature is important in determining the ice reaction to the applied forces; cold ice is harder than warm ice and is deformed at a slower rate. The ice dynamics module also uses the boundary condition characteristics between the ice and the earth. Finally, the weight of the ice depresses the ground, which lowers the surface elevation of the ice. The isostasy module computes the amount of bed depression.

Climate conditions at the surface of the ice depend upon surface elevation because temperature decreases with increasing altitude. The climate module uses the surface elevation generated by the ice dynamics and isostasy modules along with a climate model to predict surface temperatures, melting rates, and precipitation rates. The thermodynamics module uses surface temperature as well as basal conditions and geothermal heating to compute temperature throughout the ice sheet. In addition, deformation of the ice due to movement also produces heat. The water module uses bed characteristics from the isostasy module and basal temperatures to predict the presence of water.

The ice dynamics module uses partial differential equations (PDEs) derived from mass and momentum conservation principals as a basis for computing ice thickness and velocity. The thermodynamics module uses PDEs derived from energy conservation principals as a basis for computing ice temperatures. Combined with constitutive relationships that relate ice strain rates to temperature, and temperature to amount of heat, a complete system is formed for doing the fundamental calculations of ice thickness and velocity. The resulting PDEs are solved numerically using a mathematical technique known as the finite element method (FEM).

1.4 Overview of Thesis

This work is organized as follows. Chapter 1 is a short review of particular components of the multifaceted ice-flow system and a review of the University of Maine ice sheet model. Chapter 2 gives a review of the basic conservation laws and the constitutive relations that describe ice flow.

Chapter 3 details two models that have been constructed to study the flow of ice; a full-Stokes model that takes into account all stresses and an ice-shelf/ice-stream model that is a modification of MacAyeal-Morland model. In construction of the modified MacAyeal-Morland model, we included shear friction (proportional to the driving stress or velocities) in the derivation of the equation. In the original Morland equations [46], the basal drag is not included in the fundamental formulation but instead is added as a small correction, proportional to speed, to the final equations. The higher-order model and verification of the model have been presented at [57]. The modified MacAyeal-Morland model and its verification has been presented in [56].

Chapter 4 describes the finite element method discretization of the partial differential equations from Chapter 3.

Chapter 5 describes application and benchmarking of the multiprocessor software package SuperLU-DIST for solving large systems of sparse simultaneous linear equations generated by the three-dimensional higher-order model. The results of this chapter have been discussed and published in [40] and [58].

Chapter 6 is devoted to testing and verification of the models. A portion of the results has been published in [59]. We contributed to this paper by running the numerical calculations which supported the discussions and conclusion.

The concluding remarks in Chapter 7 close the thesis.

Chapter 2

MATHEMATICAL BACKGROUND - STOKES' EQUATIONS FOR ICE FLOW

This chapter introduces the basic conservation laws that are used to describe ice flow. The variables that describe ice sheet are ice thickness, ice velocity, the various stress components, and temperature at selected points. In this work we assume that ice flow is isothermal, that is, ice temperature is fixed and uniform. The other variables are found from conservation of mass, conservation of momentum, and the constitutive relation that are discussed below.

2.1 Constitutive Relation

Constitutive relations describe some property of the material. The Glen flow law for ice is a fundamental constitutive relation relating stress and strain rates:

$$\dot{\epsilon}_{ij} = R\left(\frac{\sigma'_{ij}}{B}\right)^n,$$

where R is a strain rate scalar and B is ice viscosity. Derivation of the flow law of ice can be found in [35].

A stress is a force per unit area applied to a surface. It is denoted by σ_{ij}, where the first subscript denotes the direction of the stress or force, and the second subscript denotes the direction of the normal to the surface on which the force is acting. Hydrostatic pressure changes the size of an object but not its shape. Changes in shape require non-hydrostatic stresses. Since flow of a glacier is caused by non-hydrostatic stresses, we will write the equations in terms of these stresses. By subtracting the mean stress,

$$P = \frac{1}{3}\left(\sigma_{xx} + \sigma_{yy} + \sigma_{zz},\right) \tag{2.1}$$

from the total stress, the non-hydrostatic stress, or so called deviator stress σ'_{ij}, is obtained:

$$\sigma'_{ij} = \sigma_{ij} - \delta_{ij}P, \tag{2.2}$$

where δ_{ij} is the Kronecker delta.

It is this component of the overall stress field acting on the glacier that produces the deformation.

The deviator stress tensor has three invariants, the first two are as follows:

first invariant: $$J_1 = \sigma'_{xx} + \sigma'_{yy} + \sigma'_{zz} = 0, \tag{2.3a}$$

second invariant or the effective stress: $$J_2 = (\sigma_e)^2 = \frac{1}{2}\sigma'_{ij}\sigma'_{ij}. \tag{2.3b}$$

In a deformable medium, stresses induce deformation or strain. Given a Cartesian coordinate system (x, y, z) and a velocity field $\vec{u} = (u_x, u_y, u_z)$, the strain rate tensor, $\dot{\epsilon_{ij}}$, is defined as follows:

$$\dot{\epsilon}_{ij} = \frac{d\epsilon_{ij}}{dt} = \frac{1}{2}\left(\frac{\partial u_i}{\partial j} + \frac{\partial u_j}{\partial i}\right) = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right). \tag{2.4}$$

The strain rate tensor has three invariants, the first two have special names:

incompressibility condition: $$I_1 = \dot{\epsilon_{xx}} + \dot{\epsilon_{yy}} + \dot{\epsilon_{zz}} = 0 \quad , \tag{2.5a}$$

effective strain rate: $$I_2 = \dot{\epsilon}_e^2 = \frac{1}{2}\dot{\epsilon}_{ij}\dot{\epsilon}_{ij} = \frac{1}{2}(\dot{\epsilon}_{xx}^2 + \dot{\epsilon}_{yy}^2 + \dot{\epsilon}_{zz}^2 + 2\dot{\epsilon}_{xy}^2 + 2\dot{\epsilon}_{xz}^2 + 2\dot{\epsilon}_{yz}^2). \tag{2.5b}$$

Obtained experimentally from laboratory observations as well as measurements in actual glaciers [31, 34], the Glen flow law is a non-linear relation between strain rate and stress:

Glen flow law: $$\dot{\epsilon}_{ij} = R\left(\frac{\sigma'_{ij}}{B}\right)^n , \tag{2.6}$$

where B is a temperature-dependent measure of the ice hardness, the exponent $n = 3$ is experimentally determined for ice [33].

If ice is assumed to be incompressible and isotropic, then the Glen flow law can be linearized as follows:

$$\dot{\epsilon}_{ij} = \frac{\sigma_e^{n-1}}{B^n}\sigma'_{ij}, \tag{2.7}$$

Linearizing the constitutive relation will be critical for the numerical solution of the resulting differential equations.

Solving equation (2.6) for σ_e and substituting into equation (2.7) gives

$$\dot{\epsilon}_{ij} = \frac{(\dot{\epsilon}_e)^{\frac{n-1}{n}}}{B}\sigma'_{ij}, \tag{2.8}$$

which can be solved for σ'_{ij} as:

$$\sigma'_{ij} = 2\mu\dot{\epsilon}_{ij} \tag{2.9}$$

where $2\mu = B\left(\dot{\epsilon}_e\right)^{\frac{1-n}{n}}$ is the effective viscosity. Viscosity is the material property that relates stress and strain rates in a linear fluid. The dependence of the effective viscosity on the strain rate invariant means that μ will be spatially non-uniform and dependent upon the solution itself. This approach is common with non-linear problems and requires an iterative solution.

2.2 Conservation of Mass

2.2.1 Conservation of Mass Equation

The mass conservation, or the continuity equation, states that the change in mass is equal to the gradient of the flux of material into the region. This is expressed in terms of the density as

$$\textit{continuity equation:} \quad \frac{\partial\rho}{\partial t} + \nabla\cdot\left(\rho\vec{\mathbf{u}}\right) = 0, \tag{2.10}$$

or

$$\frac{\partial\rho}{\partial t} + \left(\frac{\partial\left(\rho u_x\right)}{\partial x} + \frac{\partial\left(\rho u_y\right)}{\partial y} + \frac{\partial\left(\rho u_z\right)}{\partial z}\right) = 0.$$

If the density is constant, implying that the material is incompressible, this becomes

$$\nabla\cdot\left(\vec{\mathbf{u}}\right) = 0. \tag{2.11}$$

The principle of conservation of mass can be used to write an equation in terms of the variation in thickness of an ice mass over time. Figure 2.1 demonstrates conservation of mass in a column of ice – change in ice surface elevation is balanced by accumulation or ablation of ice and the divergence of ice flux. Consider a column of ice moving with average velocity $\vec{\mathbf{U}}$. Denote the mass flux by $\vec{\mathbf{q}} = \vec{\mathbf{U}}h$, where $\vec{\mathbf{U}} = (U_x, U_y)$ is the depth averaged velocity of the column of ice and h is the thickness of ice. Then the thickness h of the ice varies over time t as

$$\textit{prognostic equation:} \quad \frac{\partial h}{\partial t} = \dot{a} - \frac{\partial q_x}{\partial x} - \frac{\partial q_y}{\partial y} \tag{2.12}$$

where q_i is the x or y component of the mass flux and $\dot{a}$ is the surface accumulation rate, which is the ice-equivalent of snowfall in meters per year.

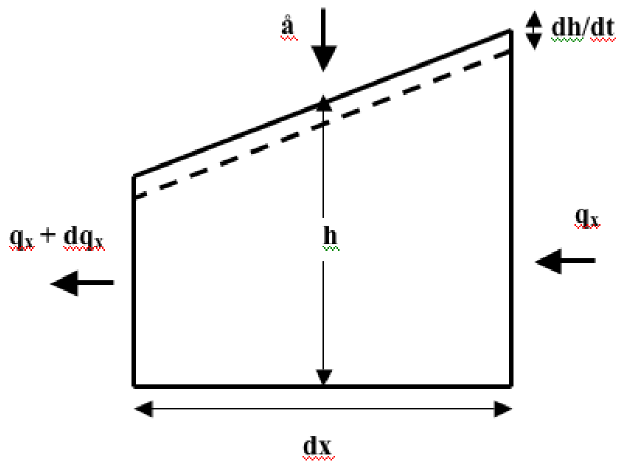

Figure 2.1. Ice sheet mass balance for a column of ice on a horizontal bed. The column has width dy, length dx, and mean height h. The change in ice surface elevation with time $\frac{\partial h}{\partial t}$ is balanced by accumulation or ablation of ice ($\dot{a}$ is accumulation/ablation rate) and the divergence of ice flux (difference between influx and outflux of ice, q_x and $q_x + dq_x$).

2.2.2 Mass Equation Boundary Conditions

The mass conservation equation (2.12) requires boundary conditions to be specified along the edge of the domain. There may be two types of the boundary conditions:

Newman (flux is specified) conditions may be imposed at the inland ice dome (flux is zero, $q_0 = 0$,), at the head of the ice stream (flux q_0 is given), or at the ice front (a free-radiation condition can be applied to avoid having ice ”pile up”, $q_0 = \left(h\vec{U}\right)|_- \cdot \vec{n}$, where $\left(h\vec{U}\right)|_-$ represents the ice transport just upstream of the ice front):

$$h\left(U_x n_x + U_y n_y\right) = h\vec{U} \cdot \vec{n} = \vec{q} \cdot \vec{n} = q_0, \tag{2.13}$$

where $\vec{n}$ is outward-pointing unit normal vector.

Dirichlet (ice thickness is specified) conditions may be imposed, for example, at the ice stream:

$$h = h_0. \tag{2.14}$$

2.3 Conservation of Momentum

2.3.1 Conservation of Momentum Equations

Conservation of momentum is Newton’s second and third laws which state that linear momentum is conserved if the sum of forces on an object are equal to zero.

Figure (2.2) on Page 18 shows the forces acting on a glacier in one direction, say x. The driving stress directed in the direction of decreasing surface slope ($\rho\, g_x$) is resisted by longitudinal deviator stresses (compressions and tensions from up and down of glacier – σ_{xx}), basal drag (friction generated at the bed – σ_{xz}), and lateral drags (frictions generated at the sides of the glacier – σ_{xy}). Similar forces are acting in y and z directions.

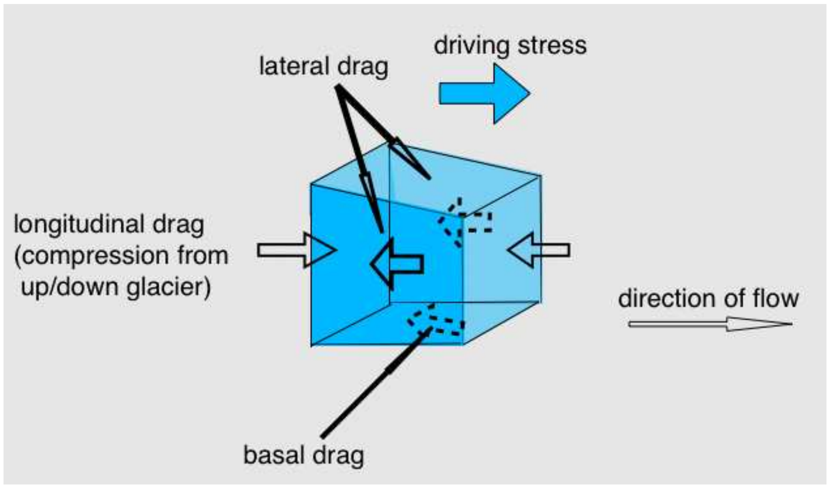

Figure 2.2. Ice Sheet: forces acting on ice in one direction.

If we assume that ice is not accelerating or decelerating, than balancing all the applied stresses acting on the various surfaces of a differential volume $dx\, dy\, dz$ with the body forces due to gravity, the linear momentum equations look as follows:

$$\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + \frac{\partial \sigma_{xz}}{\partial z} = \rho g_x, \tag{2.15a}$$

$$\frac{\partial \sigma_{yx}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{yz}}{\partial z} = \rho g_y, \tag{2.15b}$$

$$\frac{\partial \sigma_{zx}}{\partial x} + \frac{\partial \sigma_{zy}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} = \rho g_z. \tag{2.15c}$$

These momentum equations are called the *diagnostic equations*.

Note that the applied stresses are symmetric, $\sigma_{ij} = \sigma_{ji}$. This prevents free rotation with uniform motion. Symmetry of stresses reduces the problem of finding nine stresses to that of finding six stresses.

Using the fact that $g_x = 0$ and $g_y = 0$ and relations (2.1) and (2.2), the momentum equations (2.15a)-(2.15c) in terms of deviatoric stresses could be written as follows:

$$\frac{\partial\left(\sigma'_{xx} + P\right)}{\partial x} + \frac{\partial \sigma'_{xy}}{\partial y} + \frac{\partial \sigma'_{xz}}{\partial z} = 0, \tag{2.16}$$

$$\frac{\partial \sigma'_{yx}}{\partial x} + \frac{\partial\left(\sigma'_{yy} + P\right)}{\partial y} + \frac{\partial \sigma'_{yz}}{\partial z} = 0, \tag{2.17}$$

$$\frac{\partial \sigma'_{zx}}{\partial x} + \frac{\partial \sigma'_{zy}}{\partial y} + \frac{\partial\left(\sigma'_{zz} + P\right)}{\partial z} = \rho g. \tag{2.18}$$

Taking into account the constitutive relations between stresses and strain rates (2.9), the momentum equations (2.16)-(2.18) can be written in terms of strain rates as follows:

$$\frac{\partial\left(2\mu\dot{\epsilon_{xx}} + P\right)}{\partial x} + \frac{\partial\left(2\mu\dot{\epsilon_{xy}}\right)}{\partial y} + \frac{\partial\left(2\mu\dot{\epsilon_{xz}}\right)}{\partial z} = 0, \tag{2.19a}$$

$$\frac{\partial\left(2\mu\dot{\epsilon_{yx}}\right)}{\partial x} + \frac{\partial\left(2\mu\dot{\epsilon_{yy}} + P\right)}{\partial y} + \frac{\partial\left(2\mu\dot{\epsilon_{yz}}\right)}{\partial z} = 0, \tag{2.19b}$$

$$\frac{\partial\left(2\mu\dot{\epsilon_{zx}}\right)}{\partial x} + \frac{\partial\left(2\mu\dot{\epsilon_{zy}}\right)}{\partial y} + \frac{\partial\left(2\mu\dot{\epsilon_{zz}} + P\right)}{\partial z} = \rho g. \tag{2.19c}$$

Equations (2.19a)-(2.19c) can be written in a compact form as

$$T_{ij,j} - \rho g \delta_{i3} = 0, \tag{2.20}$$

where T is a tensor

$$\mathbf{T} = \begin{bmatrix} 2\mu\dot{\epsilon}_{xx} + P & 2\mu\dot{\epsilon}_{xy} & 2\mu\dot{\epsilon}_{xz} \\ 2\mu\dot{\epsilon}_{xy} & 2\mu\dot{\epsilon}_{yy} + P & 2\mu\dot{\epsilon}_{yz} \\ 2\mu\dot{\epsilon}_{xz} & 2\mu\dot{\epsilon}_{yz} & 2\mu\dot{\epsilon}_{zz} + P \end{bmatrix} \tag{2.21}$$

These differential equations, 2-nd order in terms of velocities, are the equations that we must solve.

2.3.2 Momentum Equation Boundary Conditions

For the momentum equations, the boundary conditions must be specified at the sides of the domain, at the surface of the domain, and at the bed of the domain. Let's discuss them separately.

2.3.2.1 Surface Boundary Conditions

Regardless of the type of flow (ice-sheet, ice-stream, or ice-shelf), the boundary conditions at the surface of the ice, $z = z_s$, is assumed to be stress-free, *i.e.*,

$$\mathbf{T} \cdot \mathbf{n_s} = 0. \tag{2.22}$$

where $\mathbf{n_s}$ is the outward-pointing unit normal vector given by

$$\mathbf{n_s} = \frac{-\frac{\partial z_s}{\partial x}\mathbf{n_x} - \frac{\partial z_s}{\partial y}\mathbf{n_y} + \mathbf{n_z}}{\sqrt{1 + (\frac{\partial z_s}{\partial x})^2 + (\frac{\partial z_s}{\partial y})^2}} \tag{2.23}$$

Application of the tensor/vector product in (2.22) gives the following three equations that must be satisfied at $z = z_s$:

$$\left(2\mu\dot{\epsilon}_{xx} + P\right)\frac{\partial z_s}{\partial x} + 2\mu\dot{\epsilon}_{xy}\frac{\partial z_s}{\partial y} - 2\mu\dot{\epsilon}_{xz} = 0, \tag{2.24a}$$

$$2\mu\dot{\epsilon}_{yx}\frac{\partial z_s}{\partial x} + \left(2\mu\dot{\epsilon}_{yy} + P\right)\frac{\partial z_s}{\partial y} - 2\mu\dot{\epsilon}_{yz} = 0, \tag{2.24b}$$

$$2\mu\dot{\epsilon}_{zx}\frac{\partial z_s}{\partial x} + 2\mu\dot{\epsilon}_{zy}\frac{\partial z_s}{\partial y} - \left(2\mu\epsilon_{zz} + P\right) = 0. \tag{2.24c}$$

2.3.2.2 Basal Boundary Conditions

The boundary condition at the bed of the ice, $z = z_b$, is not stress-free and is different for different types of flow.

For ice-sheets, where the bed is frozen, Dirichlet boundary conditions are the obvious choice, as the velocity is zero and can be specified as such.

For ice-shelves, boundary conditions at the bed can be derived from the assumption that the ice is floating and that the stress can be specified to be equal to the hydrostatic pressure necessary to float the ice shelf.

For ice-streams, boundary conditions should be different since ice is not floating. The velocities cannot be specified because they are unknown. But the resistive pressure at the bed is also unknown. We know that this resistive stress has a value between the driving stress (if it equals the driving stress, we have the shallow-ice approximation) and the hydrostatic pressure of the floating ice (if it is equal to the hydrostatic pressure necessary to float the ice, we have the ice-shelf approximation).

There are two ways to specify the basal boundary conditions. One is to specify the basal resistive stress as some fraction of the driving stress which we use in this work. This approach does produce the concave profile characteristic of an ice stream but the fraction (which is a model parameter) is hard to specify.

A second approach is to use a boundary-layer. This approach has been studied by Debra Kenneway for a two-dimensional version of the three-dimensional system which models a vertical slice through the ice sheet along a flowline. In this approach, a thin layer between the ice and the bed is introduced and zero velocity Dirichlet boundary conditions are imposed at the bottom boundary of this layer. To simulate sliding at the bed, greater deformation is allowed within the boundary layer. Thus, the boundary layer can be interpreted as deformable till or slush (water-saturated ice at the melting point). This approach also has some disadvantages – the geometry (thickness) and the mechanical properties (how soft the layer is) are as difficult to specify as is the fraction of the driving stress.

Below is derivation of the basal boundary conditions under assumption that the basal resistive stress is some fraction of the driving stress.

Let's assume that the resistive traction on the base has the following form:

$$\vec{f} = (f_x, f_y, f_z). \tag{2.25}$$

With this assumption, the basal boundary conditions can be specified as follows:

$$\mathbf{T} \cdot \mathbf{n_b} = \mathbf{f}\mathbf{n_b}, \tag{2.26}$$

where $\mathbf{n_b}$ is the outward-pointing unit normal vector to the bottom surface given by

$$\mathbf{n_b} = \frac{\frac{\partial z_b}{\partial x}\mathbf{n_x} + \frac{\partial z_b}{\partial y}\mathbf{n_y} - \mathbf{n_z}}{\sqrt{1 + (\frac{\partial z_b}{\partial x})^2 + (\frac{\partial z_b}{\partial y})^2}} \tag{2.27}$$

Application of the tensor/vector product in (2.26) gives the following three equations that must be satisfied at $z = z_b$:

$$(2\mu\dot{\epsilon_{xx}} + P)\,\frac{\partial z_b}{\partial x} + 2\mu\dot{\epsilon_{xy}}\frac{\partial z_b}{\partial y} - 2\mu\dot{\epsilon_{xz}} = f_x, \tag{2.28a}$$

$$2\mu\dot{\epsilon_{yx}}\frac{\partial z_b}{\partial x} + (2\mu\dot{\epsilon_{yy}} + P)\,\frac{\partial z_b}{\partial y} - 2\mu\dot{\epsilon_{yz}} = f_y, \tag{2.28b}$$

$$2\mu\dot{\epsilon_{zx}}\frac{\partial z_b}{\partial x} + 2\mu\dot{\epsilon_{zy}}\frac{\partial z_b}{\partial y} - (2\mu\dot{\epsilon_{zz}} + P) = -f_z. \tag{2.28c}$$

2.3.2.3 Side Boundary Conditions

The momentum equations (3.1a)-(3.1c) may have two types of side boundary conditions [36].

Dirichlet boundary conditions can be specified if velocities are known. For example, zero velocity can be specified in the areas of frozen bed, or areas where ice-shelves abut stagnant, zero slip coast lines, or areas where velocities are known from an experimental data sets.

Neumann boundary condition, specification of stress or force on the boundary, are usually applied at the seaward, iceberg-carving front. Multiplying stress by area on a boundary specifies force. Typically at the lateral sides of the domain, a pressure varying linearly with depth may serve as boundary conditions.

Chapter 3

MODELS

In this chapter, I will consider two models that have been constructed to study the flow of ice; a higher-order model that takes into account all stresses and ice-shelf/ice-stream Morland model. In construction of the Morland model, we included shear friction (proportional to driving stress) in the derivation of the equation. In the original Morland equations (see, for example, in [46], the basal drag is not included in the fundamental formulation but instead is added as a small correction (proportional to speed) to the final equations.

3.1 Three-Dimensional Full-Stokes Model

In the full-stress 3-dimensional model, we couple the mass (2.12) and momentum (2.16) - (2.18) conservation equations (the prognostic and diagnostic equations) that take into account all stresses. This allows us to simulate flow in regions where longitudinal stresses are important.

3.1.1 Conservation of Momentum Equation

Conservation of momentum equations (2.16) - (2.18) are written in terms of strain rates. To write the equations in terms of velocities and pressure, substitute (2.9) and (2.4) into the system (2.16) - (2.18). All stresses and strain rates are then derived quantities from the equations for velocities.

$$\frac{\partial\left(2\mu\frac{\partial u_x}{\partial x}) + P\right)}{\partial x} + \frac{\partial\left(\mu(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x})\right)}{\partial y} + \frac{\partial\left(\mu(\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x})\right)}{\partial z} = 0; \tag{3.1a}$$

$$\frac{\partial\left(\mu(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y})\right)}{\partial x} + \frac{\partial\left(2\mu\frac{\partial u_y}{\partial y}) + P\right)}{\partial y} + \frac{\partial\left(\mu(\frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y})\right)}{\partial z} = 0; \tag{3.1b}$$

$$\frac{\partial\left(\mu(\frac{\partial u_z}{\partial x} + \frac{\partial u_x}{\partial z})\right)}{\partial x} + \frac{\partial\left(\mu(\frac{\partial u_z}{\partial y} + \frac{\partial u_y}{\partial z})\right)}{\partial y} + \frac{\partial\left(2\mu\frac{\partial u_z}{\partial z}) + P\right)}{\partial z} = \rho g. \tag{3.1c}$$

These are three nonlinear, coupled, partial differential equations in terms of u_x, u_y, u_z, and P. Since this system has four variables (u_x, u_y, u_z, and P) and only three equations, we add the conservation of mass equation, expression of incompressibility (2.11), to the system:

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} = 0. \tag{3.2}$$

3.1.2 Conservation of Mass Equation

The conservation of mass equation (2.12) can be rewritten as

$$\frac{\partial h}{\partial t} = \dot{a} - \frac{\partial\left(hU_x\right)}{\partial x} - \frac{\partial\left(hU_y\right)}{\partial y}, \tag{3.3}$$

where U_x and U_y are components of the depth-averaged velocity of the column of ice.

Often, the numerical solutions of convective problems like (3.3) are corrupted by node-to-node oscillations in the regions where the solutions undergo rapid changes. One way to eliminate the oscillations is to severely refine the mesh. An alternative solution to eliminate oscillations is to add an artificial diffusion term to the equation. The drawback is a loss of accuracy (the artificial terms are only second-order accurate).

With an artificial diffusion term, the equation becomes as follows:

$$\frac{\partial h}{\partial t} = \dot{a} - \frac{\partial\left(hU_x\right)}{\partial x} - \frac{\partial\left(hU_y\right)}{\partial y} + \frac{\partial}{\partial x}\left(k_{xx}\frac{\partial h}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_{yy}\frac{\partial h}{\partial y}\right), \tag{3.4}$$

where $k_{xx} = \frac{\sqrt{2a^e}}{8}\left|\sum_{i=1,4} U^e_{x,i}\right|, \quad k_{yy} = \frac{\sqrt{2a^e}}{8}\left|\sum_{i=1,4} U^e_{y,i}\right|, \quad a^e$ - column-average finite element's area (which are discussed further on Page 44).

The addition of a diffusion term requires an additional boundary condition along with those already discussed on Page 17:

$$\nabla(kh)\cdot\bar{n} = k_{xx}\frac{\partial h}{\partial x}n_x + k_{yy}\frac{\partial h}{\partial y}n_y = 0. \tag{3.5}$$

3.1.3 Complexity of Equations

Thus, to solve the full 3-D models, we have to solve the system of five equations, (3.3) and (3.1a) - (3.2) for five variables, P, h, u_x, u_y, and u_z.

The most challenging part is solving the system of momentum equations (3.1a) - (3.2). This is a nonlinear system of four equations with four variables. The numerical approximation of the system results in iterative solution of linear equations with huge matrices. For example, for a 3-D model for a rectangular region that is $50 \times 50 \times 10 = 25,000$ nodes, the system has $100,000$ independent variables (number of nodes $\times$ 4 variables, 3 velocity variables and 1 pressure variable) and the matrix of the system could have

$100,000 \times 100,000 = 10^{10}$ elements. The challenge of solving the large size systems is discussed in Chapter 5.

3.2 Ice-stream Model

For modeling ice streams, the common approach is to treat them as barely-grounded ice shelves. In the ice shelf, the only stresses considered are the longitudinal and lateral stresses. The basal drag is not included in the fundamental formulation because the ice is supported by water. Figure (3.1) on Page 27 shows the forces acting on our differential volume. The resulting equations are called Morland equations [49].

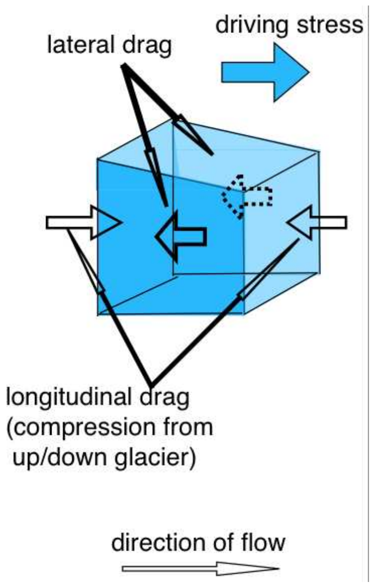

Figure 3.1. Ice-shelf: major forces acting on ice-shelf in one direction.

Ice stream flow is transitional between sheet flow and shelf flow. Hence for stream flow there will be both basal shear stresses and longitudinal stresses. A common approach to model ice stream flow is to add a small correction (proportional to speed) to Morland equations to simulate the basal drag effects after deriving the equations with the assumption of no basal drag. This addition of this friction term does violate assumptions of the Morland derivation.

This chapter solves this problem by including shear friction in the derivation of Morland equations. Otherwise, the derivation follows [46].

3.2.1 Diagnostic Equation

3.2.1.1 Ice-stream/ice-shelf Basal Boundary Conditions

We will start the derivation by re-writing the boundary conditions. The surface boundary conditions are the same for all type of flow, stress-free, and stays as (2.24a)-(2.24c).

To re-write the basal boundary condition, we will follow [49]. Figure 3.2 from [49] shows a horizontal plan of ice-flow as well as an element of the outer margin of the ice. $\vec{OX}$ and $\vec{OY}$ are the rectangular coordinates in which the equations are derived. The ice outer margin is shown by the bold line. Contour C, shown as the dashed boundary boundary, and the rear edge LOM are the boundary of the smooth steady flow. Panel *b* on Figure 3.2 shows an element of ice between C and the outer margin. It also shows local normal and tangential coordinates on C, (n, s). To re-write basal boundary conditions, we need local normal coordinates on the base of the ice $\vec{n_b}$ not shown on the figure.

Let's express zero tangential traction on base in the alternative form of components perpendicular to $\vec{n_b}$ and $\vec{OY}$ and $\vec{n_b}$ and $\vec{OX}$, where $\vec{OX}$ and $\vec{OY}$ are the rectangular coordinates in which the equations are derived. Let's denote the components of the outward-pointing unit normal vector to the bottom surface as $\vec{n_b} = (n_1, n_2, n_3)$. From (2.27), we get

102 L W MORLAND

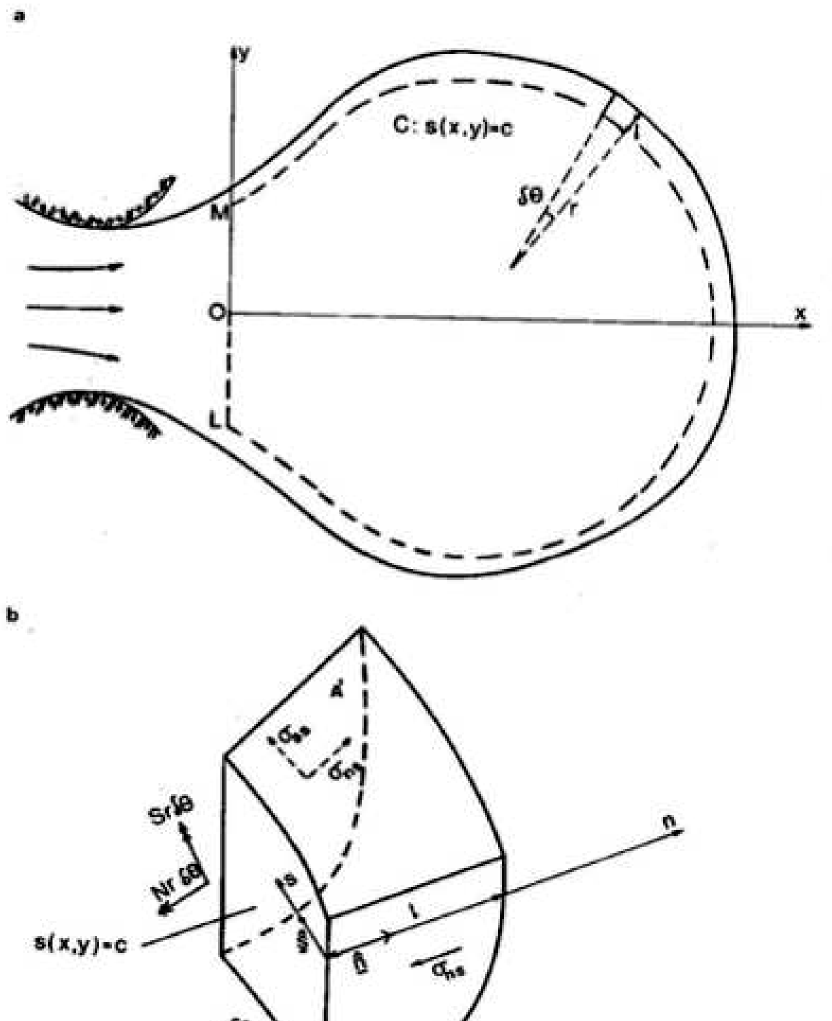

Figure 1. Unconfined ice-shelf flow. Panel a shows the horizontal plan, panel b the front region.

Figure 3.2. The horizontal plan and the front region of ice flow. From Morland(1987).

expressions for components of $\vec{n_b}$:

$$\begin{aligned} n_1 &= \frac{\frac{\partial z_b}{\partial x}}{\sqrt{1 + (\frac{\partial z_b}{\partial x})^2 + (\frac{\partial z_b}{\partial y})^2}}; \\ n_2 &= \frac{\frac{\partial z_b}{\partial y}}{\sqrt{1 + (\frac{\partial z_b}{\partial x})^2 + (\frac{\partial z_b}{\partial y})^2}}; \\ n_3 &= \frac{-1}{\sqrt{1 + (\frac{\partial z_b}{\partial x})^2 + (\frac{\partial z_b}{\partial y})^2}}. \end{aligned} \tag{3.6}$$

With $\vec{n_{bx}}$ the vector normal to $\vec{n_b}$ and $\vec{OY}$ and $\vec{n_{by}}$ the vector normal to $\vec{n_b}$ and $\vec{OX}$, we can define them as follows:

$$\vec{n_{bx}} = (\vec{OY} \times \vec{n_b}) = \begin{vmatrix} i & j & k \\ 0 & 1 & 0 \\ n_1 & n_2 & n_3 \end{vmatrix} = (n_3, 0, -n_1), \tag{3.7a}$$

$$\vec{n_{by}} = (\vec{n_b} \times \vec{OX}) = \begin{vmatrix} i & j & k \\ n_1 & n_2 & n_3 \\ 1 & 0 & 0 \end{vmatrix} = (0, n_3, -n_2) \tag{3.7b}$$

Then, the tangential traction component perpendicular to $\vec{n_b}$ and $\vec{OY}$ ($\vec{\tau_{bx}}$) is found as follows:

$$\tau_{bx} = \vec{n_b}[\vec{T}]\vec{n_{bx}\prime} = (n_1, n_2, n_3) \begin{vmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{vmatrix} \begin{pmatrix} n_3 \\ 0 \\ -n_1 \end{pmatrix} \tag{3.8}$$

$$= \Big(n_1\sigma_{xx} + n_2\sigma_{yx} + n_3\sigma_{zx},\ n_1\sigma_{xy} + n_2\sigma_{yy} + n_3\sigma_{zy},\ n_1\sigma_{xz} + n_2\sigma_{yz} + n_3\sigma_{zz}\Big) \begin{pmatrix} n_3 \\ 0 \\ -n_1 \end{pmatrix}$$

$$= n_1 n_3(\sigma_{xx} - \sigma_{zz}) + n_2 n_3 \sigma_{xy} + (n_3^2 - n_1^2)\sigma_{xz} - n_1 n_2 \sigma_{yz}.$$

Similarly, the tangential traction component perpendicular to $\vec{n_b}$ and $\vec{OX}$ ($\vec{\tau_{by}}$) is found as follows:

$$\tau_{by} = \vec{n_b}[\vec{T}]\vec{n_{by}\prime} = (n_1, n_2, n_3) \begin{vmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{vmatrix} \begin{pmatrix} 0 \\ n_3 \\ -n_2 \end{pmatrix} \tag{3.9}$$

$$= \Big(n_1\sigma_{xx} + n_2\sigma_{yx} + n_3\sigma_{zx},\ n_1\sigma_{xy} + n_2\sigma_{yy} + n_3\sigma_{zy},\ n_1\sigma_{xz} + n_2\sigma_{yz} + n_3\sigma_{zz}\Big) \begin{pmatrix} 0 \\ n_3 \\ -n_2 \end{pmatrix}$$

$$= n_2 n_3(\sigma_{yy} - \sigma_{zz}) + n_1 n_3 \sigma_{xy} + (n_3^2 - n_2^2)\sigma_{yz} - n_1 n_2 \sigma_{xz}.$$

After substituting the values of n_1, n_2, and n_3 from (3.6) in the above expression, we get the following expressions for τ_{bx} and τ_{by}:

$$\tau_{bx} = \frac{-\frac{\partial z_b}{\partial x}(\sigma_{xx} - \sigma_{zz}) - \frac{\partial z_b}{\partial y}\sigma_{xy} + \left(1 - \left(\frac{\partial z_b}{\partial x}\right)^2\right)\sigma_{xz} - \frac{\partial z_b}{\partial x}\frac{\partial z_b}{\partial y}\sigma_{yz}}{1 + (\frac{\partial z_b}{\partial x})^2 + (\frac{\partial z_b}{\partial y})^2} \tag{3.10a}$$

$$\tau_{by} = \frac{-\frac{\partial z_b}{\partial y}(\sigma_{yy} - \sigma_{zz}) - \frac{\partial z_b}{\partial x}\sigma_{xy} + \left(1 - \left(\frac{\partial z_b}{\partial y}\right)^2\right)\sigma_{yz} - \frac{\partial z_b}{\partial x}\frac{\partial z_b}{\partial y}\sigma_{xz}}{1 + (\frac{\partial z_b}{\partial x})^2 + (\frac{\partial z_b}{\partial y})^2} \tag{3.10b}$$

Define the tangential traction boundary conditions in directions n_{bx} and n_{by} as follows:

$$\tau_{bx} = f_{bx}, \quad \tau_{by} = f_{by} \tag{3.11}$$

Until now, we have been been following [49]. In MacAyeal-Morland model, the horizontal components of shear stresses at the bed in the Cartesian coordinates (x, y, z) are assumed to be zero:

$$\vec{f} = (f_x, f_y, f_z) = (0, 0, \rho g h)\,. \tag{3.12}$$

We will be using two different heuristic assumptions, that is, the horizontal shear stresses at the bed in Cartesian coordinates (x, y, z) are proportional in:

Modified Model 1: to unit velocity vectors:

$$\vec{f} = (f_x, f_y, f_z) = \left(-\tau_x \frac{u_x}{|\vec{u}|}, -\tau_y \frac{u_y}{|\vec{u}|}, \rho g h\right), \tag{3.13}$$

Modified Model 2: to driving stresses:

$$\vec{f} = (f_x, f_y, f_z) = \left(\alpha_x \rho g h \frac{\partial z_s}{\partial x}, \alpha_y \rho g h \frac{\partial z_s}{\partial y}, \rho g h\right), \tag{3.14}$$

where $\tau_x, \tau_y, \alpha_x(T)$, and $\alpha_y(T)$ are parameters ($\alpha_x(T) = \alpha_y(T) = 1$ if ice is grounded and $\alpha_x(T) = \alpha_y(T) = 0$ if ice is floating).

We denote the shear stresses at the bed in a new coordinate system (n_{bx}, n_{by}, n_b) as:

$$f = (f_{bx}, f_{by}, f_{bz}).$$

The shear stress components in new coordinate system can be found using the formula for the scalar product of two vectors:

$$\vec{f_{bx}} = \frac{(\vec{f} \cdot \vec{n_{bx}})}{|\vec{n_{bx}}|} = \frac{f_x n_3 + f_y 0 + f_z(-n_1)}{\sqrt{n_3^2 + 0^2 + (-n_1)^2}} = -\left[f_x + f_z \frac{\partial z_b}{\partial x}\right] \left\{1 + \left(\frac{\partial z_b}{\partial x}\right)^2\right\}^{-\frac{1}{2}}, \tag{3.15a}$$

$$\vec{f_{by}} = \frac{(\vec{f} \cdot \vec{n_{by}})}{|\vec{n_{by}}|} = \frac{f_x 0 + f_y n_3 + f_z(-n_2)}{\sqrt{0^2 + n_3^2 + (-n_2)^2}} = -\left[f_y + f_z \frac{\partial z_b}{\partial y}\right] \left\{1 + \left(\frac{\partial z_b}{\partial y}\right)^2\right\}^{-\frac{1}{2}}$$

Substituting (3.13) and (3.14) into (3.15a) gives the following shear stresses at the bed in the new coordinate system:

Modified Model 1:

$$\vec{f_{bx}} = -\frac{\rho g h \frac{\partial z_b}{\partial x} - \tau_x \frac{u_x}{|\vec{u}|}}{\sqrt{1 + \left(\frac{\partial z_b}{\partial x}\right)^2}}, \; \vec{f_{by}} = -\frac{\rho g h \frac{\partial z_b}{\partial y} - \tau_y \frac{u_y}{|\vec{u}|}}{\sqrt{1 + \left(\frac{\partial z_b}{\partial y}\right)^2}} \tag{3.16}$$

Modified Model 2:

$$\vec{f_{bx}} = -\rho g h \frac{\alpha_x \frac{\partial z_s}{\partial x} + \frac{\partial z_b}{\partial x}}{\sqrt{1 + \left(\frac{\partial z_b}{\partial x}\right)^2}}, \; \vec{f_{by}} = -\rho g h \frac{\alpha_y \frac{\partial z_s}{\partial y} + \frac{\partial z_b}{\partial y}}{\sqrt{1 + \left(\frac{\partial z_b}{\partial y}\right)^2}} \tag{3.17}$$

The boundary condition (3.11) can then be written as follows:

$$
\begin{aligned}
-\frac{\partial z_b}{\partial x}(\sigma_{xx}-\sigma_{zz})-\frac{\partial z_b}{\partial y}\sigma_{yx}+\left(1-\left(\frac{\partial z_b}{\partial x}\right)^2\right)\sigma_{zx}-\frac{\partial z_b}{\partial x}\frac{\partial z_b}{\partial y}\sigma_{yz}&=r_b^2\vec{f_{bx}},\\
-\frac{\partial z_b}{\partial y}(\sigma_{yy}-\sigma_{zz})-\frac{\partial z_b}{\partial x}\sigma_{xy}+\left(1-\left(\frac{\partial z_b}{\partial y}\right)^2\right)\sigma_{yz}-\frac{\partial z_b}{\partial x}\frac{\partial z_b}{\partial y}\sigma_{xz}&=r_b^2\vec{f_{by}},
\end{aligned}
$$

where $r_b^2 = 1 + (\frac{\partial z_b}{\partial x})^2 + (\frac{\partial z_b}{\partial y})^2$.

Finally substituting strain rates for stresses σ_{ij} using (2.2) with the linearized flow law of equation (2.9), the above basal boundary conditions can be re-written as follows:

$$
\begin{aligned}
-\frac{\partial z_b}{\partial x}(2\mu\dot{\epsilon}_{xx}+P)-\frac{\partial z_b}{\partial y}2\mu\dot{\epsilon}_{yx}+2\mu\dot{\epsilon}_{xz}+\frac{\partial z_b}{\partial x}(2\mu\dot{\epsilon}_{zz}+P)&\\
-\left(\frac{\partial z_b}{\partial x}\right)^2 2\mu\dot{\epsilon}_{xz}-\frac{\partial z_b}{\partial x}\frac{\partial z_b}{\partial y}2\mu\dot{\epsilon}_{yz}&=r_b^2\vec{f_{bx}}, \qquad (3.19a)\\
-\frac{\partial z_b}{\partial y}(2\mu\dot{\epsilon}_{yy}+P)-\frac{\partial z_b}{\partial x}2\mu\dot{\epsilon}_{xy}+2\mu\dot{\epsilon}_{yz}+\frac{\partial z_b}{\partial y}(2\mu\dot{\epsilon}_{zz}+P)&\\
-\left(\frac{\partial z_b}{\partial y}\right)^2 2\mu\dot{\epsilon}_{yz}-\frac{\partial z_b}{\partial x}\frac{\partial z_b}{\partial y}2\mu\dot{\epsilon}_{xz}&=r_b^2\vec{f_{by}}. \qquad (3.19b)
\end{aligned}
$$

3.2.1.2 Vertical Integration of Momentum Equations

In ice-shelf and ice-stream modeling, it is assumed that horizontal velocities and strain rates are independent of z, or,

$$
\frac{\partial \dot{\epsilon_{xx}}}{\partial z}\to 0,\;\; \frac{\partial \dot{\epsilon_{yy}}}{\partial z}\to 0,\;\; \frac{\partial \dot{\epsilon_{xy}}}{\partial z}\to 0,\;\; \frac{\partial u_x}{\partial z}\to 0,\;\; \frac{\partial u_y}{\partial z}\to 0,\;\; \dot{\epsilon_{zx}}=\dot{\epsilon_{zy}}=0. \qquad (3.20)
$$

These assumptions are justified by the flat and thin geometry of ice shelves.

Using these assumptions, the Stokes' equations (2.19a)-(2.19c) can be reduced and written in terms of a horizontal, depth-averaged ice velocity $\vec{U} = (U_x, U_y)$. To derive the reduced equations, Stokes' equations (2.19a)-(2.19c) are integrated over depth.

3.2.1.3 Integration of the z-momentum equation

Integrating the vertical component of the Stokes' equation (2.19c) from some level z to $z = z_s$, rearranging terms using Leibnitz Rule[1], and using boundary condition at the surface (2.24c) generates an equation for the pressure field:

$$P(z) = +\frac{\partial}{\partial x}\int_z^{z_s} 2\mu\dot{\epsilon}_{xz}\,dz + \frac{\partial}{\partial y}\int_z^{z_s} 2\mu\dot{\epsilon}_{yz}\,dz - 2\mu\dot{\epsilon}_{zz} - \rho g(z_s - z)$$

Using assumption (3.20) that $\dot{\epsilon}_{xz}$ and $\dot{\epsilon}_{yz}$ are independent of z, the above equation is simplified as follows:

$$P(z) = -2\mu\dot{\epsilon}_{zz} - \rho g(z - z_s) + \frac{\partial}{\partial x}\left[2\bar{\mu}\dot{\epsilon}_{xz}(z_s - z)\right] + \frac{\partial}{\partial y}\left[2\bar{\mu}\dot{\epsilon}_{yz}(z_s - z)\right] \tag{3.21}$$

where $\bar{\mu}$ is the depth-averaged effective viscosity (overbar denotes depth averaging): $\bar{\mu} = \frac{1}{z_s - z}\int_z^{z_s}\mu\,dz$.

[1]
$$\int_{z_b}^{z_s}\frac{\partial f(x,z,\dots)}{\partial x}dz = \frac{\partial}{\partial x}\int_{z_b}^{z_s} f(x,z,\dots) - f(x,z_s,\dots)\frac{\partial z_s}{\partial x} + f(x,z_b,\dots)\frac{\partial z_b}{\partial x}$$

3.2.1.4 Integration of the x-momentum equation

Integrating the $x-$ horizontal component of the Stokes' equation (2.19a) from $z = z_b$ to $z = z_s$ and again rearranging terms using Leibnitz Rule gives the following system:

$$\begin{aligned}
&\frac{\partial}{\partial x}\int_{z_b}^{z_s} P\,dz + \frac{\partial}{\partial x}\int_{z_b}^{z_s} 2\mu\dot{\epsilon}_{xx}\,dz + \frac{\partial}{\partial y}\int_{z_b}^{z_s} 2\mu\dot{\epsilon}_{xy}\,dz \\
&\quad - \left[(2\mu\dot{\epsilon}_{xx} + P)\frac{\partial z_s}{\partial x} + 2\mu\dot{\epsilon}_{xy}\frac{\partial z_s}{\partial y} - 2\mu\dot{\epsilon}_{xz}\right]_{z=z_s} \\
&+ \left[(2\mu\dot{\epsilon}_{xx} + P)\frac{\partial z_b}{\partial x} + 2\mu\dot{\epsilon}_{xy}\frac{\partial z_b}{\partial y} - 2\mu\dot{\epsilon}_{xz}\right]_{z=z_b} = 0.
\end{aligned}$$

Applying boundary conditions (2.24a) and (3.19a) gives the following system:

$$\begin{aligned}
&\frac{\partial}{\partial x}\int_{z_b}^{z_s} P\,dz + \frac{\partial}{\partial x}\int_{z_b}^{z_s} 2\mu\dot{\epsilon}_{xx}\,dz + \frac{\partial}{\partial y}\int_{z_b}^{z_s} 2\mu\dot{\epsilon}_{xy}\,dz + \frac{\partial z_b}{\partial x}\,(2\mu\dot{\epsilon}_{zz} + P) \\
&-(\frac{\partial z_b}{\partial x})^2 2\mu\dot{\epsilon}_{xz} - \frac{\partial z_b}{\partial x}\frac{\partial z_b}{\partial y}2\mu\dot{\epsilon}_{yz} - r_b^2\vec{f_{bx}} \;=\; 0.
\end{aligned}$$

To estimate the first term of the above equation, let's integrate pressure over depth (we also use the incompressibility condition, $\dot{\epsilon}_{zz} = -(\dot{\epsilon}_{xx} + \dot{\epsilon}_{yy})$):

$$\begin{aligned}
\frac{\partial}{\partial x}\int_{z_b}^{z_s} P\,dz \;=\; &\frac{\partial}{\partial x}\int_{z_b}^{z_s} 2\mu\dot{\epsilon}_{xx}\,dz + \frac{\partial}{\partial x}\int_{z_b}^{z_s} 2\mu\dot{\epsilon}_{yy}\,dz - \frac{\partial}{\partial x}\frac{\rho g h^2}{2} \\
&+\frac{\partial}{\partial x}\int_{z_b}^{z_s}\frac{\partial}{\partial x}\left[2\bar{\mu}\dot{\epsilon}_{xz}(z_s - z)\right]\,dz + \frac{\partial}{\partial x}\int_{z_b}^{z_s}\frac{\partial}{\partial y}\left[2\bar{\mu}\dot{\epsilon}_{yz}(z_s - z)\right]\,dz
\end{aligned}$$

With this assumption, the equation for x-momentum becomes:

$$\begin{aligned}\frac{\partial}{\partial x}\int_{z_b}^{z_s} 2\mu\dot{\epsilon}_{xx}\,dz + \frac{\partial}{\partial x}\int_{z_b}^{z_s} 2\mu\dot{\epsilon}_{yy}\,dz - \frac{\partial}{\partial x}\frac{\rho g h^2}{2} + \frac{\partial}{\partial x}\int_{z_b}^{z_s} 2\mu\dot{\epsilon}_{xx}\,dz + \frac{\partial}{\partial y}\int_{z_b}^{z_s} 2\mu\dot{\epsilon}_{xy}\,dz \\ + \frac{\partial}{\partial x}\int_{z_b}^{z_s} \frac{\partial}{\partial x}\left(2\bar{\mu}\dot{\epsilon}_{xz}(z_s - z)\right)\,dz + \frac{\partial}{\partial x}\int_{z_b}^{z_s} \frac{\partial}{\partial y}\left(2\bar{\mu}\dot{\epsilon}_{yz}(z_s - z)\right)\,dz \\ + \frac{\partial z_b}{\partial x}\left(2\mu\dot{\epsilon}_{zz} + P\right) - (\frac{\partial z_b}{\partial x})^2 2\mu\dot{\epsilon}_{xz} - \frac{\partial z_b}{\partial x}\frac{\partial z_b}{\partial y}2\mu\dot{\epsilon}_{yz} - r_b^2\vec{f_{bx}} = 0.\end{aligned}$$

Using assumption (3.20) that $\dot{\epsilon}_{xx}$, $\dot{\epsilon}_{yy}$, and $\dot{\epsilon}_{yz}$ are independent of z and $\dot{\epsilon}_{xz} = 0$ and $\dot{\epsilon}_{yz} = 0$, the above equation is simplified as:

$$\begin{aligned}&\frac{\partial}{\partial x}\left[2\bar{\mu}h(2\dot{\epsilon}_{xx} + \dot{\epsilon}_{yy})\right] + \frac{\partial}{\partial y}(2\bar{\mu}h\dot{\epsilon}_{xy}) + \frac{\partial z_b}{\partial x}\left[2\mu\dot{\epsilon}_{zz} + P\right] = \\ &\rho g h\frac{\partial h}{\partial x} + r_b^2\vec{f_{bx}}.\end{aligned}$$

If we substitute the fact that at the bed $P = -2\mu\dot{\epsilon}_{zz} - \rho g h$ (3.21) and use formula $h + z_b = z_s$, then the above equation can be written as follows:

$$\frac{\partial}{\partial x}\left[2\bar{\mu}h(2\dot{\epsilon}_{xx} + \dot{\epsilon}_{yy})\right] + \frac{\partial}{\partial y}(2\bar{\mu}h\dot{\epsilon}_{xy}) = \rho g h\frac{\partial z_s}{\partial x} + r_b^2\vec{f_{bx}}. \tag{3.22}$$

Equation (3.22) is the x-component of the reduced stress-equation for ice streams.

To summarize and to ease the comparison with the original MacAyeal-Morland equation, we can re-write the above x-component of the reduced balance of momentum equation:

$$\frac{\partial}{\partial x}\left[2\bar{\mu}h(2\dot{\epsilon}_{xx} + \dot{\epsilon}_{yy})\right] + \frac{\partial}{\partial y}(2\bar{\mu}h\dot{\epsilon}_{xy}) = RHS, \tag{3.23}$$

where the right-hand side, RHS, is as follows in:

Morland Model for ice-shelves:	$\rho g h \frac{\partial z_s}{\partial x}$,
MacAyeal Model for ice-streams:	$\rho g h \frac{\partial z_s}{\partial x} + \tau_c \frac{u_x}{\lvert\vec{u}\rvert}$,
Modified Model 1:	$\rho g h \frac{\partial z_s}{\partial x} + \left(\tau_x \frac{u_x}{\lvert\vec{u}\rvert} - \rho g h \frac{\partial z_b}{\partial x}\right) g_{bx}$,
Modified Model 2:	$\rho g h \frac{\partial z_s}{\partial x} - \rho g h \left(\alpha_x \frac{\partial z_s}{\partial x} + \frac{\partial z_b}{\partial x}\right) g_{bx}$,

where $g_{bx} = \left(1 + (\frac{\partial z_b}{\partial x})^2 + (\frac{\partial z_b}{\partial y})^2\right) \left(1 + (\frac{\partial z_b}{\partial x})^2\right)^{-\frac{1}{2}}$

The equations show that the modified models include the bed slopes. In the case when the bed surface is flat, $\frac{\partial z_b}{\partial x} = 0, \ \frac{\partial z_b}{\partial y} = 0$, the right-hand side of the equation becomes as follows:

Morland Model for ice-shelves:	$\rho g h \frac{\partial z_s}{\partial x}$,
MacAyeal Model for ice-streams:	$\rho g h \frac{\partial z_s}{\partial x} + \tau_c \frac{u_x}{\lvert\vec{u}\rvert}$,
Modified Model 1:	$\rho g h \frac{\partial z_s}{\partial x} + \tau_x \frac{u_x}{\lvert\vec{u}\rvert}$,
Modified Model 2:	$\rho g h (1 - \alpha_x) \frac{\partial z_s}{\partial x}$.

We can see that for the flat-bed ice-streams, derived equations of the Modified Model 1 are identical to the ones of MacAyeal model. In the case of Modified Model 2, the shear stresses at the bed of ice-streams are adjusted not by subtracting from ice-sheet shear stresses the forces proportional to velocities but multiplying the ice-sheet shear stresses by factor $(1 - \alpha_x)$.

Below are two extreme cases of this equation:

- **for ice-shelf ($\alpha_x = 0$):** equation (3.22) becomes exactly equation (3.26) in [46]:

$$\frac{\partial}{\partial x}\left[2\bar{\mu} h (2\dot{\epsilon}_{xx} + \dot{\epsilon}_{yy})\right] + \frac{\partial}{\partial y}(2\bar{\mu} h \dot{\epsilon}_{xy}) = \rho g h \frac{\partial z_s}{\partial x}$$

- **for ice-sheet ($\alpha_x = 1$):** equation (3.22)becomes:

$$\frac{\partial}{\partial x}\left[2\bar{\mu} h (2\dot{\epsilon}_{xx} + \dot{\epsilon}_{yy})\right] + \frac{\partial}{\partial y}(2\bar{\mu} h \dot{\epsilon}_{xy}) = 0.$$

Finally, the $x-$ component of the diagnostic equation (3.22) and the similar equation for $y-$ component of the diagnostic equation can be written in terms of depth-averaged horizontal velocities using the definition of strain-rate components (2.4) as follows:

$$\frac{\partial}{\partial x}\left(2\overline{\mu}h\left(2\frac{\partial U_x}{\partial x}+\frac{\partial U_y}{\partial y}\right)\right)+\frac{\partial}{\partial y}\left(\overline{\mu}h\left(\frac{\partial U_x}{\partial y}+\frac{\partial U_y}{\partial x}\right)\right)=\rho g h\frac{\partial z_s}{\partial x}+r_b^2\vec{f_{bx}}, \tag{3.24a}$$

$$\frac{\partial}{\partial y}\left(2\overline{\mu}h\left(2\frac{\partial U_y}{\partial y}+\frac{\partial U_x}{\partial x}\right)\right)+\frac{\partial}{\partial x}\left(\overline{\mu}h\left(\frac{\partial U_x}{\partial y}+\frac{\partial U_y}{\partial x}\right)\right)=\rho g h\frac{\partial z_s}{\partial y}+r_b^2\vec{f_{by}}. \tag{3.24b}$$

3.2.1.5 Boundary Conditions along the Edge of the Domain

Two types of boundary conditions can be specified along the edge ($\partial\Omega$) of the domain (Ω), Dirichlet and Neumann. Dirichlet boundary conditions, specification of the depth-averaged velocity, are applied at zero slip coast-lines or where ice streams flow into the ice shelf or at stagnant ice-shelf boundaries. Neumann boundary conditions are specified at the seaward, iceberg-caving front. The depth-integrated balance of forces at the ice front is formulated as a balance of the depth-integrated force transmitted across the ice front due to internal stresses and the integral of the hydrostatic pressure in the seawater beyond the ice front over the face of the ice front:

$$\int_{z_b}^{z_s}\sigma_{ij}n_j dz=-\frac{\rho_w g}{2}\left(\frac{\rho}{\rho_w}h\right)^2\vec{n} \tag{3.25}$$

where ρ_w is the average density of seawater. In the above equations, it is assumed that the ice shelf floats in hydrostatic equilibrium with seawater, $z_b=-\frac{\rho}{\rho_w}h$.

In the cases when the ice front extends along the $y-$ and $x-$ axes, that is, when $\vec{n} = n_x$ and $\vec{n} = n_y$ consequently, boundary condition (3.25) becomes:

$$\vec{n} = n_x : \qquad \int_{z_b}^{z_s} \left(2\mu \dot{\epsilon_{xx}} + P\right) dz = -\frac{\rho_w g}{2}\left(\frac{\rho}{\rho_w} h\right)^2 ; \tag{3.26a}$$

$$\vec{n} = n_y : \qquad \int_{z_b}^{z_s} \left(2\mu \dot{\epsilon_{yy}} + P\right) dz = -\frac{\rho_w g}{2}\left(\frac{\rho}{\rho_w} h\right)^2 . \tag{3.26b}$$

Finally, using (3.21) and incompressibility condition $\dot{\epsilon_{xx}} + \dot{\epsilon_{xx}} + \dot{\epsilon_{xx}} = 0$, they can be written as follows:

$$\vec{n} = n_x : \qquad 2\overline{\mu} h \left(2\frac{\partial U_x}{\partial x} + \frac{\partial U_y}{\partial y}\right) = \frac{\rho g h^2}{2}\left(1 - \frac{\rho}{\rho_w}\right); \tag{3.27a}$$

$$\vec{n} = n_y : \qquad 2\overline{\mu} h \left(\frac{\partial U_x}{\partial x} + 2\frac{\partial U_y}{\partial y}\right) = \frac{\rho g h^2}{2}\left(1 - \frac{\rho}{\rho_w}\right). \tag{3.27b}$$

3.2.2 Prognostic Equation

The above momentum equations yield the z-independent horizontal velocity field $\vec{\mathbf{U}} = (U_x(x,y), U_y(x,y))$ that corresponds to an instantaneous *snap shot* of the ice-thickness field, $h(x,y,t)$. The time-evolution of the ice shelf is governed by the prognostic equation:

$$\frac{\partial h}{\partial t} = -\frac{\partial\left(h \cdot U_x\right)}{\partial x} - \frac{\partial\left(h \cdot U_y\right)}{\partial y} + \dot{a}. \tag{3.28}$$

It is important to notice that the ice-shelf flux term in (3.28) is a *non-local* function of ice-thickness. It depends on the flux of ice from the surrounding locations. This is in stark contrast to the grounded ice sheet SIA model.

3.2.3 Complexity of Ice-shelf Equations

In the ice-shelf model the basal stress is neglected. This allows us to reduce the 3-dimensional equations to quasi-2-dimensional equations (x and y) with variables in z directions integrated out. That is, to solve Morland equations, we have to solve the system of three equations with three degrees of freedom, equations (3.28) and (3.24a) - (3.24b) for three (3) variables, h, U_x, and U_y.

Equations (3.24a) - (3.24b) are nonlinear two-dimensional equations. The linearization of the flow law (2.9) for the numerical solution of the system allows us to solve the linear problem, and then iterate on the effective viscosity, which itself depends on the velocity field.

For a rectangular region that is $50 \times 40 = 2,000$ nodes, the system has $4,000$ independent variables (number of nodes $\times$ 2 velocity variables) and the matrix of the system has $4,000 \times 4,000 = 1.6 \times 10^6$ elements which is 100 times smaller than the corresponding matrix for a 3-dimensional model.

Chapter 4

NUMERICAL SOLUTION - FINITE ELEMENT METHOD

The models constructed for full 3-dimensional system and for 2-dimensional ice-shelf equations share several key points. First, the time-stepping procedure is split into a two-step algorithm. One step involves solution of the diagnostic equation which determines the velocity field from the ice-thickness field and boundary conditions. Since the diagnostic equations are non-linear, they are solved iteratively. The second step involves solution of the prognostic equation which updates the ice-thickness distribution using the new velocity field and boundary conditions.

Both the diagnostic and prognostic equations are discretized using the finite element method (FEM). The finite-element method is a standard numerical technique which can be successfully applied to any of the conservation equations described in Chapter 3, either in a steady-state or a time-dependent situation. The domain on which the conservation equation is to be solved can be complexly irregular, with no need for the curvilinear or normalized coordinates often required by the finite-difference method. Boundary conditions can be easily specified along this irregular boundary as a mixture of essential boundary conditions (specified state variable) or natural boundary conditions (specified flux or specified linear combination of flux and state variable).

Equations such as (3.1a)-(3.1c) or (3.3) are called the “strong” or classical formulation of the problem. In this formulation, a solution is required to satisfy the differential equations at any point in the domain as well as boundary conditions.

However, if data have any irregularities, or are not smooth enough, or the domain of the problem is very complex, then the strong solution may not exist or cannot be found. In this case, we can look for a *weak* or *variational* solution that may not satisfy the equation and boundary conditions at every point of the domain but may satisfy them in average.

The variational formulation of the problem can be obtained by multiplying the differential equation by an arbitrary test or weight function and integrating over the whole domain. The choice and requirements of test function is discussed, for example, in [3]. To solve the variational problem, the Galerkin method is used. It approximates the solution as a finite linear combination of basis functions. Criteria in choosing basis functions include the following considerations:

1. is there a systematic way of constructing the basis functions? The construction of basis functions become a complex problem in two- and three- dimensional boundary problems where functions must be designed to fit the boundary conditions on domains with complex geometries.
2. is the resulting matrix of the generated system of linear equations, called a *stiffness* matrix a sparse matrix (which would greatly reduce the memory and time required to solve the system)? and
3. is the calculation of the stiffness matrix elements simple?

These difficulties are resolved in *Finite Element Method* by

- dividing the domain of the problem into finite number of subregions called *finite elements*; and
- approximating both the unknown solution and the arbitrary introduced test function as a finite sum of some predefined basis functions which are constructed as a piecewise functions (polynomials of low degree) that are different from zero only on a few adjacent subregions.

These approximations reduce the variational problem to a system of linear equations where unknowns are the values of the unknown solution at the nodes of elements of the finite element mesh. A major advantage of FEM over spectral or Taylor series solutions is the sparseness of the resulting matrix due to the fact that the basis functions are different from zero only on a few adjacent subregions of the domain. More on FEM can be found, for example, in [3].

4.1 FEM formulation of Higher-Order 3-D Model

Paragraphs below show the variational forms of momentum and mass conservation equations, as well as the final systems of linear equations generated by FEM for 3-dimensional full momentum equations.

4.1.1 Approximation of Conservation of Momentum Equations

4.1.1.1 Variational Form

The variational form of the x-component of the momentum equations is constructed by multiplying the residual error function from equation (3.1a):

$$\frac{\partial\left(P+2\mu\frac{\partial u_x}{\partial x}\right)}{\partial x}+\frac{\partial\left(\mu(\frac{\partial u_x}{\partial y}+\frac{\partial u_y}{\partial x})\right)}{\partial y}+\frac{\partial\left(\mu(\frac{\partial u_x}{\partial z}+\frac{\partial u_z}{\partial x})\right)}{\partial z}+\rho f_x$$

by a test function $\psi(x,y,z)$ and integrating over the problem's domain Ω:

$$\begin{aligned}\iiint_\Omega \psi(x,y,z) \quad &\left\{\frac{\partial\left(P+2\mu\frac{\partial u_x}{\partial x}\right)}{\partial x}+\right.\\ &\left.\frac{\partial\left(\mu(\frac{\partial u_x}{\partial y}+\frac{\partial u_y}{\partial x})\right)}{\partial y}+\frac{\partial\left(\mu(\frac{\partial u_x}{\partial z}+\frac{\partial u_z}{\partial x})\right)}{\partial z}+\rho f_x\right\} dx\,dy\,dz=0.\end{aligned}$$

Use of the divergence theorem converts the above equation to the following form:

$$\begin{aligned}\iiint_\Omega \quad &\left\{-\rho f_x\psi+\left(P+2\mu\frac{\partial u_x}{\partial x}\right)\frac{\partial\psi}{\partial x}\right.\\ &\left.+\mu\left(\frac{\partial u_x}{\partial y}+\frac{\partial u_y}{\partial x}\right)\frac{\partial\psi}{\partial y}+\mu\left(\frac{\partial u_x}{\partial z}+\frac{\partial u_z}{\partial x}\right)\frac{\partial\psi}{\partial z}\right\} dx\,dy\,dz- \\ \iint_{\partial\Omega}\psi \quad &\left(\left(P+2\mu\frac{\partial u_x}{\partial x}\right)n_x+\mu\left(\frac{\partial u_x}{\partial y}+\frac{\partial u_y}{\partial x}\right)n_y+\mu\left(\frac{\partial u_x}{\partial z}+\frac{\partial u_z}{\partial x}\right)n_z\right)ds=0.\end{aligned} \tag{4.1}$$

4.1.1.2 Application of Boundary Conditions

Let's assume that the boundary $\partial\Omega$ consists of the following parts:

$$\partial\Omega = \partial\Omega_{surface} + \partial\Omega_{bedD} + \partial\Omega_{bedN} + \partial\Omega_{sideD} + \partial\Omega_{sideN},$$

In the expression above

$\partial\Omega_{surface}$ is a surface boundary of the domain where stress-free conditions (2.24a)-(2.24c) are specified;

$\partial\Omega_{bedD}$ is a basal boundary of the domain where Dirichlet boundary conditions (known velocities) are specified;

$\partial\Omega_{bedN}$ is a basal boundary of the domain where Newman boundary conditions (2.28a)-(2.28c) are specified;

$\partial\Omega_{sideD}$ is side boundary of the domain where Dirichlet boundary conditions are specified; and

$\partial\Omega_{sideN}$ is the remaining side boundary of the domain where Newman boundary conditions are specified.

Using ice surface and basal boundary conditions, the boundary integral in (4.1) can be re-written as follows:

$$\iint_{\partial\Omega} \psi R ds = \iint_{\partial\Omega_{surface}} \psi R ds + \iint_{\partial\Omega_{bedD}} \psi R ds + \iint_{\partial\Omega_{bedN}} \psi R ds + \iint_{\partial\Omega_{sideD}} \psi R ds + \iint_{\partial\Omega_{sideN}} \psi R ds \tag{4.2}$$

where

$$R = \left(P + 2\mu\frac{\partial u_x}{\partial x}\right)n_x + \mu\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right)n_y + \mu\left(\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x}\right)n_z.$$

In (4.2), integral $\iint_{\partial\Omega_{surface}} \psi R ds = 0$ because of stress-free conditions (2.24a)-(2.24c); integrals $\iint_{\partial\Omega_{bedD}} \psi R ds$ and $\iint_{\partial\Omega_{sideD}} \psi R ds$ are made equal to zero by choosing the arbitrary test functions ψ to be equal to zero on those boundaries; and $\iint_{\partial\Omega_{bedN}}\psi R ds = -\iint_{\partial\Omega_{bedN}}\psi(\alpha_x \rho g h \frac{\partial z_b}{\partial x})ds$ because of (2.28a). The last integral $\iint_{\partial\Omega_{sideN}} \psi R ds$ can be defined in a similar way. Assuming for simplicity that all lateral boundary conditions are Dirichlet boundary conditions, the boundary integral in (4.1) can be re-written as follows:

$$\iint_{\partial\Omega} \psi R ds = \iint_{\partial\Omega_{bedN}} \psi R ds = -\iint_{\partial\Omega_{bedN}} \psi\left(\alpha_x \rho g h \frac{\partial z_b}{\partial x}\right) ds \tag{4.3}$$

The variational forms of y- and z- components of the momentum equations are derived in a similar way.

4.1.1.3 Finite Element Algorithm

The above equations apply for the domain as a whole, but they also apply for a particular sub-domain, or element. Thus we would have a set of element equations corresponding to the above, where only Ω and $\delta\Omega$ would be changed to Ω^e and $\delta\Omega^e$ for element e. The important consequence of that fact is that the integrals in equations (4.1) can be calculated by adding contributions from integrals over each individual element.

Let's consider shape functions $\psi_j^e = \psi_j^e(x, y, z)$, which are the simple polynomial pieces of the piece-wise defined basis functions mentioned on page 44.. For an element with $N_e = 8$ nodes, the x-, y-, z- components of the velocity at any point within the element can be expressed as a sum of values at the nodes times shape functions evaluated at the point:

$$u_x^e = \sum_{j=1}^{N_e} u_{x_j}^{\ e} \psi_j^e, \quad u_y^e = \sum_{j=1}^{N_e} u_{y_j}^{\ e} \psi_j^e, \quad u_z^e = \sum_{j=1}^{N_e} u_{z_j}^{\ e} \psi_j^e, \tag{4.4}$$

and the test function at any point within the element can be expressed as:

$$\psi^e = \sum_{i=1,N_e} \Psi_i^e \psi_i^e, \tag{4.5}$$

where Ψ_i^e is the value of function ψ at node i in element e, $u_{x_i}^{\ e}$ is the value of u_x an node i in element e, etc.

After substituting (4.5) into (4.1) written for an element and requiring that it is satisfied for any Ψ_i^e, we get the following equations for $i = 1, ..., N_e$ (N_e - is the number of nodes) for the x-component of the momentum equation:

$$\begin{aligned}
\iiint_{\Omega_e} \quad & \left\{ -\rho f_x \psi_i^e + \left(P + 2\mu \frac{\partial u_x}{\partial x} \right) \frac{\partial \psi_i^e}{\partial x} \right. \\
& \left. + \mu (\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}) \frac{\partial \psi_i^e}{\partial y} + \mu (\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x}) \frac{\partial \psi_i^e}{\partial z} \right\} dx\, dy\, dz - \\
\iint_{\partial \Omega_e} \psi_i^e \; & \left(\left(P + 2\mu \frac{\partial u_x}{\partial x} \right) n_x + \mu \left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \right) n_y + \mu \left(\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x} \right) n_z \right) ds = 0.
\end{aligned}$$

The above equation can be rewritten as follows:

$$\begin{aligned}\iiint_{\Omega_e} \mu \; & \left\{2\frac{\partial u_x}{\partial x}\frac{\partial \psi_i^e}{\partial x} + \left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right)\frac{\partial \psi_i^e}{\partial y} + \left(\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x}\right)\frac{\partial \psi_i^e}{\partial z}\right\} dx\, dy\, dz = \\ & \iiint_{\Omega_e}\left(\rho f_x \psi_i^e - P\frac{\partial \psi_i^e}{\partial x}\right) dx\, dy\, dz + \iint_{\partial\Omega_e} \psi_i^e P n_x\, ds + \\ & \iint_{\partial\Omega_e} \mu\psi_i^e \left\{2\frac{\partial u_x}{\partial x}n_x + \left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right)n_y + \left(\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x}\right)n_z\right\} ds.\end{aligned}$$

Then, after substituting u_x^e, u_y^e, u_z^e into the above formula, we get the following equations for the x-component of the momentum equation:

$$\begin{aligned}\textstyle\sum_{j=1}^{N_e}\iiint_{\Omega_e} \quad & \mu\left\{2u_{xj}^{e}\frac{\partial \psi_j^e}{\partial x}\frac{\partial \psi_i^e}{\partial x} + \left(u_{xj}^{e}\frac{\partial \psi_j^e}{\partial y} + u_{yj}^{e}\frac{\partial \psi_j^e}{\partial x}\right)\frac{\partial \psi_i^e}{\partial y}\right. \\ & \left. + \left(u_{xj}^{e}\frac{\partial \psi_j^e}{\partial z} + u_{zj}^{e}\frac{\partial \psi_j^e}{\partial x}\right)\frac{\partial \psi_i^e}{\partial z}\right\} dx\, dy\, dz = \\ & \iiint_{\Omega_e}\left(\rho f_x \psi_i^e - P\frac{\partial \psi_i^e}{\partial x}\right) dx\, dy\, dz + \iint_{\partial\Omega_e} \psi_i^e P n_x\, ds + \\ \textstyle\sum_{j=1}^{N_e}\iint_{\partial\Omega_e} \quad & \mu\psi_i^e\left\{2u_{xj}^{e}\frac{\partial \psi_j^e}{\partial x}n_x + \left(u_{xj}^{e}\frac{\partial \psi_j^e}{\partial y} + u_{yj}^{e}\frac{\partial \psi_j^e}{\partial x}\right)n_y\right. \\ & \left. + \left(u_{xj}^{e}\frac{\partial \psi_j^e}{\partial z} + u_{zj}^{e}\frac{\partial \psi_j^e}{\partial x}\right)n_z\right\} ds.\end{aligned}$$

Thus, for a eight-node quadrilateral element the unknowns consist of the 24 velocity components and the pressure, (u_{1x}, u_{1y}, u_{1z}, u_{2x}, u_{2y}, u_{2z}, ..., u_{8x}, u_{8y}, u_{8z}, P). And the momentum balance equation for x-component after some rearrangement is as follows:

$$
\begin{aligned}
&\sum_{j=1}^{N_e}\iiint_{\Omega_e}\mu\left\{u_{x_j}^{e}\left(2\frac{\partial\psi_j^e}{\partial x}\frac{\partial\psi_i^e}{\partial x}+\frac{\partial\psi_j^e}{\partial y}\frac{\partial\psi_i^e}{\partial y}+\frac{\partial\psi_j^e}{\partial z}\frac{\partial\psi_i^e}{\partial z}\right)+u_{y_j}^{e}\frac{\partial\psi_j^e}{\partial x}\frac{\partial\psi_i^e}{\partial y}+u_{z_j}^{e}\frac{\partial\psi_j^e}{\partial x}\frac{\partial\psi_i^e}{\partial z}\right\}dxdydz\\
&\quad=\iiint_{\Omega_e}\left(\rho f_x\psi_i^e-P\frac{\partial\psi_i^e}{\partial x}\right)dx\,dy\,dz+\iint_{\partial\Omega_e}\psi_i^e P n_x\,ds+ \qquad (4.6)\\
&\quad\sum_{j=1}^{N_e}\iint_{\partial\Omega_e}\mu\psi_i^e\left\{u_{x_j}^{e}\left(2\frac{\partial\psi_j^e}{\partial x}n_x+\frac{\partial\psi_j^e}{\partial y}n_y+\frac{\partial\psi_j^e}{\partial z}n_z\right)+u_{y_j}^{e}\frac{\partial\psi_j^e}{\partial x}n_y+u_{z_j}^{e}\frac{\partial\psi_j^e}{\partial x}n_z\right\}ds.
\end{aligned}
$$

Momentum equations for y- and z- component of the velocity are similar:

$$
\begin{aligned}
&\sum_{j=1}^{N_e}\iiint_{\Omega_e}\mu\left\{u_{x_j}^{e}\frac{\partial\psi_j^e}{\partial y}\frac{\partial\psi_i^e}{\partial x}+u_{y_j}^{e}\left(\frac{\partial\psi_j^e}{\partial x}\frac{\partial\psi_i^e}{\partial x}+2\frac{\partial\psi_j^e}{\partial y}\frac{\partial\psi_i^e}{\partial y}+\frac{\partial\psi_j^e}{\partial z}\frac{\partial\psi_i^e}{\partial z}\right)+u_{z_j}^{e}\frac{\partial\psi_j^e}{\partial y}\frac{\partial\psi_i^e}{\partial z}\right\}dxdydz\\
&\quad=\iiint_{\Omega_e}\left(\rho f_y\psi_i^e-P\frac{\partial\psi_i^e}{\partial y}\right)dx\,dy\,dz+\iint_{\partial\Omega_e}\psi_i^e P n_y\,ds+ \qquad (4.7)\\
&\quad\sum_{j=1}^{N_e}\iint_{\partial\Omega_e}\mu\psi_i^e\left\{u_{x_j}^{e}\frac{\partial\psi_j^e}{\partial y}n_x+u_{y_j}^{e}\left(\frac{\partial\psi_j^e}{\partial x}n_x+2\frac{\partial\psi_j^e}{\partial y}n_y+\frac{\partial\psi_j^e}{\partial z}n_z\right)+u_{z_j}^{e}\frac{\partial\psi_j^e}{\partial y}n_z\right\}ds.
\end{aligned}
$$

and

$$
\begin{aligned}
&\sum_{j=1}^{N_e}\iiint_{\Omega_e}\mu\left\{u_{x_j}^{e}\frac{\partial\psi_j^e}{\partial z}\frac{\partial\psi_i^e}{\partial x}+u_{y_j}^{e}\frac{\partial\psi_j^e}{\partial z}\frac{\partial\psi_i^e}{\partial y}+u_{z_j}^{e}\left(\frac{\partial\psi_j^e}{\partial x}\frac{\partial\psi_i^e}{\partial x}+\frac{\partial\psi_j^e}{\partial y}\frac{\partial\psi_i^e}{\partial y}+2\frac{\partial\psi_j^e}{\partial z}\frac{\partial\psi_i^e}{\partial z}\right)\right\}dxdydz\\
&\quad=\iiint_{\Omega_e}\left(\rho f_z\psi_i^e-P\frac{\partial\psi_i^e}{\partial z}\right)dx\,dy\,dz+\iint_{\partial\Omega_e}\psi_i^e P n_z\,ds+ \qquad (4.8)\\
&\quad\sum_{j=1}^{N_e}\iint_{\partial\Omega_e}\mu\psi_i^e\left\{u_{x_j}^{e}\frac{\partial\psi_j^e}{\partial z}n_x+u_{y_j}^{e}\frac{\partial\psi_j^e}{\partial z}n_y+u_{z_j}^{e}\left(\frac{\partial\psi_j^e}{\partial x}n_x+\frac{\partial\psi_j^e}{\partial y}n_y+2\frac{\partial\psi_j^e}{\partial z}n_z\right)\right\}ds.
\end{aligned}
$$

The components of the global system of equation are obtained by summing equations (4.6), (4.7), and (4.8) over all the elements of the mesh. This will generate $3N$ equations for $3N+E$ variables, $3N$ components of velocities defined at the N nodes of the mesh and E

components of pressure defined at the centers of E elements of the mesh. The remaining E equations are obtained by adding the incompressibility expression (3.2) to the system.

Variational Form of the Incompressibility Expression The variational form of the x-component of the momentum equations is constructed by multiplying the residual error function from equation (3.2):

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}$$

by a test function $\psi(x, y, z)$ and integrating over the problem's domain Ω:

$$\iiint_\Omega \psi(x, y, z) \left\{ \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} \right\} dx\, dy\, dz = 0.$$

Use of the divergence theorem converts the above equation to the following form:

$$\iint_\Omega \left(u_x \frac{\partial \psi}{\partial x} + u_y \frac{\partial \psi}{\partial y} + u_z \frac{\partial \psi}{\partial z} \right) dx\, dy - \int_{\partial\Omega} \psi \left(u_x n_x + u_y n_y + u_z n_z \right) ds = 0. \tag{4.9}$$

Due to boundary conditions, the integral over the domain boundary in the formula above is zero.

Finite Element Algorithm of the Incompressibility Expression In the equation (4.9), the integrals over the domain Ω and it's boundary $\partial\Omega$ can be calculated by adding contributions from integrals over each individual element of the mesh and boundaries of the elements. This can be shown by considering an element of the mesh, Ω_e.

After substituting (4.5) and (4.4) into (4.9) written for an element and requiring that it is satisfied for any Ψ_i^e, we get the following equations for $i = 1, ..., N_e$:

$$\sum_{j=1}^{N_e} \iiint_{\Omega_e} \left(\frac{\partial \psi_i^e}{\partial x} {u_x}_j^e + \frac{\partial \psi_i^e}{\partial y} {u_y}_j^e + \frac{\partial \psi_i^e}{\partial z} {u_z}_j^e \right) dx\, dy\, dz = 0. \tag{4.10}$$

Summing equations (4.10) for $i = 1, ..., N_e$ generates remaining E equations for solving conservation of momentum problem. Thus, solving the momentum equation is reduced to solving the system of linear equations (4.10) and (4.6) - (4.8). Properties of this system is discussed in chapter 5 on page 61.

4.1.2 Approximation of Conservation of Mass Equation

4.1.2.1 Variational Form

.

The variational form of the mass conservation equation is constructed in a manner similar to the variational form of the momentum conservation equations and is as follows:

$$\iint_\Omega \psi(x,y)\left\{\frac{\partial h}{\partial t} - \dot{a} + \frac{\partial (hU_x)}{\partial x} + \frac{\partial (hU_y)}{\partial y} - \frac{\partial}{\partial x}\left(k_{xx}\frac{\partial h}{\partial x}\right) - \frac{\partial}{\partial y}\left(k_{yy}\frac{\partial h}{\partial y}\right)\right\} dx\,dy = 0. \tag{4.11}$$

Use of the divergence theorem converts the above equation to the following form:

$$\begin{aligned}\iint_\Omega \left(\psi\frac{\partial h}{\partial t} - \psi\dot{a} - hU_x\frac{\partial \psi}{\partial x} - hU_y\frac{\partial \psi}{\partial y} + k_{xx}\frac{\partial h}{\partial x}\frac{\partial \psi}{\partial x} + k_{yy}\frac{\partial h}{\partial y}\frac{\partial \psi}{\partial y}\right) dx\,dy \\ + \int_{\partial\Omega} \psi h\left(U_x n_x + U_y n_y\right) ds + \int_{\partial\Omega} \psi\left(k_{xx}\frac{\partial h}{\partial x}n_x + k_{yy}\frac{\partial h}{\partial y}n_y\right) ds = 0.\end{aligned} \tag{4.12}$$

Due to boundary condition (3.5), the last integral in the formula above is zero. The other boundary integral either equal zero at the boundaries where Dirichlet conditions are specified (due to choice of an arbitrary test functions ψ) or can be rewritten as

$$\int_{\partial\Omega} \psi h\left(U_x n_x + U_y n_y\right) ds = \int_{\partial\Omega_1} \psi q_0\, ds,$$

where flux q_0 is boundary flux (see (2.13)) and $\partial\Omega_1$ is the part of the boundary where boundary flux is specified..

Then, the variational form of the mass equation is

$$\iint_{\Omega}\left(\psi\frac{\partial h}{\partial t}-\psi\dot{a}-hU_x\frac{\partial\psi}{\partial x}-hU_y\frac{\partial\psi}{\partial y}+k_{xx}\frac{\partial h}{\partial x}\frac{\partial\psi}{\partial x}+k_{yy}\frac{\partial h}{\partial y}\frac{\partial\psi}{\partial y}\right)dx\,dy \\ +\int_{\partial\Omega_1}\psi q_0\,ds=0. \tag{4.13}$$

4.1.2.2 Finite Element Algorithm

In the equation (4.13), the integrals over the domain Ω and it's boundary $\partial\Omega$ can be calculated by adding contributions from integrals over each individual element of the mesh and boundaries of the elements. This can be shown by considering an element of the mesh, Ω_e. Let's consider the following approximate solution h^e and local shape functions ψ^e over the element of the form:

$$h^e=\sum_{j=1}^{N_e}h_j^e\psi_j^e,\quad \psi^e=\sum_{i=1,N_e}\Psi_i^e\psi_i^e, \tag{4.14}$$

where N_e is the number of nodes in Ω_e.

After substituting ψ^e into the variational equation over Ω_e and requiring that it is satisfied for any Ψ_i^e, the linear system for the element Ω_e is obtained:

$$\iint_{\Omega_e}\left(\psi_i^e\frac{\partial h}{\partial t}-\psi_i^e\dot{a}-hU_x^e\frac{\partial\psi_i^e}{\partial x}-hU_y^e\frac{\partial\psi_i^e}{\partial y}+k_{xx}^e\frac{\partial h}{\partial x}\frac{\partial\psi_i^e}{\partial x}+k_{yy}^e\frac{\partial h}{\partial y}\frac{\partial\psi_i^e}{\partial y}\right)dx\,dy \\ +\int_{\partial\Omega_e}\psi_i^e h\left(U_x^e n_x+U_y^e n_y\right)ds\,ds=0,\quad i=1,2,\ldots,N_e$$

We assume that column-average velocities u_x^e and u_y^e and diffusion coefficients k_{xx}^e and k_{yy}^e are defined at the centroid of the elements.

After substituting h^e into above formula and approximating the time variable with a difference scheme, we get the following equations for $i = 1, ..., N_e$:

$$\iint_{\Omega_e} \left\{ \psi_i^e \tfrac{1}{\Delta t} \left(\sum_{j=1}^{N_e} h_j^{e,m+1} \psi_j^e - \sum_{j=1}^{N_e} h_j^{e,m} \psi_j^e \right) - \psi_i^e \dot{a} \right.$$
$$- \sum_{j=1}^{N_e} \left(h_j^{e,m+1} \psi_j^e \right) U_x^e \tfrac{\partial \psi_i^e}{\partial x} - \sum_{j=1}^{N_e} \left(h_j^{e,m+1} \psi_j^e \right) U_y^e \tfrac{\partial \psi_i^e}{\partial y}$$
$$\left. + k_{xx}^e \tfrac{\partial \left(\sum_{j=1}^{N_e} h_j^{e,m+1} \psi_j^e \right)}{\partial x} \tfrac{\partial \psi_i^e}{\partial x} + k_{yy}^e \tfrac{\partial \left(\sum_{j=1}^{N_e} h_j^{e,m+1} \psi_j^e \right)}{\partial y} \tfrac{\partial \psi_i^e}{\partial y} \right\} dx\, dy$$
$$+ \int_{\partial \Omega_e} \psi_i^e \left(\sum_{j=1}^{N_e} h_j^{e,m+1} \psi_j^e \right) \left(U_x^e n_x + U_y^e n_y \right) \, ds = 0.$$

After some rearrangement, we get the following system:

$$\sum_{j=1}^{N_e} \iint_{\Omega_e} \left\{ \psi_i^e \tfrac{1}{\Delta t} \left(h_j^{e,m+1} \psi_j^e - h_j^{e,m} \psi_j^e \right) - \psi_i^e \dot{a} \right.$$
$$- \left(h_j^{e,m+1} \psi_j^e \right) U_x^e \tfrac{\partial \psi_i^e}{\partial x} - \left(h_j^{e,m+1} \psi_j^e \right) U_y^e \tfrac{\partial \psi_i^e}{\partial y}$$
$$\left. + k_{xx}^e \tfrac{\partial \left(h_j^{e,m+1} \psi_j^e \right)}{\partial x} \tfrac{\partial \psi_i^e}{\partial x} + k_{yy}^e \tfrac{\partial \left(h_j^{e,m+1} \psi_j^e \right)}{\partial y} \tfrac{\partial \psi_i^e}{\partial y} \right\} dx\, dy$$
$$+ \sum_{j=1}^{N_e} \int_{\partial \Omega_e} \psi_i^e \left(h_j^{e,m+1} \psi_j^e \right) \left(U_x^e n_x + U_y^e n_y \right) \, ds = 0.$$

4.1.2.3 Finite Element Calculations

The above equations can be rewritten as follows:

$$\sum_{j=1}^{N_e} \left(C_{ij}^e + K_{ij}^e + \sigma_{ij}^e \right) h_j^{e,m+1} = f_i^e + \sum_{j=1}^{N_e} C_{ij}^e h_j^{e,m}, \quad i = 1, 2, \ldots, N_e \tag{4.15}$$

where

$$C_{ij}^e = \iint_{\Omega_e} \frac{\psi_i^e \psi_j^e}{\Delta t} \, dx\, dy, \qquad \sigma_{ij}^e = \int_{\partial \Omega_e} \psi_i^e \psi_j^e \left(U_x^e n_x + U_y^e n_y \right) \, ds,$$

$$K_{ij}^e = \iint_{\Omega_e} \left(-U_x^e \psi_j^e \frac{\partial \psi_i^e}{\partial x} - U_y^e \psi_j^e \frac{\partial \psi_i^e}{\partial y} + k_{xx}^e \frac{\partial \psi_j^e}{\partial x} \frac{\partial \psi_i^e}{\partial x} + k_{yy}^e \frac{\partial \psi_j^e}{\partial y} \frac{\partial \psi_i^e}{\partial y} \right) dx\, dy,$$

$$f_i^e = \iint_{\Omega_e} \psi_i^e \dot{a}\, dx\, dy,$$

The components of the global system of equation are obtained by summing equations (4.15) over all the elements of the mesh:

$$(C + K + \sigma) h^{m+1} = F + C h^m, \tag{4.16}$$

where the matrix and vector entries are given by

$$C_{ij} = \sum_{e=1}^{E} C_{ij}^e \tag{4.17}$$

$$K_{ij} = \sum_{e=1}^{E} K_{ij}^e \tag{4.18}$$

$$\sigma_{ij} = \sum_{e=1}^{E} \sigma_{ij}^e \tag{4.19}$$

$$F_i = \sum_{e=1}^{E} f_i \tag{4.20}$$

where E is the number of elements.

Thus, solving the prognostic equation is reduced to solving a system of linear equations with matrix $C + K + \sigma$, which is called the stiffness matrix of the system.

4.2 FEM formulation of an Ice Shelf Morland Equations

In this section, we construct the FEM formulation of Morland ice shelf/ice stream model discussed in the previous chapter. The variational forms of momentum and mass equations, as well as the final systems of linear equations generated by FEM are derived.

4.2.1 Approximation of the Diagnostic Equation

4.2.1.1 Variational Form

The variational form of the x-component of the momentum equations is constructed by multiplying the residual error function from equation (3.24a) by a test function $\psi(x, y)$ and integrating over the problem's domain Ω:

For the x-component of the momentum equations, the Galerkin method requires that:

$$\begin{aligned}\iint_\Omega \psi(x,y)\Bigg[\frac{\partial}{\partial x}\left(\bar{\mu}h\left(\frac{\partial U_x}{\partial x}+\frac{1}{2}\frac{\partial U_y}{\partial y}\right)\right)+\frac{\partial}{\partial y}\left(\bar{\mu}h\frac{1}{4}\left(\frac{\partial U_x}{\partial y}+\frac{\partial U_y}{\partial x}\right)\right)\\ -\frac{\rho g h}{4}(1-\alpha_x g_{bx})\frac{\partial z_s}{\partial x}+\frac{\rho g h}{4}g_{bx}\frac{\partial z_b}{\partial x}\Bigg]\,dx\,dy=0.\end{aligned} \tag{4.21}$$

Use of divergence theorem converts equation (4.21) to the following form:

$$\begin{aligned}&\iint_\Omega\left(\bar{\mu}h\left(\frac{\partial U_x}{\partial x}+\frac{1}{2}\frac{\partial U_y}{\partial y}\right)\frac{\partial\psi}{\partial x}+\bar{\mu}h\frac{1}{4}\left(\frac{\partial U_x}{\partial y}+\frac{\partial U_y}{\partial x}\right)\frac{\partial\psi}{\partial y}\right)dx\,dy\\ &+\iint_\Omega\frac{\rho g h}{4}\left((1-\alpha_x g_{bx})\frac{\partial z_s}{\partial x}-g_{bx}\frac{\partial z_b}{\partial x}\right)\psi\,dx\,dy\\ &-\int_{\partial\Omega}\psi\left[\bar{\mu}h\left(\frac{\partial U_x}{\partial x}+\frac{1}{2}\frac{\partial U_y}{\partial y}\right)+\bar{\mu}h\frac{1}{4}\left(\frac{\partial U_x}{\partial y}+\frac{\partial U_y}{\partial x}\right)\right]ds=0.\end{aligned} \tag{4.22}$$

The boundary contour integral in the equation above can be re-written taking into account discussion of the boundary conditions on page 39. Contour integrals along the contour where Dirichlet conditions are applied can be made equal to zero by choosing the arbitrary functions ψ while the contour integrals along the ice-shelf front where Newman's conditions are applied can be replaced by the integral::

$$\int_{\partial\Omega} \frac{\rho g h^2}{8}\left(1-\frac{\rho}{\rho_w}\right)\psi\, ds$$

Then the $x-$ and $y-$ components of the diagnostic equation look as follows:

$$\begin{aligned}
&\iint_\Omega \left(\bar{\mu} h\left(\frac{\partial U_x}{\partial x}+\frac{1}{2}\frac{\partial U_y}{\partial y}\right)\frac{\partial \psi}{\partial x}+\bar{\mu} h\frac{1}{4}\left(\frac{\partial U_x}{\partial y}+\frac{\partial U_y}{\partial x}\right)\frac{\partial \psi}{\partial y}\right)dx\,dy \\
&+\iint_\Omega \frac{\rho g h}{4}\left((1-\alpha_x g_{bx})\frac{\partial z_s}{\partial x}-g_{bx}\frac{\partial z_b}{\partial x}\right)\psi\,dx\,dy=\int_{\partial\Omega}\frac{\rho g h^2}{8}\left(1-\frac{\rho}{\rho_w}\right)\psi\,ds;
\end{aligned} \tag{4.23}$$

$$\begin{aligned}
&\iint_\Omega \left(\bar{\mu} h\left(\frac{\partial U_y}{\partial y}+\frac{1}{2}\frac{\partial U_x}{\partial x}\right)\frac{\partial \psi}{\partial y}+\bar{\mu} h\frac{1}{4}\left(\frac{\partial U_x}{\partial y}+\frac{\partial U_y}{\partial x}\right)\frac{\partial \psi}{\partial x}\right)dx\,dy \\
&+\iint_\Omega \frac{\rho g h}{4}\left((1-\alpha_y g_{by})\frac{\partial z_s}{\partial y}-g_{by}\frac{\partial z_b}{\partial y}\right)\psi\,dx\,dy=\int_{\partial\Omega}\frac{\rho g h^2}{8}\left(1-\frac{\rho}{\rho_w}\right)\psi\,ds,
\end{aligned} \tag{4.24}$$

where $\partial\Omega$ is an ice-front where ice shelf floats in hydrostatic equilibrium with seawater.

4.2.1.2 Finite Element Algorithm

Equations (4.23)-(4.24) apply for the domain as a whole, but they also apply for a particular sub-domain, or element. Thus we would have a set of element equations corresponding to the above, where only Ω and $\delta\Omega$ would be changed to Ω^e and $\delta\Omega^e$ for element e. Let's consider shape functions $\psi_j^e = \psi_j^e(x, y)$. For an element with $N_e = 4$ nodes, the x-, y- components of the velocity at any point within the element can be expressed as a sum of values at the nodes times shape functions evaluated at the point:

$$U_x^e = \sum_{j=1}^{N_e} u_{xj}^{\ e} \psi_j^e, \quad U_y^e = \sum_{j=1}^{N_e} u_{yj}^{\ e} \psi_j^e, \tag{4.25}$$

where ${u_x}_i^e$ is the value of u_x an node i in element e, etc.

After substituting $\psi^e = \sum_{i=1,N_e} \Psi_i^e \psi_i^e$, into above formula and requiring that it is satisfied for any Ψ_i^e, we get the following equations for $i = 1, ..., N_e$ (N_e - is the number of elements) for the x-component of the momentum equation:

$$\iint_{\Omega_e} \left(\bar{\mu} h \left(\frac{\partial u_x}{\partial x} + \frac{1}{2} \frac{\partial u_y}{\partial y} \right) \frac{\partial \psi_i^e}{\partial x} + \bar{\mu} h \frac{1}{4} \left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \right) \frac{\partial \psi_i^e}{\partial y} \right) dx\, dy =$$
$$- \iint_{\Omega} \frac{\rho g h}{4} \left((1 - \alpha_x g_{bx}) \frac{\partial z_s}{\partial x} - g_{bx} \frac{\partial z_b}{\partial x} \right) \psi_i^e \, dx\, dy + \int_{\partial\Omega} \frac{\rho g h^2}{8} \left(1 - \frac{\rho}{\rho_w} \right) \psi_i^e \, ds.$$

Then, after substituting u_x^e, u_y^e into the above formula, we get the following equations for the x-component of the momentum equation:

$$\sum_{j=1}^{N_e} \iint_{\Omega_e} \left(\bar{\mu} h \left(u_{xj}^e \frac{\partial \psi_j^e}{\partial x} + \frac{1}{2} u_{yj}^e \frac{\partial \psi_j^e}{\partial y} \right) \frac{\partial \psi_i^e}{\partial x} + \bar{\mu} h \frac{1}{4} \left(u_{xj}^e \frac{\partial \psi_j^e}{\partial y} + u_{yj}^e \frac{\partial \psi_j^e}{\partial x} \right) \frac{\partial \psi_i^e}{\partial y} \right) dx\, dy$$
$$= - \iint_{\Omega_e} \frac{\rho g h}{4} \left((1 - \alpha_x g_{bx}) \frac{\partial z_s}{\partial x} - g_{bx} \frac{\partial z_b}{\partial x} \right) \psi_i^e \, dx\, dy + \int_{\partial\Omega} \frac{\rho g h^2}{8} \left(1 - \frac{\rho}{\rho_w} \right) \psi_i^e \, ds$$

The momentum balance equation for x-component, after some rearrangement, is as follows:

$$\sum_{j=1}^{N_e}\left[u_{xj}^e\iint_{\Omega_e}\bar{\mu}h\left(\frac{\partial\psi_j^e}{\partial x}\frac{\partial\psi_i^e}{\partial x}+\frac{1}{4}\frac{\partial\psi_j^e}{\partial y}\frac{\partial\psi_i^e}{\partial y}\right)dxdy+u_{yj}^e\iint_{\Omega_e}\bar{\mu}h\left(\frac{1}{2}\frac{\partial\psi_j^e}{\partial y}\frac{\partial\psi_i^e}{\partial x}+\frac{1}{4}\frac{\partial\psi_j^e}{\partial x}\frac{\partial\psi_i^e}{\partial y}\right)dxdy\right]$$
$$=-\iint_{\Omega_e}\frac{\rho g h}{4}\left((1-\alpha_x g_{bx})\frac{\partial z_s}{\partial x}-g_{bx}\frac{\partial z_b}{\partial x}\right)\psi_i^e\,dx\,dy+\int_{\partial\Omega}\frac{\rho g h^2}{8}\left(1-\frac{\rho}{\rho_w}\right)\psi_i^e\,ds. \tag{4.26}$$

Momentum equations for y- component of the velocity is similar:

$$\sum_{j=1}^{N_e}\left[u_{yj}^e\iint_{\Omega_e}\bar{\mu}h\left(\frac{\partial\psi_j^e}{\partial y}\frac{\partial\psi_i^e}{\partial y}+\frac{1}{4}\frac{\partial\psi_j^e}{\partial x}\frac{\partial\psi_i^e}{\partial x}\right)dxdy+u_{xj}^e\iint_{\Omega_e}\bar{\mu}h\left(\frac{1}{2}\frac{\partial\psi_j^e}{\partial x}\frac{\partial\psi_i^e}{\partial y}+\frac{1}{4}\frac{\partial\psi_j^e}{\partial y}\frac{\partial\psi_i^e}{\partial x}\right)dxdy\right]$$
$$==\iint_{\Omega_e}\frac{\rho g h}{4}\left((1-\alpha_y g_{by})\frac{\partial z_s}{\partial y}-g_{by}\frac{\partial z_b}{\partial y}\right)\psi_i^e\,dx\,dy+\int_{\partial\Omega}\frac{\rho g h^2}{8}\left(1-\frac{\rho}{\rho_w}\right)\psi_i^e\,ds. \tag{4.27}$$

The components of the global system of equation are obtained by summing equations (4.26 and (4.27) over all the elements of the mesh. This will generate $2N$ equations for $2N$ variables ($x-$ and $y-$ components of velocities defined at the N nodes of the mesh)

4.2.2 Approximation of the Prognostic Equation

Since the ice-stream/ice-shelf prognostic equation (3.28) is exactly the same as the ice-sheet prognostic equation (3.3), the finite element formulation of the equation as well as the final systems of linear equations generated by FEM will be the same. They have been discussed on page 53.

Chapter 5

PERFORMANCE ANALYSES OF SuperLU PARALLEL SOLVER

This chapter describes application and benchmarking of the multiprocessor software package SuperLU-DIST for solving large systems of sparse simultaneous linear equations generated by the three-dimensional full-Stokes model.

5.1 Introduction

Solving the system of linear equations (SLE) generated by a three-dimensional higher-order ice-sheet model is a demanding task because usage of indirect iterative methods is impossible while usage of direct banded-Gaussian elimination method is impractical due to types of matrices of the generated systems. Figure 5.1 on Page 63 is a scatter plot showing the non-zero entries from an actual matrix generated by FEM for the higher-order 3D model. It is a banded matrix with right and lower borders of non-zero elements. As it can be seen from the picture, the matrices generated by FEM for solving conservation of momentum equation have the following properties:

Matrices of the systems are not diagonally dominant. Since the diagonally dominance of the matrices is the necessary condition for the iterative methods to converge, that means that iterative methods cannot be used to solve these systems. For this reason, the direct methods, as opposed to iterative methods, are investigated for solving the equations generated by FEM.

Matrices of the systems do not have a strict banded structure. Non-zero elements on right and lower borders of the matrix (Figure 5.1 on Page 63) are entries generated by pressure terms in the momentum equations. These entries make the matrices non-banded. Without the banded matrix structure, storing the matrices as two dimensional arrays and straightforwardly applying banded-Gaussian elimination methods is impractical for large size problems.

Matrices have extremely large sizes. For example, space grid of size $50 \times 50 \times 10$ generates $\approx 10^5$ equations. But the space grid may range $1000 \times 1000 \times 50$ which generates $\approx 10^8$ equations.

Matrices of the systems are very sparse. Systems of equations that have many more zero entries than non-zero entries are called sparse [1]. For example, for a 3-D model for a rectangular region that is $50 \times 40 \times 5 = 10,000$ nodes, the system has $40,000$ independent variables (number of nodes $\times$ 3 velocity variables and 1 pressure variable) and the matrix of the system has $40,000 \times 40,000 = 1.6 \times 10^8$ elements. But only $81 \times 40,000 = 3.24 \times 10^6$ of them are non-zero elements, that is, only ≈ 100 entries per equation are non-zero.

Since indirect methods cannot be applied for solving ice-sheet systems, the choice is between different direct methods. Table 5.1 on Page 63 compares characteristics of dense matrix methods (based on Gauss elimination) and sparse matrix methods (such as SuperLU, UMFPACK) applied to a matrix of size $\approx 10^5$ generated by grid $50 \times 50 \times 10$.

[1]Defining a sparse matrix as a matrix with some fraction of nonzero entries is inappropriate. Instead, it is recognized that sparsity is an economic issue; if you can save time and memory by exploiting the zeros, then the matrix is sparse. The sparse matrix community defines a sparse matrix as any matrix with enough zeros that it pays to take advantage of them.

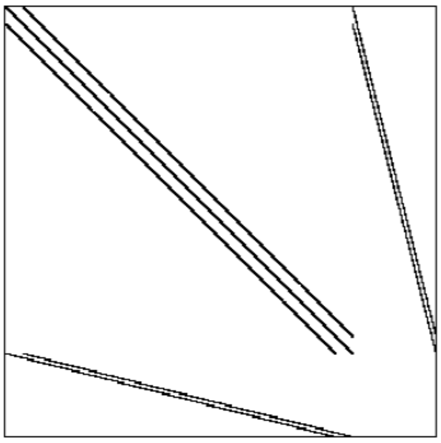

Figure 5.1. Scatter plot of non-zero entries in an ice-sheet matrix

Dense Matrix Algorithms	**Sparse Matrix Algorithms**
memory (store A): $O(n^2) \approx 80GB$	*memory* (store only non-zeros of A, row- and column- pointers, and nonvisible for the user matrices L and U): $O(n) \approx 0.16GB$
runtime (LU decomposition): $O(n^3) \approx 10^{15} FLOPs \approx 277$ hours[a]	*runtime* (perform operations only on nonzeros): $O(n^2) \sim O(n^2 log_2 n) \approx C \times (1 \sim 16.6) \times 10^{10} FLOPs \approx C \times (10 \sim 166)$ seconds. If constant $C \approx 100$, then runtime may range from 2 min to ~ 4 hours.

[a]for a typical processor that performs 10^9 FLOPs per second.

Table 5.1. Comparison of dense and sparse matrix algorithms for solving a system $Ax = b$ with matrix $A(n, n)$, where $n \approx 10^5$.

We can see from the table that application of dense matrix algorithms is impossible – it requires $\approx 80\,GB$ memory to solve even a modest size system and it takes about $270\,hours$ to solve the system. Sparse matrix algorithms require significantly less memory (they store only non-zero elements of the matrix and some additional needed matrices that describe the positions of these nonzeros within the total matrix). However, the run-time of the algorithm is in the range of $2\,min$ to $\sim 4\,hours.$ Ideally we want to solve systems of much higher-order (say, $200,000,000$ which is generated when we use a grid of $100 \times$

100×50 points in space) and solve them not once but $100,000$ times to simulate evolution of ice-sheet with time.

The above reasons have motivated us to explore the possibility of using multiprocessors to solve these systems of equations.

5.1.0.1 Which part of the code to parallelize?

To begin with, we have a question, should we write a parallel code ourselves or use some available software. The answer depends on how unique is the problem we are solving. Our code consists of two steps, (1) a specific step, generating a SLE from mathematical ice-sheet model using FEM, and (2) a general problem of solving the generated SLEs using a direct solver. Running time of the first step, building the SLEs, is more than an order less than the running time of the second step, solving the generated SLEs. Constructing the SLEs is $O(n)$, while solving the SLEs (for example, using serial SuperLU [2]) is $O(n^2) - O(n^2 log_2 n)$.[3]

We timed these two steps using serial SuperLU. Table 5.2 on Page 65 shows the timing results. Figure 5.2 on Page 66 displays the time required to build the system of linear equations using FEM and the time required to solve the system as functions of problem size.

We can see from the table that the first step, specific to our model, takes only about 2% of total running time for our benchmark problem of size $100,000$ and less than 1% for bigger size problems. We can see from the graph that time to build the SLEs increases

[2]The software application for solving general systems of linear equations described further in this chapter.

[3]The running time of the algorithm depends on the type of matrix of the problem because complexity of factorization of matrices depends on the type of the matrices.

linearly when problem size increases, while the time required to solve the systems grows much faster.

Grid	**Matrix Order**	**Building Time**	**Solving Time**	**$\%$ Building Time of Total Time**
$23 \times 23 \times 8$	16,084	5.07	43.68	10.40
$33 \times 33 \times 8$	33,304	11.85	208.81	5.37
$40 \times 40 \times 8$	49,047	16.06	382.40	4.03
$45 \times 45 \times 10$	78,174	26.18	1,118.99	2.29
$55 \times 55 \times 10$	116,994	39.32	1,768.93	2.17
$65 \times 65 \times 10$	163,614	55.47	3,046.54	1.79
$75 \times 75 \times 10$	218,034	73.74	7,919.75	0.92
$85 \times 85 \times 10$	280,254	102.18	15,521.83	0.65

Table 5.2. FEM matrix building time vs. Solving time (in sec).

The experiments show that parallelizing the first step of the UMISM code is not worth the effort – for a system of order $\approx 10^5$, solving the system takes $\sim 95 - 98\%$ of running time and building the system takes only $\sim 5 - 2\%$ of total running time. That is, it is enough to parallelize solving the SLEs part of the code!

Moreover, if we need to parallelize only solving the SLEs, then we can use freely available packages for solving SLEs on multiprocessors.

In this chapter, we explore an application of a distributed SuperLU software package to solving the system of linear equations generated by 3-D higher-order ice sheet model and evaluate and benchmark the performance characteristics of the package.

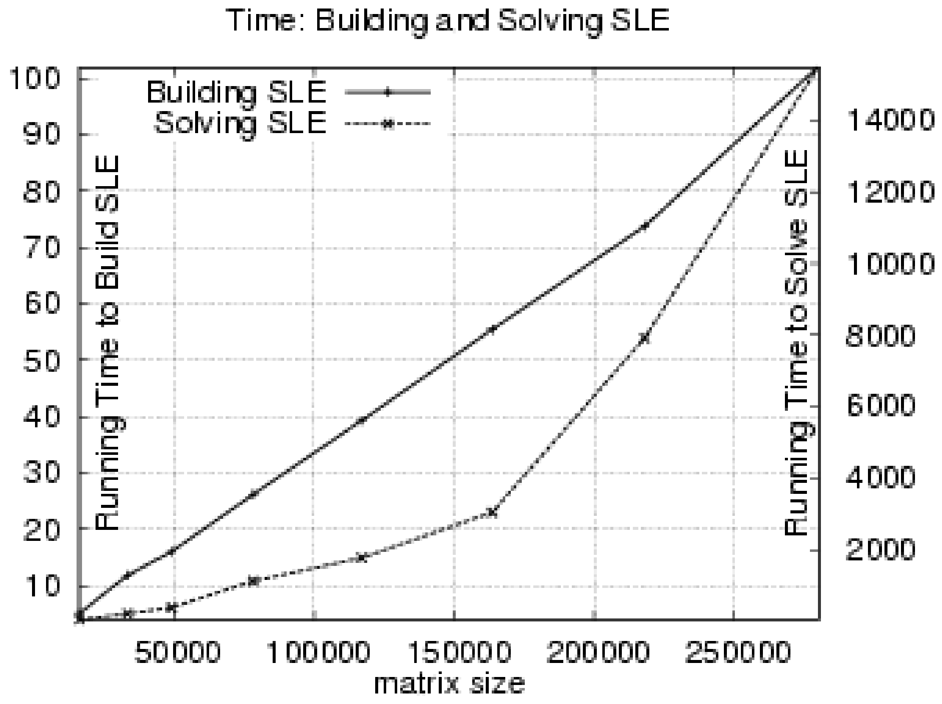

Figure 5.2. Running time required to build and solve SLE generated by Ice Sheet model (in seconds) as functions of matrix size. Left axis is used for scaling running time of building SLEs while right axis is used for scaling running time of solving the SLEs.

5.2 Distributed SuperLU

Solving large systems of sparse simultaneous linear equations is a common task in science and engineering problems. Over the years a great deal of work has been done in this area by researchers. Software to solve these problems has been developed and is readily available. In his Master's thesis, Rodney Jacobs [39] chose and evaluated two current software packages with respect to the ice sheet problem. One of them is SuperLU [7, 8, 44] – a library of ANSI C subroutines for solving general sparse linear systems. The principal developers are Xiaoye (Sherry) Li, James Demmel, and John Gilbert.[4] The SuperLU libraries are freely available for commercial and non-commercial use.

[4]Xiaoye (Sherry) Li, Computer Scientist, Lawrence Berkeley National Laboratory; James Demmel, Professor of Computer Science and Mathematics, University of California at Berkeley; and John Gilbert, Professor of Computer Science, University of California at Santa Barbara.

SuperLU comes in three versions, for single processor computers (known as $SuperLU$), for shared memory multiprocessors[5] (known as Multithreaded SuperLU or $SuperLU-MT$), and for distributed memory parallel computers (known as Distributed SuperLU or $SuperLU-DIST$). SuperLU-DIST was used in this work.

SuperLU-DIST uses Message Passing Interface (MPI) for interprocess communications. The authors claim these versions are designed to make optimum use of the sparsity of A and the computer's architecture with attention given to optimum use of cache memory and parallelism.

This section describes SuperLU-DIST package, the algorithm it uses, and distribution of matrices and vectors among processors used in SuperLU-DIST.

5.2.1 SuperLU Algorithm Phases

The solver is based on sparse Gaussian elimination. To solve a system of equations $Ax = b$, it uses factorization

$$A = D_r^{-1} P_r^{-1} LU P_c^{-1} D_c^{-1} \tag{5.1}$$

with following forward and backward substitution to solve for x

$$x = \left(D_r^{-1} P_r^{-1} LU P_c^{-1} D_c^{-1}\right)^{-1} b = D_c P_c U^{-1} L^{-1} P_r D_r b, \tag{5.2}$$

[5]A shared memory multiprocessor is a parallel computer that allows all processors to access any main memory location. Access to main memory by the processors is coordinated by the computer's hardware. Each processor in a distributed memory computer has its own memory. A communications network between the processors is used to share data and coordinate activities.

where

P_r is a row permutation matrix for maintaining stability,

P_c is a column permutation matrix for maintaining sparsity,

D_r and D_c are diagonal row and column scaling matrices chosen to make the diagonal elements large compared to the off-diagonal elements. This will minimize the sensitivity of the matrix to round off errors.

SuperLU computes each of these four matrices with various levels of control available to the user. Because SuperLU uses LU factorization, it can compute x for multiple right hand sides.

In SuperLU terminology, driver routines are the user-callable routines for performing major tasks. The expert driver, available in SuperLU-DIST, performs the following steps.

1. Equilibration of A by computing the row and column scaling matrices D_r and D_c so that $\bar{A} = D_r A D_c$ is better conditioned than A (this reduces round off errors and improves stability).
2. Row permutations of $\bar{A}$ for stability. If row permutations are done from the values of $\bar{a_{ij}}$ before any factorization is performed, then the process is called *static pivoting*. In some algorithms, the row permutations are determined during factorization. Such a process is called *threshold pivoting*. The interprocess communication required to perform threshold pivoting is not practical on a distributed memory parallel computer.
3. Column permutations of $\bar{A}$ to reduce fill-in of L and U and increase parallelism in SuperLU-DIST. This step is also called *symbolic factorization*; in this step, column

ordering is defined using fill-reducing heuristics.[6] In sparse LU factorization, some zero elements may become nonzeros at runtime due to factorization and pivoting. Predicting these elements can help avoid costly data structure variations during the factorization. The static symbolic factorization can identify the worst case fill-ins without knowing numerical values of elements. This enables the symbolic processing phase to be completely separated from numerical factorization. As a result, the symbolic computation needs to be performed only once for matrices with the same initial structure but different numerical values.

For the unsymmetric factorizations (that is, for factorization of general matrix A), the preordering for sparsity is less well understood than that for the Cholesky factorization (factorization of symmetric matrix $A = A^T$). Most unsymmetric ordering methods, SuperLU-DIST including, use the symmetric ordering techniques, called *Multiple Minimum Degree*, applied on a symmetrized matrix $A^T + A$, denoted as $MMD(A^T + A)$, or on a symmetrized matrix $A^T A$, denoted as $MMD(A^T A)$. In this technique, fill-reducing ordering is computed on a symmetric matrix $A^T + A$ or $A^T A$ and applied symmetrically to the rows and columns of matrix A.

The L and U factors generally have many more non-zero entries than A due to fill-in. Since P_r and P_c are computed before factorization begins, SuperLU-DIST can

[6]The process of factoring a sparse matrix is expressed by a directed acyclic task-dependency graph (DAG). The vertices of this directed acyclic graph (DAG) correspond to the tasks of factoring rows or columns or groups of rows and columns of the sparse matrix and the edges correspond to the dependencies between the tasks. A task is ready for execution if and only if all tasks with incoming edges to it have completed. Symbolic algorithms inexpensively compute an a-priori minimal task-dependency graph and near-minimal data dependency graph for factoring a general sparse matrix that are valid for any amount of pivoting induced by the numerical values during LU factorization.

determine what the fill-in requirements will be apriori and allocate the correct amount of memory.

Complexity of symbolic factorization is $O(nonzeros(L+U))$ [43] and may depend on matrix structure as well as the heuristic used in symbolic factorization step.

4. Numeric factorization LU with control of diagonal magnitude by replacing tiny pivots by $\sqrt{\epsilon} \parallel A \parallel$, where ϵ is a small number.
5. Triangular solves of the system of equations using L and U.
6. Iterative refinement to improve the solution if needed.
7. Computation of error bounds. SuperLU can compute the component-wise relative backward error (*BERR*). The meaning of BERR is that $\bar{x}$, the computed value of x, is the exact solution of the perturbed linear system of equations $(A+E)\bar{x} = b+f$, where

$$|e_{ij}| \leq BERR \times |a_{ij}| \text{ and } |f_j| \leq BERR \times |b_j| \text{ for all } i \text{ and } j. \tag{5.3}$$

The authors claim that by combining static pivoting with row and column scaling and iterative refinement, the distributed algorithm is as stable as partial pivoting for most matrices observed in actual applications. In cases where computations are not stable, BERR provides an indication of a problem.

In SuperLU-DIST, the most time-consuming steps (4) to (7) have been parallelized, while preprocessing and analysis steps (1) to (3) are mostly performed sequentially at present.

5.2.2 Distribution of Matrices among Processors

SuperLU routine takes the matrix A in either compressed-column format[7] or compressed-row format. The right hand side of the system of equations, b, may be presented to the routine as a dense vector if there is only a single right hand side, or as a dense matrix in column major order if there are multiple right hand sides. The solution x overwrites b. Both A and b are distributed among all processes using a distribution based on block rows. That is, each process owns a block of consecutive rows of A and b.

Matrices L and U are distributed among processes in a two-dimensional (2D) block-cyclic fashion. The routine first identifies the supernode boundary based on nonzero structure of L. A *supernode* is a range of columns of L such that the triangular block of L below the diagonal is completely filled. In addition, each row of L within this range of columns either has all zero entries or all non-zero entries. Because the supernodes are not necessarily symmetric, the U portion of the supernode does not have the same dense pattern as L. The matrix in Figure 5.3 illustrates such a partition.

Blocks of L and U are distributed among p processes that are arranged as a 2D grid of dimension $p_{row} \times p_{col} = p$. The user can set the shape of the process grid, such as 2×3 or 3×2, *etc.*. In block-cyclic mapping, block (I, J) $(0 \leq I, J \leq N - 1)$, where N is the number of supernodes, is mapped into the process at coordinate $((I-1) mod p_{row}, (J -$

[7]Compressed column format is a data format for A that is compatible with both SuperLU and UMFPACK. Three one-dimensional arrays are used to store a matrix in this format. One array is used to store the non-zero entries of A in column-row order. The second array is used to store the row number of each corresponding non-zero entry in the values array. The third array contains the index values of the first and second arrays where the first non-zero entry for each column of A is stored. The matrix column number is the index to this array.

We used a special modified compressed column structures and supporting routines designed by Rodney [39] that allowed us to exchange data with SuperLU software. Information about compressed column format can be found in Appendix B.

$1)modp_{col})$ of the process grid. In this 2D mapping, each block column of L is spread across every processor in a single column of the process grid. Figure 5.3 on Page 72 shows 2×3 process-grid and distribution of matrices L and U among the processes.

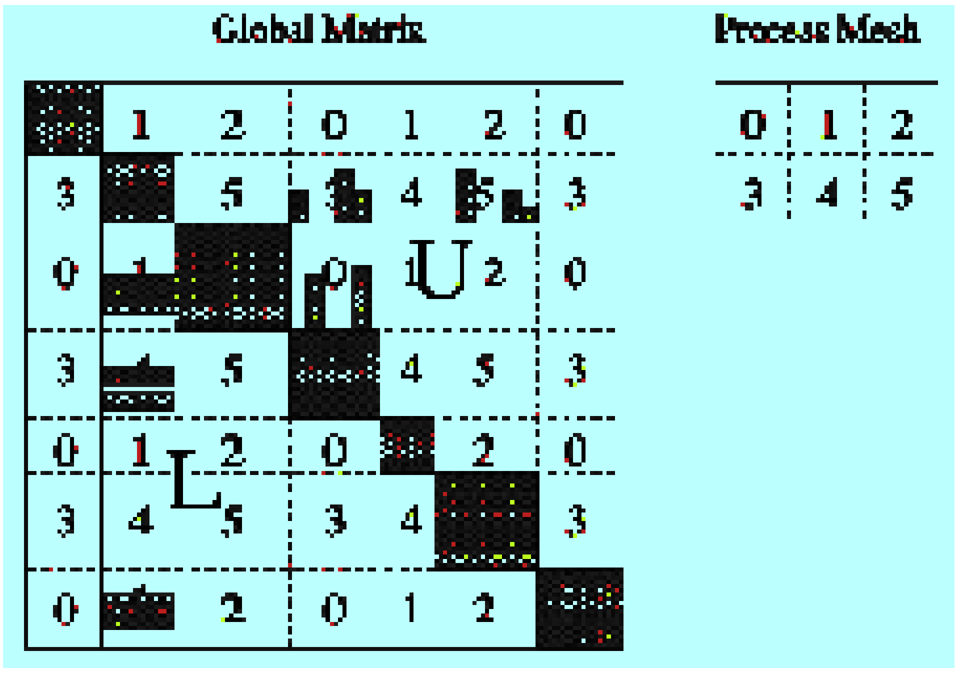

Figure 5.3. SuperLU-DIST 2D block-cyclic mapping of matrix to processes (from Baertschy & Li (2001)

In addition to default communicator MPI_COM_WORLD, process-groups are created using $p_{row} \times p_{col}$ processes. The majority of SuperLU's computation is updating the unfactored submatrix of the supernode using the following block mode update.

$$A(I, J) \leftarrow A(I, J) - L(I, K)U(K, J),$$

where A is the unfactored portion of the supernode, L and U are the factored portions of the supernode, I is the range of rows of the unfactored portion, J is the range of columns of the unfactored portion, and K is the number of columns of L and the number of rows of U in the supernode. This looks like a BLAS level 3 operation, and that is in fact what SuperLU-DIST uses. In the LU factorization, some communication occur only among the processes in a row or column and not among all processes, so creating process-groups using 2D process grid reduces inefficient addressing.

Thus, decomposition of matrices L and U into blocks of 2D submatrices and using 2D block-cyclic mapping allows the authors to exploit dense submatrices in L and U (”supernodes”), use Level 3 BLAS operations, reduce inefficient, indirect addressing (scatter/gather), and enhance load balance and scalability.

5.3 Performance Analyses of the Parallel Solver

5.3.1 Test Environment

We ran the tests on the Boston University SCV[8] IBM pSeries 690 (IBMp690) and 655 (IBMp655). Table 5.3 on Page 74 shows the characteristics of some of IBMp690 and IBMp655 nodes used to perform experiments described in this work.

IBMp690 composed of four nodes, named *kite.bu.edu*, *pogo.bu.edu*, *frisbee.bu.edu*, and *domino.bu.edu*, each consisting of Power4 processors running at 1.3 GHz and sharing 1 GB of memory per processor. There are three levels of cache on this machine. Each processor has a 32KB L1 cache and then each pair of processors share a 1.41MB L2 cache, and each set of eight processors share a 128MB L3 cache. The combined peak performance of p690 system is 580 GFLOPS.

[8]SCV stands for Scientific Computing and Visualization

IBMp655 is a 48-processor system composed of six nodes, named *twister.bu.edu*, *scrabble.bu.edu*, *marbles.bu.edu*, *crayon.bu.edu*, *litebrite.bu.edu* and *hotwheels.bu.edu*, each consisting of 8 Power4 processors running at 1.1 GHz and sharing 16 GB of memory. There are three levels of cache on this machine. Each processor has a 32KB L1 cache and then each pair of processors share a 1.41MB L2 cache, and each p655 node shares a 128MB L3 cache. Twister is the interactive machine for the entire set of pSeries machines and is the only one of the machines which users can log in to. The other machines are all reserved for batch processing.

Host	**Model**	**# Processors**	**Memory**	**Network**
twister	IBMp655	8×1.1 GHz	16GB	1Gbps Ethernet
kite	IBMp690	32×1.3 GHz	32GB	1Gbps Ethernet
frisbee	IBMp690	32×1.3 GHz	32GB	1Gbps Ethernet
pogo	IBMp690	32×1.3 GHz	32GB	1Gbps Ethernet

Table 5.3. Characteristics of Boston University IBMp690 and IBMp655 nodes used for computations.

5.3.2 Test Problems

To study the applicability of the package to our problem, we tested it on matrices of sizes varying from 16,000 to 163,600. Characteristics of the matrices are shown in Table 5.4 on Page 76. They include the problem's name or grid size (*Problem*), order of matrices (n), number of nonzeros in matrices A and in L and U factors (*fill-in*), gigaflops required to factorize the matrix, and the average number of nonzero elements in a supernode. The last characteristic can be a certain measure of the sparsity of the filled matrix.

To compare the properties of our matrix with the ones that have been solved on the multiprocessor by the SuperLU-DIST developers, we included Table 5.4 on Page 77 from [45] that shows the characteristics of benchmark matrices analyzed by Xiaoye Li and Yu Wang.

Table 5.5 on Page 85 shows other parameters of the matrices derived from the parameters shown in Table 5.4 and Table 5.4, such as the average number of nonzero elements of matrix A per row (which characterizes the sparsity of matrix A), the average number of nonzero elements of L and U per row (which characterizes how the sparsity of the problem changes after LU factorization), average number of rows per supernode (smaller the number, more evenly workload is distributed among processes), and number of megaflops, it took to factorize the matrix, divided by matrix order (which characterizes how fast running time growth when the problem size increases).

From Table 5.5, we can see that our matrices are the most dense matrices of all the benchmark matrices. Only problem *mixing-tank* is close in sparsity of A to our problems. It is close in size of A to our problem $33 \times 33 \times 8$, they have almost the same number of nonzero elements per row, but the size of our matrix is a little bit bigger than the size of matrix *mixing-tank*. Other characteristics of these two problems, such as sparsity of filled L and U, average size of a supernode, average number of FLOPS it took to factorize the matrices, are also close. Comparing these parameters with similar parameters of other benchmark matrices, we can conclude that matrices large in dimension and number of nonzeros require more time to factorize.

Problem	**Order** n	**nnz**(A)	**nnz**$(L+U-I)$ $\times 10^6$	**#supernodes** **(N)**	**Flops** $\times 10^9$	$\frac{nnz(L+U-I)}{N}$ $\times 10^3$
$23 \times 23 \times 8$	16,084	1,051,446	23.92	1,795	28.80	13.32
$33 \times 33 \times 8$	33,304	2,207,046	59.06	3,262	88.20	18.11
$40 \times 40 \times 8$	49,047	3,268,008	107.55	4,849	216.06	22.18
$65 \times 65 \times 10$	163,614	14,561,646	610.89	16,376	2,134.25	37.30

Table 5.4. Characteristics of ice-sheet benchmark matrices used in this work. They include the problem's name, order of matrices (n), number of nonzeros in matrices, FLOPs required to factorize the matrices, and the average number of nonzero elements in a supernode.

The most important characteristic is the sparsity of matrices L and U which is shown as the number of nonzero elements of the matrix $L + U - I$. It is a key parameter in evaluating speed and memory requirements of the algorithm.

Since the algorithm factorization time depends on the the number of nonzeros of this matrix, the faster this number grows when the size of the problem increases, the more time is required to factorize bigger matrices and the more memory is required to store the matrices.

Figure 5.5 on Page 78 show the number of nonzero elements in matrices A and $L + U - I$ as functions of problem size. While number of nonzero elements in the matrix A is proportional to the order of the matrix (linear graph in Figure 5.5, which is so close to axis x that it is barely noticeable), the number of nonzeros of matrices L and U (number of nonzeros of matrices $L + U - I$ on the figure) grows much faster than the number of nonzeros of A. Our matrices result in the most dense matrices L and U after factorization among all benchmark matrices of similar sizes in Table 5.5. I speculate that the reason is the fact that our matrices A have many zero diagonal elements and elements at right

Matrix	Order	$nnz(A)$	$nnz(L+U)$ $\times 10^6$	N	$Flops(F)$ $\times 10^9$	$nnz(L+U)/N$ $\times 10^3$
bbmat	38744	1771722	36.1	7128	26.28	5.06
fidapm11	22294	623554	25.6	2610	24.11	9.81
wang4	26064	177196	10.7	6961	8.79	1.54
twotone	120750	1224224	11.4	38939	7.59	0.29
mixing-tank	29957	1995041	44.6	2538	76.84	17.57
inv-extrusion-1	30412	1793881	30.2	4129	32.46	7.31
ecl32	51993	380415	41.9	19168	60.74	2.19
ir	186230	2910202	(MMD) 132.7	25778	148.48	5.15
			(ND) 89.8	31498	66.19	2.85
dds.quadratic	380698	15844364	(MMD) 642.5	36877	2120.75	17.42
			(ND) 325.0	41362	491.04	7.86
dds15	834575	13100653	(MMD) 875.3	114929	1576.43	7.62
			(ND) 526.6	141060	600.58	3.73

Figure 5.4. Characteristics of benchmark matrices analyzed in Li & Wang (2003). Characteristics include the number of supernodes N, the number of nonzeros in L and U using MMD ordering on $A^T + A$, and, for some matrices, the number of nonzeros in L and U using the nested dissection (ND) ordering.

and lower borders (see Fig. 5.1). This fact may increase the number of column and row permutations and, consequently, increase fill-ins.

5.3.3 Performance Characteristics

The number of non-zero entries in the L and U factors is a key parameter in evaluating speed and memory requirements of an algorithm. The number of floating point operations and the amount of memory required tend to increase with increasing numbers of non-zero entries, and runtime tends to increase with increasing numbers of floating point operations. In evaluating performance of the algorithm, we will look at numbers of non-zero entries in L and U, number of floating point operations, runtime, efficiency, and scalability of the algorithm.

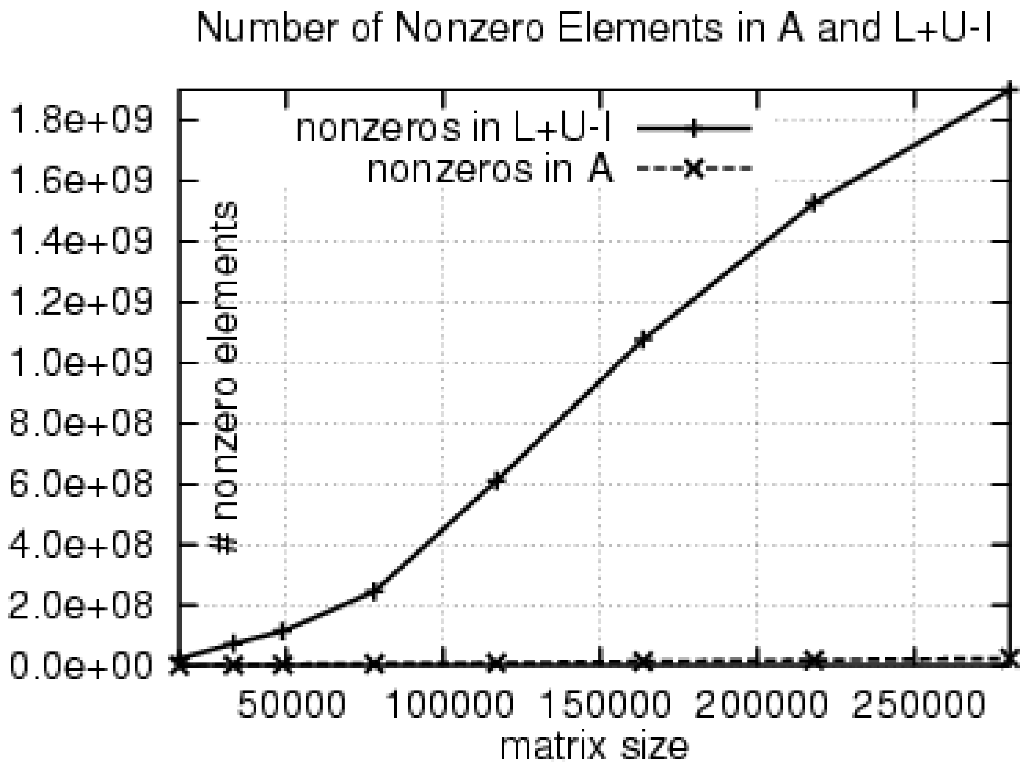

Figure 5.5. Number of nonzero elements in matrices A and $L + U - I$ as functions of problem size.

5.3.3.1 Scalability

Figure 5.6 on Page 80 demonstrates timing results, running time and efficiency, obtained using ”square” processor-grids $P = 1, 2 \times 2, 3 \times 3, 4 \times 4, 5 \times 5, 5 \times 6,$ and 4×8 for computing benchmark problems described in Table 5.4 on Page 76.

Efficiency (E) is a measure of process utilization in a parallel program, relative to the serial program. It can be also defined as the *speedup*, the ratio of the runtime on one processor to that of parallel program running on p processors, divided into number of processors p:

$$Speedup = \frac{T(1)}{T(p)}, \qquad E = \frac{T(1)}{p \cdot T(p)},$$

where $T(p)$ - run time with p processors.

Efficiency of a parallel algorithm depends mainly on how the workload is distributed and how much time is spent in communication. For dense matrices, the LU factorization algorithm have been shown to exhibit good scalability, where it can be approximately maintained as the number of processors increases when the memory requirements per processor are held constant [9]. For sparse matrices, however, the efficiency is much harder to predict since the sparsity patterns vary with different applications.

Figure 5.6 shows results that are as expected on multiprocess computations, that is,

1. running time decreases as the number of processors increases; and it decreases faster for smaller size problem than for bigger one;
2. as expected, the efficiency degrades faster for problems of smaller size and slower for problems of bigger size;
3. for problems of order $33,000$ or higher, efficiency still maintains at 40% even with 32 processors.

Thus, we can conclude that the factorization phase scales quite well for our type of matrices.

Table 5.6 on Page 86 shows speedup reached on 2×2, 3×3, 4×4, 5×5, 5×6, 4×8 processor-grids. As can be seen from the table,

- for small size problems, speedup levels off at 16 processors;
- for big size problems, speedup levels off at 25-30 processors.

5.3.3.2 Workload Distribution and Optimal Process-Block Size

Efficiency of a parallel algorithm depends mainly on how the workload is distributed and how much time is spent in communication. One way to measure workload distribution

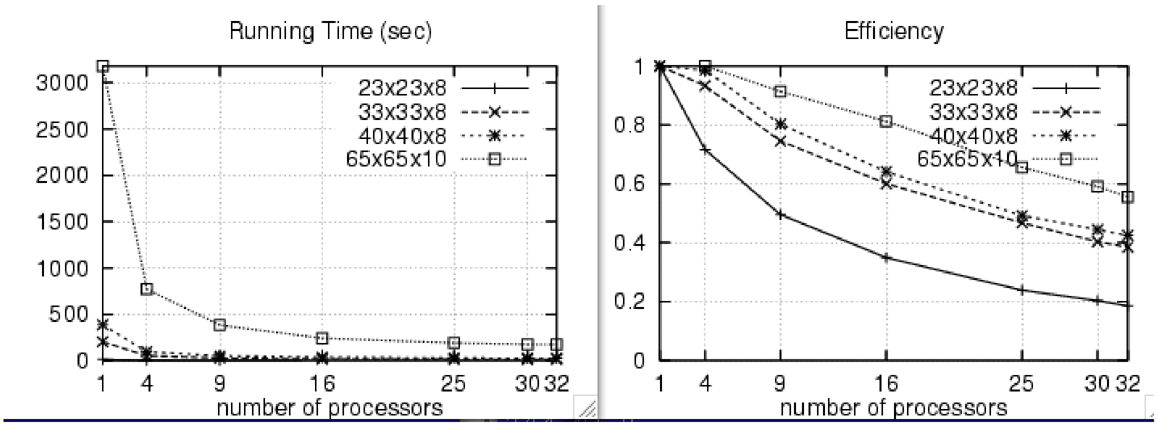

Figure 5.6. Running time and efficiency as functions of number of processors and problem size. Processor-grids are 1, 2×2, 3×3, 4×4, 5×5, 5×6, 4×8. Left: running time. Right: efficiency.

is to compute the *load balance factor, LBF*, which is the average workload divided by the maximum workload. It is clear that $0 < LBF \leq 1$, and higher LBF indicates better load balance. The parallel runtime is at least the runtime of the slowest process, whose workload is highest.

$$LBF = \frac{\sum_i f_i}{p \cdot \max_i f_i} = \frac{average\, workload}{maximum\, workload}, \quad 0 < LBF \leq 1,$$

where f_i - number of floating-point operations performed on processor i.

Figure 5.7 on Page 81 shows a block-cyclic distribution of a matrix among four processors on two different processor-grids, rectangular 1×4 grid and square 2×2 grid. From the figure, we can see that a square processor-grids provides

1. better, more even, workload distribution. For example, in a rectangular processor grid 1×4, workload of Processor 0 is much less than than workload of Processors 1, 2, or 3 while in a square processor grid 2×2, all processors more evenly distributed workload.

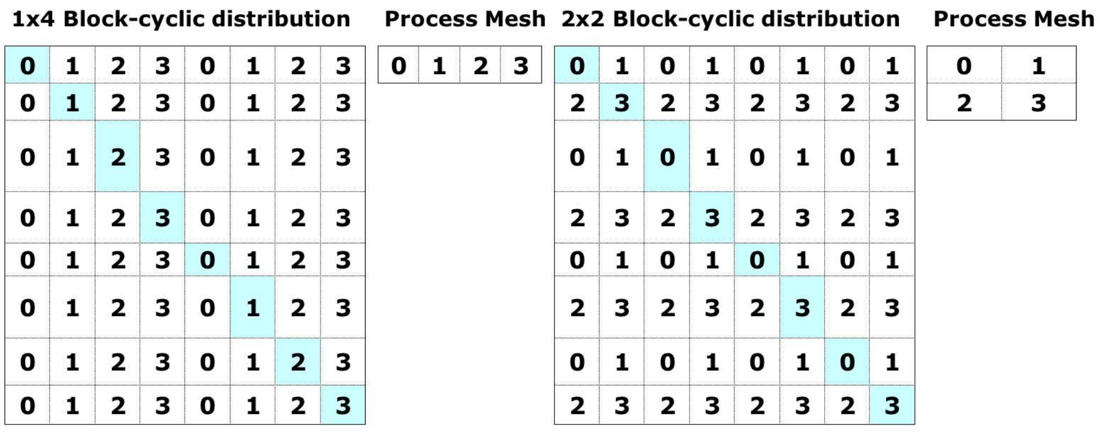

Figure 5.7. Block-cyclic distribution of a matrix on a rectangular 1×4 and square 2×2 processor-grids.

2. less communication overhead. In a rectangular processor-grid 1×4 distribution, each processor has to communicate with every other processor, while in a square processor-grid 2×2 distribution, each processor has to communicate only with processors in its row and column. The square grid minimizes the number of communications.

To study how the shape of processor-grids affect the workload distribution among processors, we have run tests with different size problems on different shape processor-grids, varying from more rectangular $n \times 1$ and $n \times 2$ to more ”square” grids $n \times 4$:

$n \times 1 = 1, 2 \times 1, 3 \times 1, 4 \times 1, ..., 32 \times 1$,

$n \times 2 = 1 \times 2, 2 \times 2, 3 \times 2, ..., 16 \times 2$, and

$n \times 4 = 1 \times 4, 2 \times 4, ..., 8 \times 4$.

Figures 5.8 on Page 82 show the load balance factor for the factorization phase of the algorithm for problems of size $23 \times 23 \times 8$ and $65 \times 65 \times 10$ calculated using different shape processor-grids.

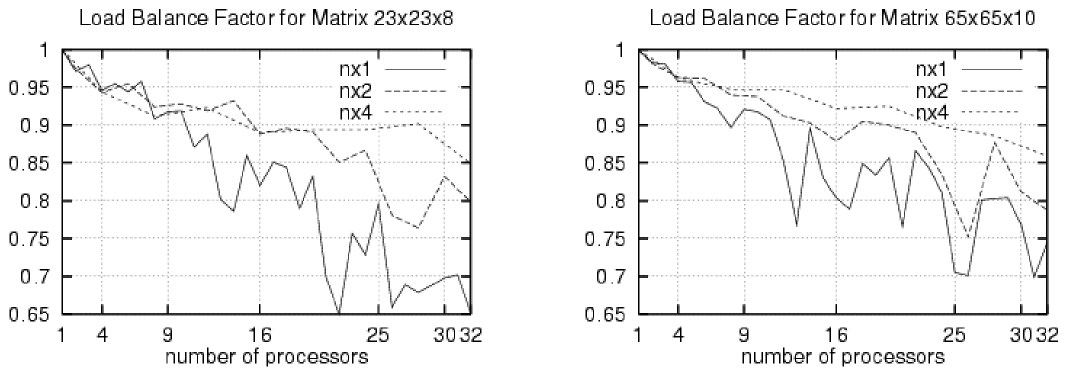

Figure 5.8. Load Balance Factor as a function of a number of processors and processor-grid shape. The following rectangular processor-grid shapes are used: $n \times 1$, $n \times 2$, and $n \times 4$; Left: problem $23 \times 23 \times 8$; Right: problem $65 \times 65 \times 10$i.

Figures 5.9 on Page 83 show the same graphs of the load balance factor as a function of a number of processors and matrix sizes ($23 \times 23 \times 8$, $40 \times 40 \times 8$, and $65 \times 65 \times 10$) displayed for different rectangular shape processor-grids $n \times 1$, $n \times 2$, and $n \times 4$.

Finally, Figures 5.10 on Page 87 demonstrate efficiency of the algorithm as a function of a number of processors and matrix sizes ($23 \times 23 \times 8$, $40 \times 40 \times 8$, and $65 \times 65 \times 10$) for different shape processor-grids, $n \times 1$, $n \times 2$, and $n \times 4$.

Figures 5.8, 5.9, and 5.10 show that

- distribution of the workload degrades when the number of processes increases;
- distribution of the workload degrades monotonically for ”more square” processor-grid $(n \times 4)$, and degrades erratically for rectangular processor-grids $(n \times 2$ and $n \times 1)$;
- distribution of the workload is more even (higher LBF) for ”more square” processor-grid $(n \times 4)$ than for rectangular processor-grids $(n \times 2$ or $n \times 1)$.
- for all three different size problems, efficiency is higher for ”more square” processor-grid $(n \times 4)$ than for rectangular processor-grids $(n \times 2,\ n \times 1)$.

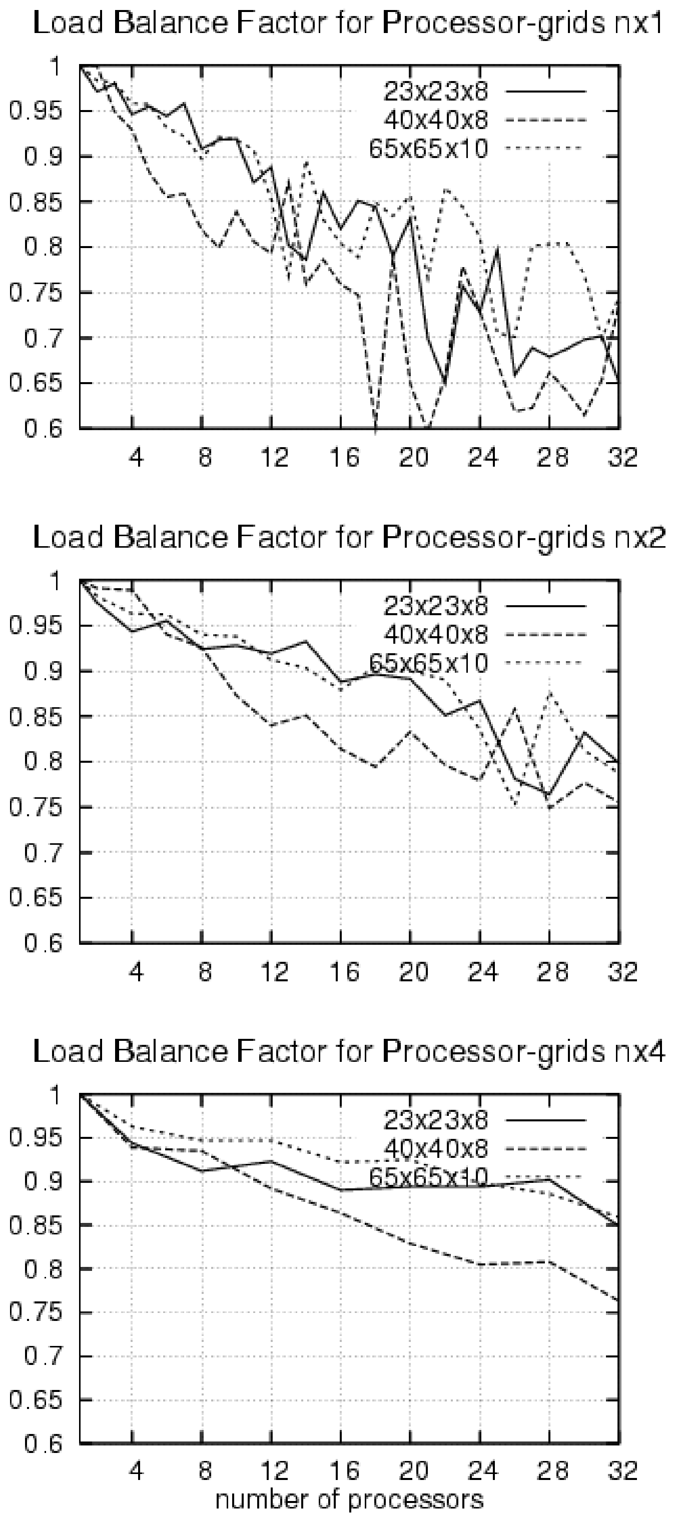

Figure 5.9. Load Balance Factor as a function of a number of processors and matrix sizes ($23 \times 23 \times 8$, $40 \times 40 \times 8$, and $65 \times 65 \times 10$). Top: for processor-grids shape $n \times 1$. Middle: for processor-grid shape $n \times 2$. Bottom: for processor-grid shape $n \times 4$.

Thus, we can conclude that problems are more scalable for square processor-grids than for rectangular processor-grids.

Ice-sheet Matrices used in this work					
Problem	**Order** n	**A sparsity** $\frac{nnz(A)}{n}$	L **and** U **sparsity** $\frac{nnz(L+U-I)}{n}$	**avg. size of a supernode:** $\frac{n}{N}$	**avg.** $\#$ **flops per row:** $\frac{Flops}{n} \times 10^6$
$23 \times 23 \times 8$	16,084	65.37	1,487	8.96	1.79
$33 \times 33 \times 8$	33,304	66.27	1,773	10.21	2.65
$40 \times 40 \times 8$	49,047	66.63	2,195	10.11	4.41
$65 \times 65 \times 10$	163,614	89.00	3,734	9.99	13.04
Benchmark matrices analyzed by Li & Wang (2003)					
	n	$\frac{nnz(A)}{n}$	$\frac{nnz(L+U-I)}{n}$	**supernode:** $\frac{n}{N}$	**row:** $\frac{Flops}{n} \times 10^6$
bbmat	38.744	45/73	932	5.44	0.68
fidapm11	22,294	27.97	1,148	8.54	1.08
wang4	26,064	6.80	411	3.74	0.34
twotone	120,750	10.14	94	3.10	0.06
mixing-tank	29,957	66.60	1,489	11.80	2.57
inv-extrusion-1	30,412	58.99	993	7.37	1.07
ecl32	51,993	7.32	806	2.71	1.17
ir	186,230	15.63	482	5.91	0.36
dds.quadratic	380,698	41.62	854	9.20	1.29
dds15	834,575	15.70	631	5.92	0.72

Table 5.5. Characteristics of ice-sheet benchmark matrices and benchmark matrices from Li & Wang (2003). Theses characteristics are derived from the parameters in Table 5.4 and Table 5.4 and include the average number of nonzero elements of A per row, the average number of nonzero elements of L and U per row, average size of a supernode, and FLOPs (took to factorize the matrix) divided by matrix order.

$\#$ **processors**	**23x23x8**	**33x33x8**	**40x40x8**	**65x65x10**
4	2.86	3.73	3.94	4.00
9	4.46	6.71	7.23	8.22
16	5.60	9.61	10.26	12.99
25	5.97	11.69	12.30	16.38
30	6.10	12.11	13.34	17.73
32	5.95	12.32	13.60	17.78

Table 5.6. Speedup on processor-grids 2×2, 3×3, 4×4, 5×5, 5×6, 4×8.

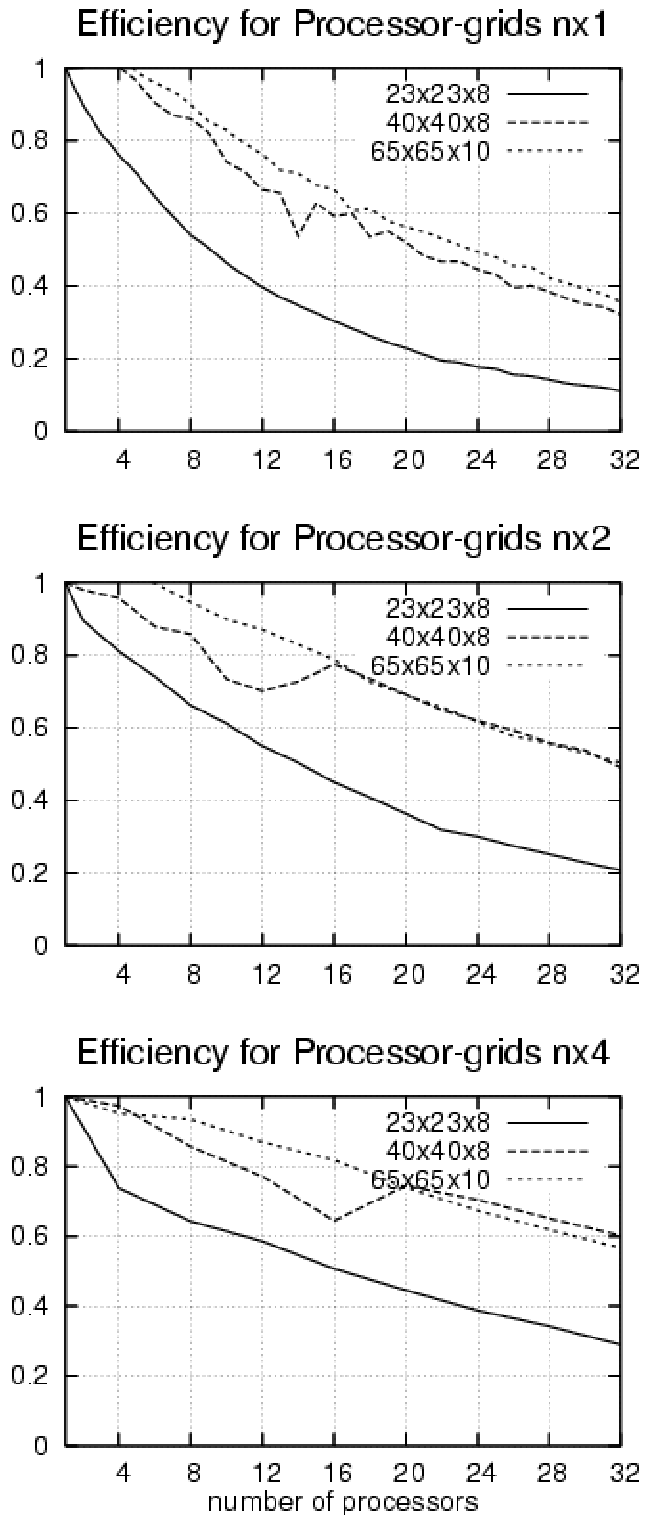

Figure 5.10. Efficiency as a function of a number of processors and matrix sizes ($23 \times 23 \times 8$, $40 \times 40 \times 8$, and $65 \times 65 \times 10$). Top: for processor-grids shape $n \times 1$. Middle: for processor-grid shape $n \times 2$. Bottom: for processor-grid shape $n \times 4$.

5.3.3.3 Column Ordering Strategies

To check which MMD fill-reducing orderings (based on $A^T + A$ or $A^T A$) is better for our type matrices, we tested performance of the algorithm with these orderings on different size problems. Table 5.7 on Page 89 shows characteristics of the algorithm run with two different column-ordering techniques, $MMD(A^T + A)$ and $MMD(A^T A)$. Tests are run with three different size problems.

We can see from the table that $MMD(A^T + A)$ ordering generates fewer numbers of supernodes than $MMD(A^T A)$ ordering for all three different size problems. This means that the average supernode size is larger for $MMD(A^T + A)$ ordering than for $MMD(A^T A)$ ordering. The average size of supernodes generated by $MMD(A^T + A)$ is about $9 - 10$, while the the average size of supernodes generated by $MMD(A^T A)$ is about 6.5. The supernode size determines the size of the matrix passed to matrix-vector multiply and other Level 2 BLAS routines.

More important than average size is the distribution of supernode sizes. Figure 5.11 on Page 89 shows histograms of supernodes distribution generated with these two column-ordering techniques, in red are distribution generated by $MMD(A^T + A)$, and in green are distribution generated by $MMD(A^T A)$. In the figure, the number at the bottom of a bin indicates the smallest supernode size in that bin. The figure shows that $MMD(A^T + A)$, shown in red, generates supernodes distributed over a wider spectrum, that is, relatively smaller number of smaller size supernodes and relatively bigger number of bigger size supernodes than $MMD(A^T A)$ ordering, shown in green. This distribution reduces communication overhead and makes the algorithm faster. Thus, for the type of matrices generated by our 3D higher-order model, column-ordering based on $MMD(A^T + A)$ works better than column-ordering based on $MMD(A^T A)$.

problem	MMD($A^T + A$)				MMD($A^T A$)			
	# supernodes	**avg.size**	**fill-ins**	**run.time**	**# supernodes**	**avg.size**	**fill-ins**	**run.time**
23x23x8	1,795	8.96	13,324	13.71	2,538	6.34	8,529	28.21
40x40x8	4,849	10.11	22,180	73.28	7,598	6.46	12,879	184.15
65x65x10	16,376	9.99	37,304	784.93	24,920	6.57	20,675	1,980.99

Table 5.7. Performance of the algorithm with $MMD(A^T + A)$ and $MMD(A^T A)$ column-ordering methods; *avg.size* = $\frac{problem\ size}{\#supernodes}$; *fill-ins* = $\frac{\#nonzeros(L+U-I)}{\#supernodes}$. Computations are done on 4×4 processor-grid.

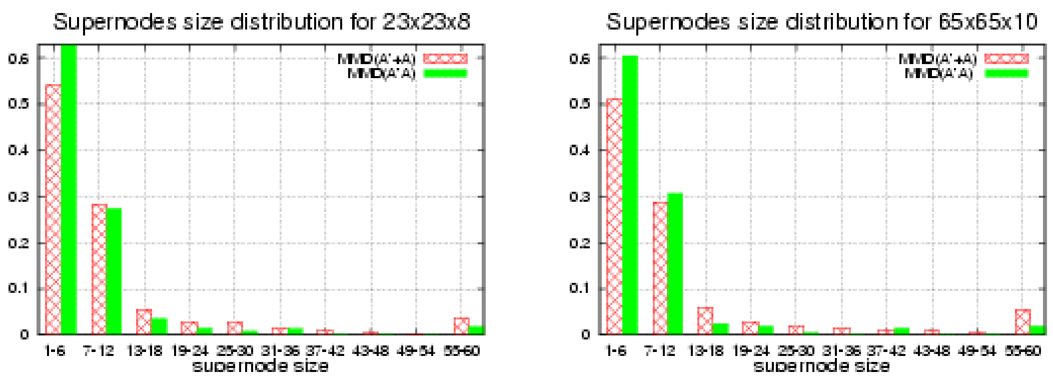

Figure 5.11. Supernodes size distribution for column-ordering techniques $MMD(A^T + A)$ and $MMD(A^T A)$. Left: for problem $23 \times 23 \times 8$. Right: for problem $65 \times 65 \times 10$.

5.3.3.4 Algorithm Stability and Numerical Error

In addition to computing a solution to a system of equations, we must also evaluate the accuracy of the computed solution. Real numbers on a computer are generally represented in single precision or double precision floating point format. Single precision numbers have about 6 decimal digits of precision, while double precision numbers have about 16 decimal digits. These formats are unable to represent real numbers exactly. As computations are performed, we must concern ourselves with round off error and the evolving accuracy. In addition to round off error, there is likely to be uncertainty in the values of A and b that must also be taken into account.

In theory, we should be able to put error bounds on the computations by following the sequence of operations performed by the algorithm used to solve the system of equations. In practice this approach tends to grossly overstate the errors that are actually observed because a portion of the round off error is reduced due to cancellation. Instead, the standard practice is to answer the two following questions [10].

1. Is the computed solution x the exact solution of a nearby problem?
2. If small changes are made to the given problem, are changes to the exact solution also small?

A problem $\bar{A}x = \bar{b}$ is considered nearby $Ax = b$ when small perturbations to A and b produce $\bar{A}$ and $\bar{b}$. When the first question is answered yes, the computational error has been kept under control. An algorithm that satisfies this property is called *stable*. When the algorithm is stable, it is as though we made small perturbations to the problem and solved the perturbed system exactly.

If the answer to the second question is yes, then the problem is called *well-conditioned*. If the problem is well-conditioned and the algorithm is stable, then our calculated solution is a good estimate of the exact solution. If the answer to the second question is no, then the problem is called *ill-conditioned*. If a problem is ill-conditioned, then our solution is likely to have a large error even if the algorithm used to compute it is stable. The condition of a problem is a property of the problem.

Checking if the problem is well-conditioned or not involves calculating all eigenvalues of the matrix, which is at least $O(n^3)$ and is not feasible for big size problems like our problems.

Since there are no useful formulas that indicate the stability of Gaussian elimination and LU factorization in practice, the common approach to ensuring the calculations are stable is to measure the precision of the solution after it has been calculated.

SuperLU-DIST does not calculate the matrix condition number. But it calculates backward error, $BERR$, using Demmel's approach [8] as follows:

$$BERR = max_i \frac{|b_i - \sum_j A_{ij}\bar{x_j}|}{\sum_j |A_{ij}||\bar{x_j}| + |b_i|} \tag{5.4}$$

That is, the calculated solution $(\bar{x})$ can be considered as an exact solution of the perturbed system:

$$(A + H)\bar{x} = b + f, \tag{5.5}$$

where

$$|H_{ij}| \leq BERR \times |A_{ij}|,$$

$$|f_i| \leq BERR \times |b_i|.$$

Knowing that the calculations have been stable and the extent to which the system of equations must be perturbed in order for $\bar{x}$ to be an exact solution does not yet answer how accurately $\bar{x}$ represents the solution of the original problem. If small perturbations of the problem result in large changes to the solution, then $\bar{x}$ may be an inaccurate estimation of the solution. However, we do not have an effective way of estimating the conditionality of the system. So, we limit ourselves to estimating stability of calculations.

Table 5.8 on Page 92 shows that the accumulated errors are quite big even for relatively small size problems (n=33,304) and become about the same size as the elements of A or b for problems of size n=49,047. Solutions calculated with such big errors cannot be trusted.

Grid	**Matrix Order**	**BERR**
$23 \times 23 \times 8$	16,084	$3.03E-16$
$33 \times 33 \times 8$	33,304	$2.33E-03$
$40 \times 40 \times 8$	49,047	$9.91E-01$

Table 5.8. Component-wise relative backward error (BERR) for different size problems.

Is it possible to reduce error? Are they big because our matrix is ill-conditioned or because the algorithm couldn't find the right row/column permutations to maintain stability? Would the error decrease if we permute the rows of the matrix before solving the system using apriori knowledge of the matrix? If so, what criteria should we use to permute the rows?

Figure 5.1 on Page 63 shows that an ice-sheet matrix has zero elements on some of the diagonals. The rows with zero elements on diagonals correspond to equations approximating the ice incompressibility equations. Will the error reduce if we permute the rows of the matrix to make a matrix with non-zero elements on the diagonals? To do so, we need some rule of what rows to exchange.

To demonstrate the concept, we consider a small 2-dimensional problem on a 2×2 grid. The grid consists of 4 elements and 9 nodes as shown in Figure 5.12 on Page 93.

Variables we want to solve for are horizontal and vertical components of velocities defined in the nodes of the grid and pressures defined in the centers of the elements. This problem generates a SLE of size 22. Scatter plot of non-zero entries in the matrix are shown on Figure 5.13 on Page 94.

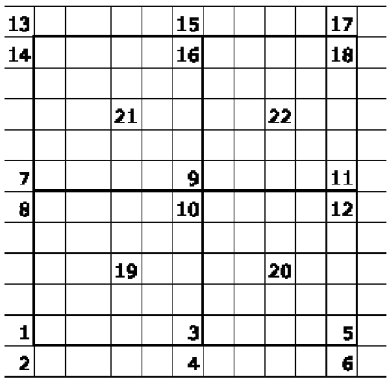

Figure 5.12. 2D problem grid 2×2 (1-18 – velocities, x- and y- components, 19-22 – pressure).

To make matrix diagonal elements nonzero, we exchanged rows in the matrix corresponding to pressure in grid element with the rows corresponding to x-component of momentum equation in the same element, or we exchanged the following rows: $1 \leftrightarrow 19$, $3 \leftrightarrow 20$, $7 \leftrightarrow 21$, and $9 \leftrightarrow 22$.

This will change the global matrix from the one on Figure 5.13 on Page 94 to the one on Figure 5.14 on Page 95. The diagonal elements of this matrix are not zeros. This permutation will not make the matrix a diagonally dominant one, to do so, we have to use upwinding functions in approximating the ice incompressibility equations.

To test if the rows permutations improved stability, we run two problems with matrices of sizes 33,304 and 163,614. The results of calculations without row permutations

$$
\begin{array}{c|cccccccccccccccccccccc|c|c|c}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & & & \\
\hline
1 & x&x&x&x&&&x&x&x&x&&&&&&&&&x&&& & u_1 & & f_1 \\
2 & x&x&x&x&&&x&x&x&x&&&&&&&&&x&&& & v_1 & & f_2 \\
3 & x&x&x&x&x&x&x&x&x&x&x&x&&&&&&&x&x&& & u_2 & & f_3 \\
4 & x&x&x&x&x&x&x&x&x&x&x&x&&&&&&&x&x&& & v_2 & & f_4 \\
5 & &&x&x&x&x&&&x&x&x&x&&&&&&&&x&& & u_3 & & f_5 \\
6 & &&x&&x&x&&&x&x&x&x&&&&&&&&x&& & v_3 & & f_6 \\
7 & x&x&x&x&&&x&x&&x&&&x&x&x&x&&&x&&x& & u_4 & & f_7 \\
8 & x&x&x&x&&&x&x&x&x&&&x&x&x&x&&&x&&x& & v_4 & & f_8 \\
9 & x&x&x&x&x&x&x&x&x&x&x&x&x&x&x&x&x&x&x&x&x&x & u_5 & & f_9 \\
10 & x&x&x&x&x&x&x&x&x&x&x&x&x&x&x&x&x&x&x&x&x&x & v_5 & & f_{10} \\
11 & &&x&x&x&x&&&x&x&x&x&&&x&x&x&x&&x&&x & u_6 & & f_{11} \\
12 & &&x&x&x&x&&&x&x&x&x&&&x&x&x&x&&x&&x & v_6 & = & f_{12} \\
13 & &&&&&&x&x&x&x&&&x&x&x&x&&&&&x& & u_7 & & f_{13} \\
14 & &&&&&&x&x&x&x&&&x&x&x&x&&&&&x& & v_7 & & f_{14} \\
15 & &&&&&&x&x&x&x&x&x&x&x&x&x&x&x&&&x&x & u_8 & & f_{15} \\
16 & &&&&&&x&x&x&x&x&x&x&x&x&x&x&x&&&x&x & v_8 & & f_{16} \\
17 & &&&&&&&&x&x&x&x&&&x&x&x&x&&&&x & u_9 & & f_{17} \\
18 & &&&&&&&&x&x&x&x&&&x&x&x&x&&&&x & v_9 & & f_{18} \\
19 & x&x&x&x&&&x&x&x&x&&&&&&&&&&&& & p_1 & & f_{19} \\
20 & &&x&x&x&x&&&x&x&x&x&&&&&&&&&& & p_2 & & f_{20} \\
21 & &&&&&&x&x&x&x&&&x&x&x&x&&&&&& & p_3 & & f_{21} \\
22 & &&&&&&&&x&x&x&x&&&x&x&x&x&&&& & p_4 & & f_{22}
\end{array}
$$

Figure 5.13. Matrix of 2×2 2D problem.

and with above described row permutations are shown on Table 5.9 on Page 96. The table shows that the row permutations have

1. increased number of supernodes (N);
2. decreased number of nonzero elements of matrices $L + U - I$ (which saves memory and computation time),
3. made, in average, matrices L and U more sparse ($nnz(L + U - I)/N$ decreases),
4. decreased number of FLOPs,
5. decreased running time significantly, and
6. decreased the error, BERR, by ~ 100 times.

$$
\begin{array}{c|cccccccccccccccccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 \\
\hline
19 & x & x & x & x & & & x & x & x & x & & & & & & & & & & & & \\
2 & x & x & x & x & & & x & x & x & x & & & & & & & & & x & & & \\
20 & & & x & x & x & x & & & x & x & x & x & & & & & & & & & & \\
4 & x & x & x & x & x & x & x & x & x & x & x & x & & & & & & & x & x & & \\
5 & & & x & x & x & x & & & x & x & x & x & & & & & & & & x & & \\
6 & & & x & x & x & x & & & x & x & x & x & & & & & & & & x & & \\
21 & & & & & & & x & x & x & x & & & x & x & x & x & & & & & & \\
8 & x & x & x & x & & & x & x & x & x & & & x & x & x & x & & & x & & x & \\
22 & & & & & & & & & x & x & x & x & & & x & x & x & x & & & & \\
10 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
11 & & & x & x & x & x & & & x & x & x & x & & & x & x & x & x & & x & & x \\
12 & & & x & x & x & x & & & x & x & x & x & & & x & x & x & x & & x & & x \\
13 & & & & & & & x & x & x & x & & & x & x & x & x & & & & & x & \\
14 & & & & & & & x & x & x & x & & & x & x & x & x & & & & & x & \\
15 & & & & & & & x & x & x & x & x & x & x & x & x & x & x & x & & & x & x \\
16 & & & & & & & x & x & x & x & x & x & x & x & x & x & x & x & & & x & x \\
17 & & & & & & & & & x & x & x & x & & & x & x & x & x & & & & x \\
18 & & & & & & & & & x & x & x & x & & & x & x & x & x & & & & x \\
1 & x & x & x & x & & & x & x & x & x & & & & & & & & & x & & & \\
3 & x & x & x & x & x & x & x & x & x & x & x & x & & & & & & & x & x & & \\
7 & x & x & x & x & & & x & x & x & x & & & x & x & x & x & & & x & & x & \\
9 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x \\
\hline
\end{array}
\left|\begin{array}{c}
 \\
p_1 \\ v_1 \\ p_2 \\ v_2 \\ u_3 \\ v_3 \\ p_3 \\ v_4 \\ p_4 \\ v_5 \\ u_6 \\ v_6 \\ u_7 \\ v_7 \\ u_8 \\ v_8 \\ u_9 \\ v_9 \\ u_1 \\ u_2 \\ u_4 \\ u_5
\end{array}\right|
=
\left|\begin{array}{c}
 \\
f_{19} \\ f_2 \\ f_{20} \\ f_4 \\ f_5 \\ f_6 \\ f_{21} \\ f_8 \\ f_{22} \\ f_{10} \\ f_{11} \\ f_{12} \\ f_{13} \\ f_{14} \\ f_{15} \\ f_{16} \\ f_{17} \\ f_{18} \\ f_1 \\ f_3 \\ f_7 \\ f_9
\end{array}\right|
$$

Figure 5.14. Matrix of 2×2 2D problem with permuted rows.

Thus, as can be seen from the table, row permutations reduced backward error (BERR) by about 100 times. It also reduced running time significantly. It also reduced backward error (BERR) by about 100 times.

5.4 Conclusion

SuperLU-DIST package was used to parallelize solving the systems of linear equations generated by the 3-D higher-order model. The following conclusions can be made from the performed experiments:

1. In UMISM code, which consists of two steps, building the system of linear equations using FEM and solving the system of linear equations, it is enough to parallelize

problem	**order**	**r**ows not permuted					
		N	**nnz (L+U-I)**	**nnz(L+U-I)/N** $\times 10^3$	**flops** $\times 10^{10}$	**running time**	**BERR**
33x33x8	33,304	3,262	59,085,689	18.11	8.82	33.34	2.33E-03
65x65x10	163,614	15,804	593,525,103	37.56	210.42	743.89	9.99E-01
problem	**order**	**r**ows are permuted					
33x33x8	33,304	5,805	32,664,317	5.63	3.88	11.78	1.22E-05
65x65x10	163,614	26,585	288,318,180	10.85	76.88	158.36	4.52E-04

Table 5.9. Comparison of performance characteristics for the test matrices and matrices with permuted rows.

solving the SLE step. Experiments show that for problems of size $\approx 10^5$, building the SLE takes $\approx$ 5 percent of total time, while solving SLEs takes $\approx$ 95 percent.

2. $SuperLU - DIST$ is a reasonable software package for applying to UMISM problems when the problem size is not too big. For big size problems, $\sim 5 \times 10^5$, the algorithm produces solutions with high error measures $\sim 10^{-4}$.

3. Sparse matrices generated by FEM for UMISM are scalable.

   - For example, running time of solving system $\sim 1.6 \times 10^5$ equations is reduced from ~ 53 min. on one processor to ~ 6 min. on 9 processors.
   - Scalability is better on problems when square processor-grids are used rather than rectangular ones.
   - For problems of sizes $\leq 10^5$, there is no need to use more than 16 processors.

4. MMD on $A^T + A$ column ordering method generates smaller number of (relatively bigger) supernodes than MMD on $A^T A$ column ordering method, thus, making the algorithm faster.

5. It is possible to decrease running time, memory usage, and improve accuracy by using a priori knowledge of the matrix and permute rows to make diagonal elements of the matrix nonzero.[9]

[9]These row permutations do not make the matrix a diagonally dominant one; to do so, we have to use upwinding functions in approximating the ice incompressibility equations.

Chapter 6

APPLICATION OF MODELS TO GLACIOLOGY PROBLEMS

One way to validate a numerical model is to compare the model output with observations and data received from the field, such as Radio-Echo Sounding (RES) or the Ice, Cloud, and land Elevation Satellite (ICESat) data. To validate the model, we have simulated the iceberg profiles [59] and ice flow over subglacial lakes.

Another way to validate a numerical model is to compare the numerical results with the results produced by the other modelers. So to verify the shelf/stream model, we have simulated the flow of ice shelf confined by a rectangular embayment into which an ice stream discharges [46].

6.1 Simulation of Iceberg Profiles

6.1.1 Previous Research

Using an analytic solution [54], Reeh analyzed deformation of the frontal part of a glacier and the state of stresses using a method analogous with the beam theory [54]. Assuming that the glacier is very thin (no z-dimension) and infinitely wide (no y-dimension), he derived the equation for the deflection curve of the floating glacier (5-th order differential equation on x and t) which he solved using *numerical* integration. His calculations show a downward deformation of the frontal part of a glacier at the stages preceding calving. Curves on Figure 6.1 on Page 99 show the progress in time of the deflections in the frontal part of the glacier.

In 1984, Fastook [20] simulated iceberg profiles using SIA model discretized with finite element method and obtained similar results.

Most of the rift and berg profiles in the Ross area show the head down profile that these models predicted.

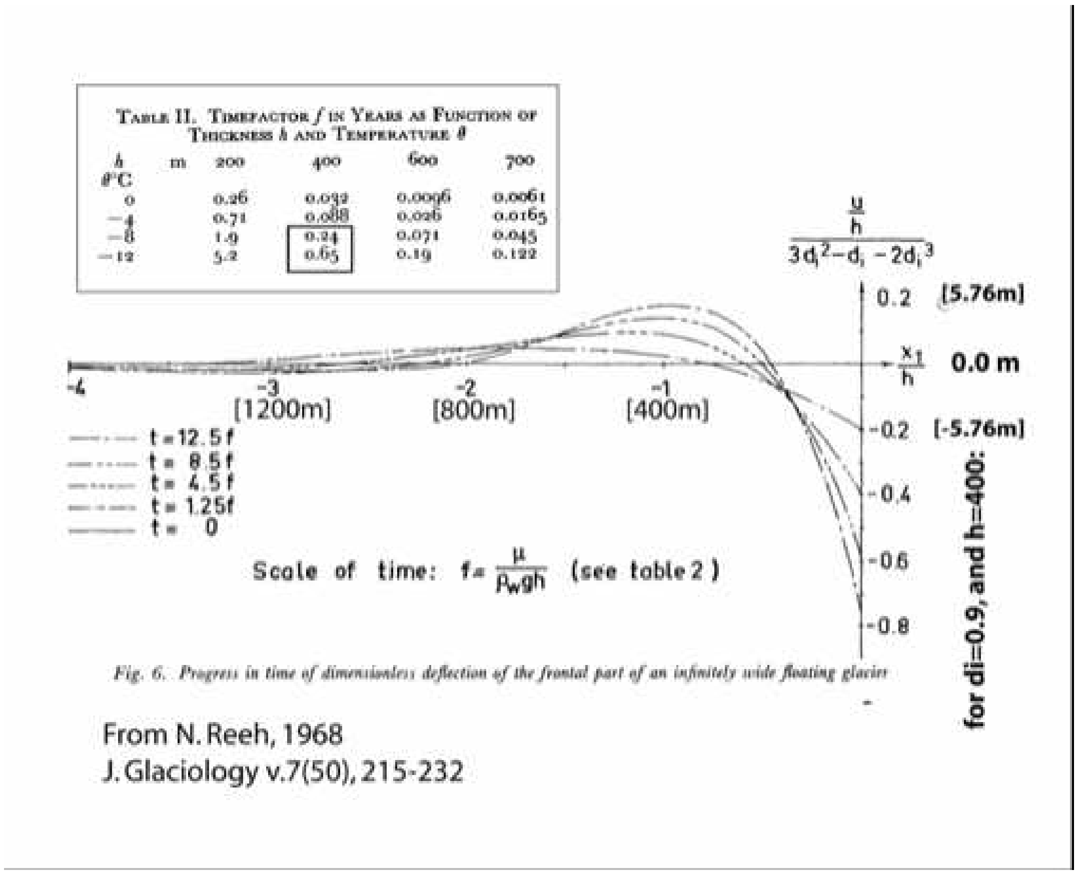

Table II. Timefactor f in Years as Function of Thickness h and Temperature θ

h / θ°C	m	200	400	600	700
0		0.26	0.032	0.0096	0.0061
−4		0.71	0.088	0.026	0.0165
−8		1.9	$\boxed{0.24}$	0.071	0.045
−12		5.2	$\boxed{0.65}$	0.19	0.122

Figure 6.1. Deformation of a frontal part of a glacier.

6.1.2 ICESat Observations

Ted Scambos has examined several changes that occur during iceberg drift using ICESat data. ICESat carries an instrument that gathers elevation data. Specifically, Dr. Scambos has examined data gathered from three large icebergs named A38, A43, and A44.

These icebergs calved in late 1998 and early 2000 from the Ronne Ice Shelf, calved into smaller icebergs, and drifted north along paths shown on Picture 6.2.

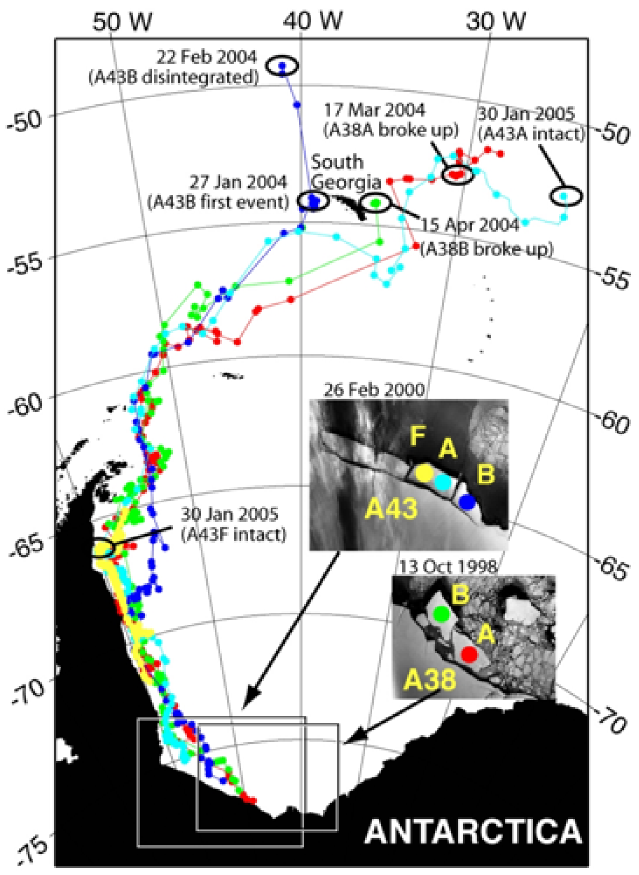

Figure 6.2. Iceberg drift tracks for the five Ronne Ice Shelf-derived icebergs studies. Berg locations are plotted every 10 days. Insets are satellite images of the shelf front soon after the initial calving events. (Provided by Ted Scambos.)

The picture shows that the icebergs have been calved in $70° S$ latitudes (cold waters) and drifted far to the North to the $50° S$ latitudes (to warmer waters).

Figure 6.3 shows elevation profiles from the examined icebergs and the Ronne Ice Shelf front for different dates. The Ronne Ice Shelf and all icebergs within sea ice (south of $63°$), showed berm-type profiles, having 0.6 m raised rounded berms with maximum height at about 2 ice thicknesses from the shelf/berg edge. These profiles are consistent with the ones simulated by Reeh and Fastook shown in Figure 6.1.

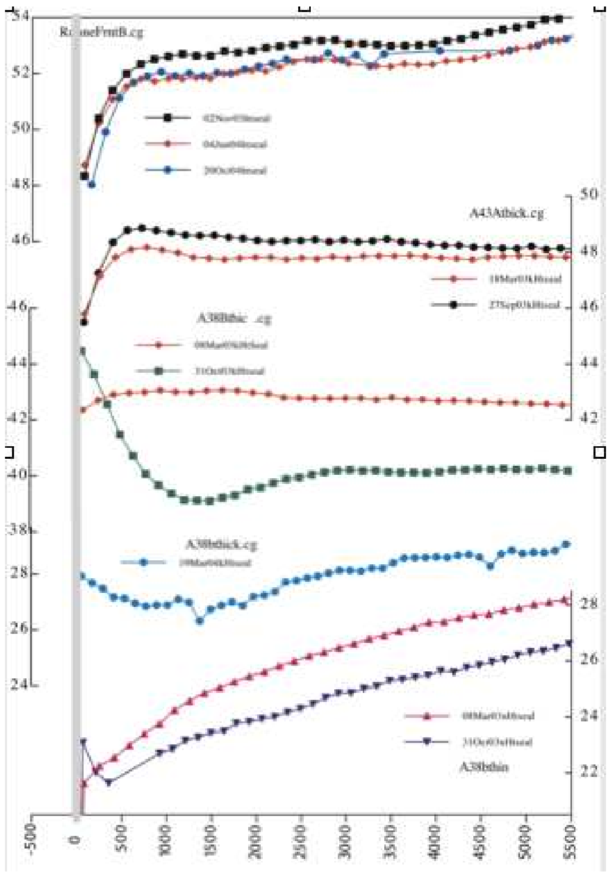

Figure 6.3. Examples of ICESat elevation profiles over iceberg and ice shelf margins. Open symbols indicate the shelf or berg front was in sea ice; solid symbols indicate the iceberg was in open water; gray-fill symbols (A38B, 08 March 2003) indicate partial sea ice cover. (Provided by Ted Scambos)

Icebergs north of the sea ice edge have a consistent pattern of raised edges, 'ramparts', with shallow (50 to 100 cm deep) 'moat' areas inboard and parallel to the margin. These profiles are not consistent with the ones above but are supported by the photographs made by astronauts aboard International Space Station in January 2004. The photograph (Figure 6.4) reveals extensive melt ponds, some impounded by edge-parallel moats. This and other similar photographs initiated Ted Scambos' study.

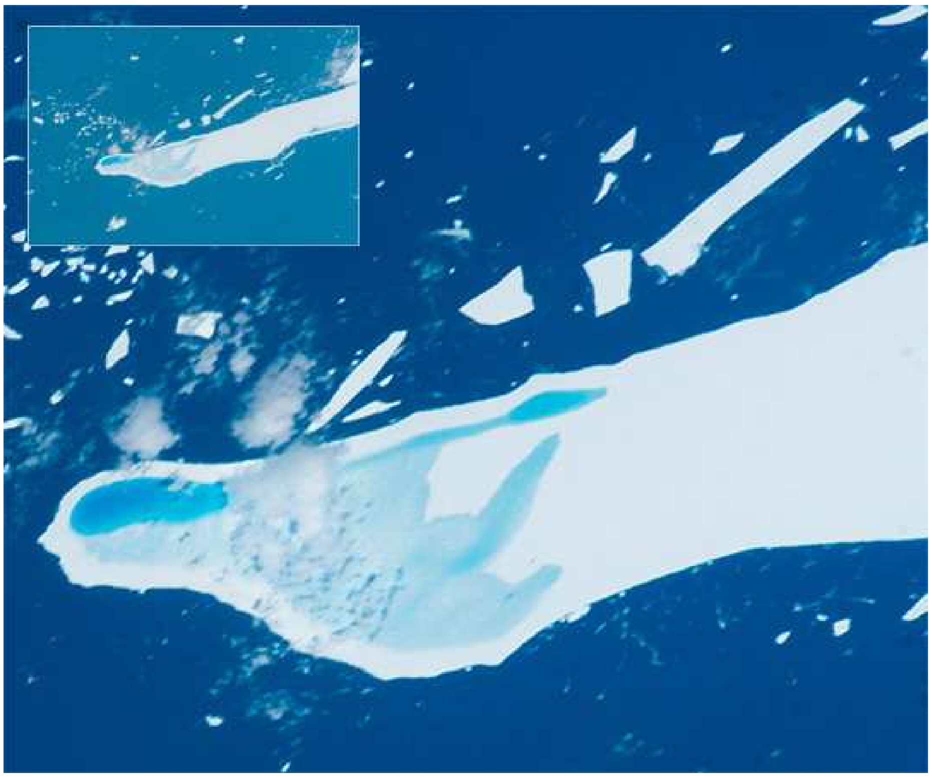

Figure 6.4. Photograph from ISS of iceberg A43B near South Georgia Island showing melt ponding and subsequent break-up. The iceberg calved from the Ronne Ice Shelf of Antarctica. Photograph made on January 22, 2004, shows extensive melt ponds on the berg. Over the next few days to weeks, the berg underwent a rapid disintegration. (Provided by Ted Scambos)

6.1.3 Forces at an Iceberg Face

'Berm' profiles consistent with the ones investigated by Fastook and Reeh could be explained easily if we use Professor Terry Hughes' geometric force balance method. The difference between the lithostatic pressure in ice and the hydrostatic pressure in water is shown by blue arrow. This difference pulls the ice forward leading to toe-down profile.

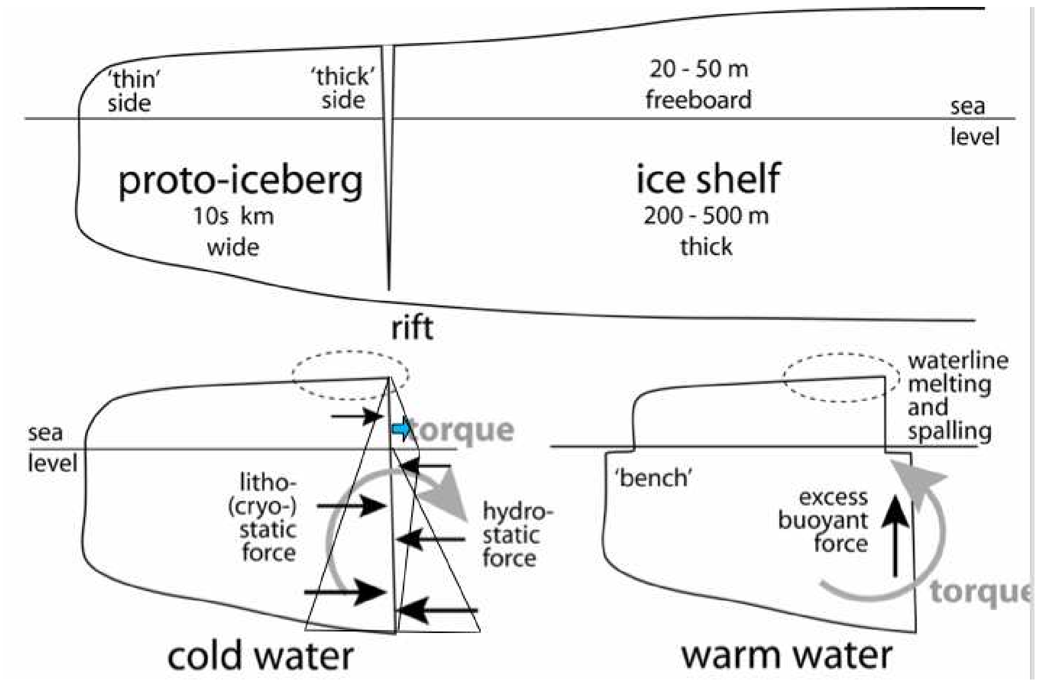

Figure 6.5. Schematic cross-section of ice shelf and icebergs, illustrating basic characteristics, nomenclature, and the physical basis for the model experiments.

However, as the bergs are drifting into ocean that has a thin 'warm' mixed zone, the warm upper layer and waves are acting to erode the front of the iceberg; so that the edge profile becomes like a stair-step: vertical for the deep underwater part, then a shelf of some 10 to 100 meters perhaps, then the freeboard face of the berg. This shelf, or bulge, leads to a reversed torque on the berg front, lifting the freeboard edge upward.

6.1.4 Modeling

We modeled this problem using a 2-D flowline ($x - z$ plane) version of our full-Stoke model. The Figure 6.6 shows two 'cold' profiles and two 'warm' profiles, with our model results next to them.

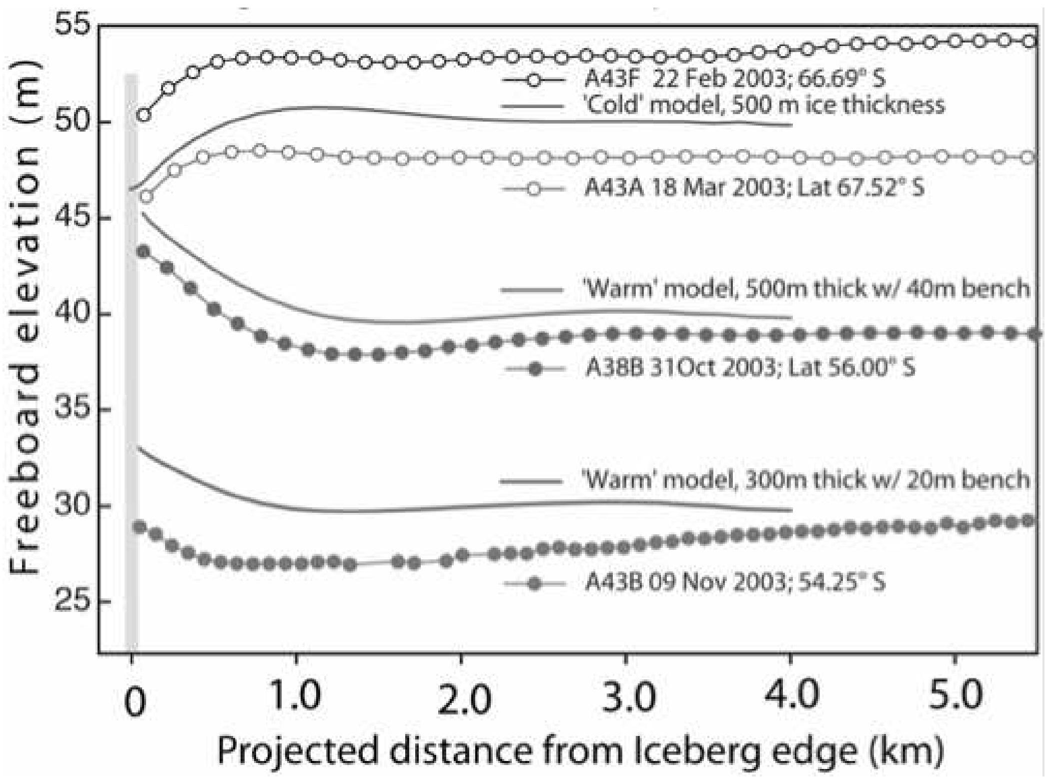

Figure 6.6. Comparison of ICESat observed profiles and our model runs. ICESat profiles used in setting model parameters are shown with ancillary information similar to Figure 6.3

Cold/warm water is simulated as a modified shelf front (and remains modified on the berg edge). The model generates an ’equilibrium state’ profile, i.e. time $\rightarrow$ infinity.

For the ”rampart-moat’ case, we examine the effects of an ice bench of varying widths with an upper surface 5m below water level. Benches of just a few meters width were sufficient to completely eliminate the ’berm’ shape and lift the iceberg margin higher than the mean freeboard. We find that benches of 20 to 40 meters width best match the observed warm-water berg profiles.

As the graphs show, iceberg profiles generated by our model fit quite well the observed iceberg profiles.

6.2 Simulation of Ice Flow over Subglacial Lakes

The next example shows a 2-D and 3-D simulation of ice flow over subglacial lake.

6.2.1 RES Observations

Radio-echo sounding in East Antarctica has revealed the existence of numerous subglacial lakes. Subglacial lakes have relatively flat surface consistent with a surface of an ice shelf. The largest of detected subglacial lakes is Lake Vostok, which is 250 km east of Ridge B (Figure 6.7). It is beneath 4 km of ice, is 250 km long and 50 km wide. Flow of ice across the lake is dominated by the general eastward flow of the grounded ice sheet.

Figure 6.8 shows a Radarsat image of the ice-sheet surface across subglacial Lake Vostok.

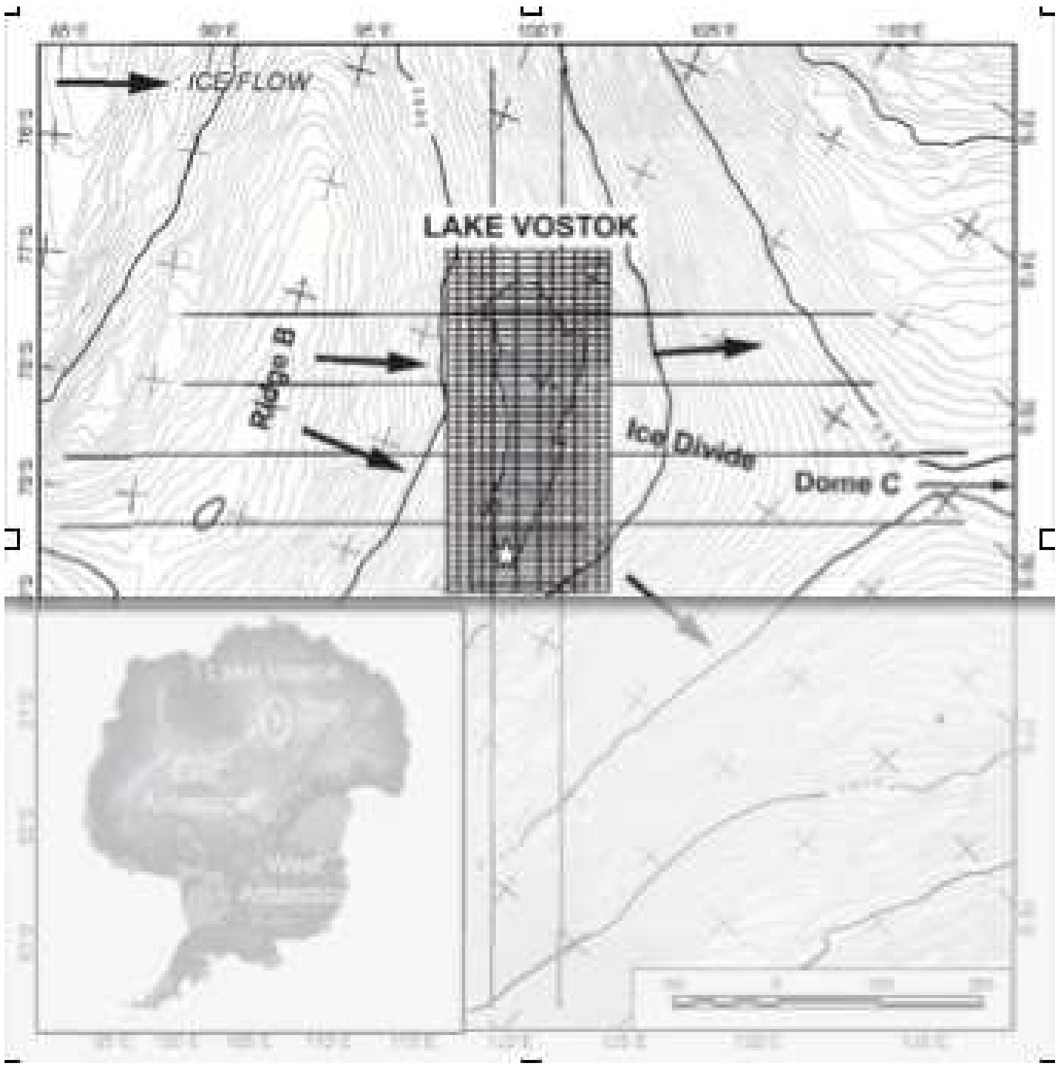

Figure 6.7. Lake Vostok location map from Anahita *et al.*(1902). The white star denotes the location of the Vostok Ice Core. Map scale is in kilometers.

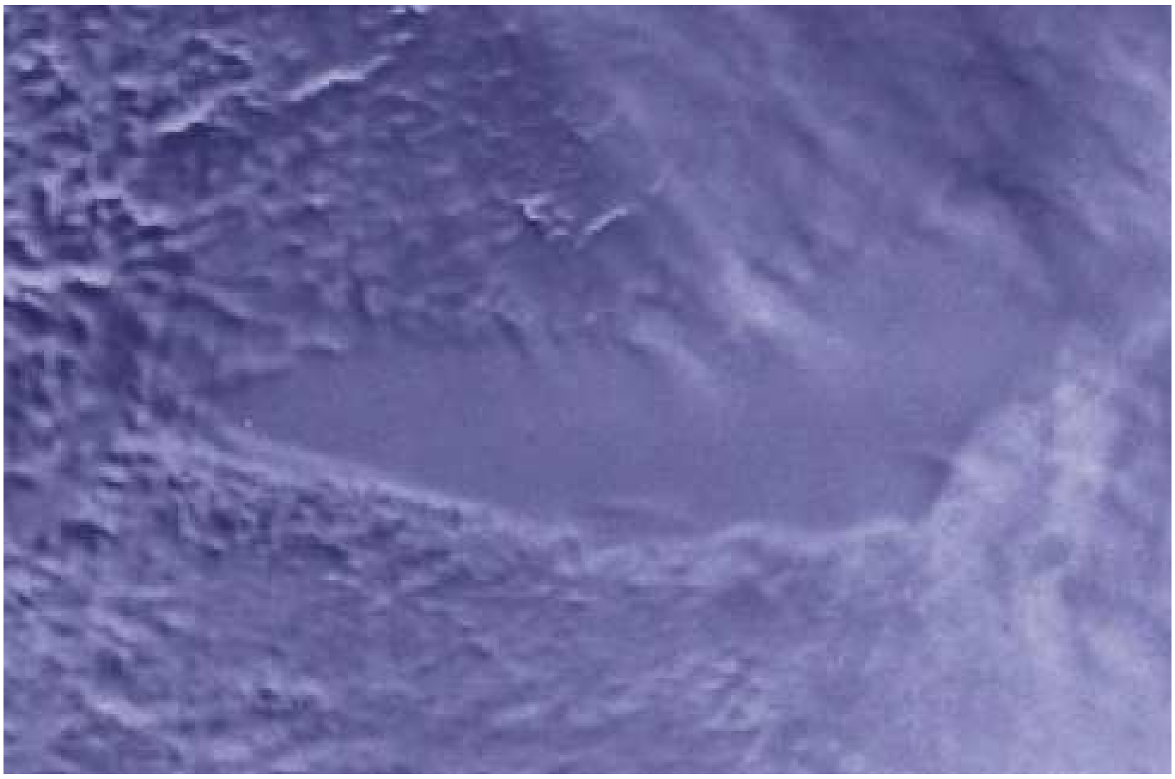

Figure 6.8. Radarsat image of the ice-sheet surface across subglacial Lake Vostok (@ RADARSAT)

Several RES data [42, 61, 4] over East-West transect of Lake Vostok observed one distinct feature of the ice flow over the lake: there is a trench (about 2-5 meters) in ice sheet surface profile in the western margin of the lake and a rise (about 5-10 meters) in the ice sheet surface in the eastern margin of the lake (Figure 6.9).

Figure 6.9 on page 110 is from Science Frontiers no. 107, Sept-Oct, 1996. William R. Corliss. It shows surface and bed elevations on East-West transect of Lake Vostok.

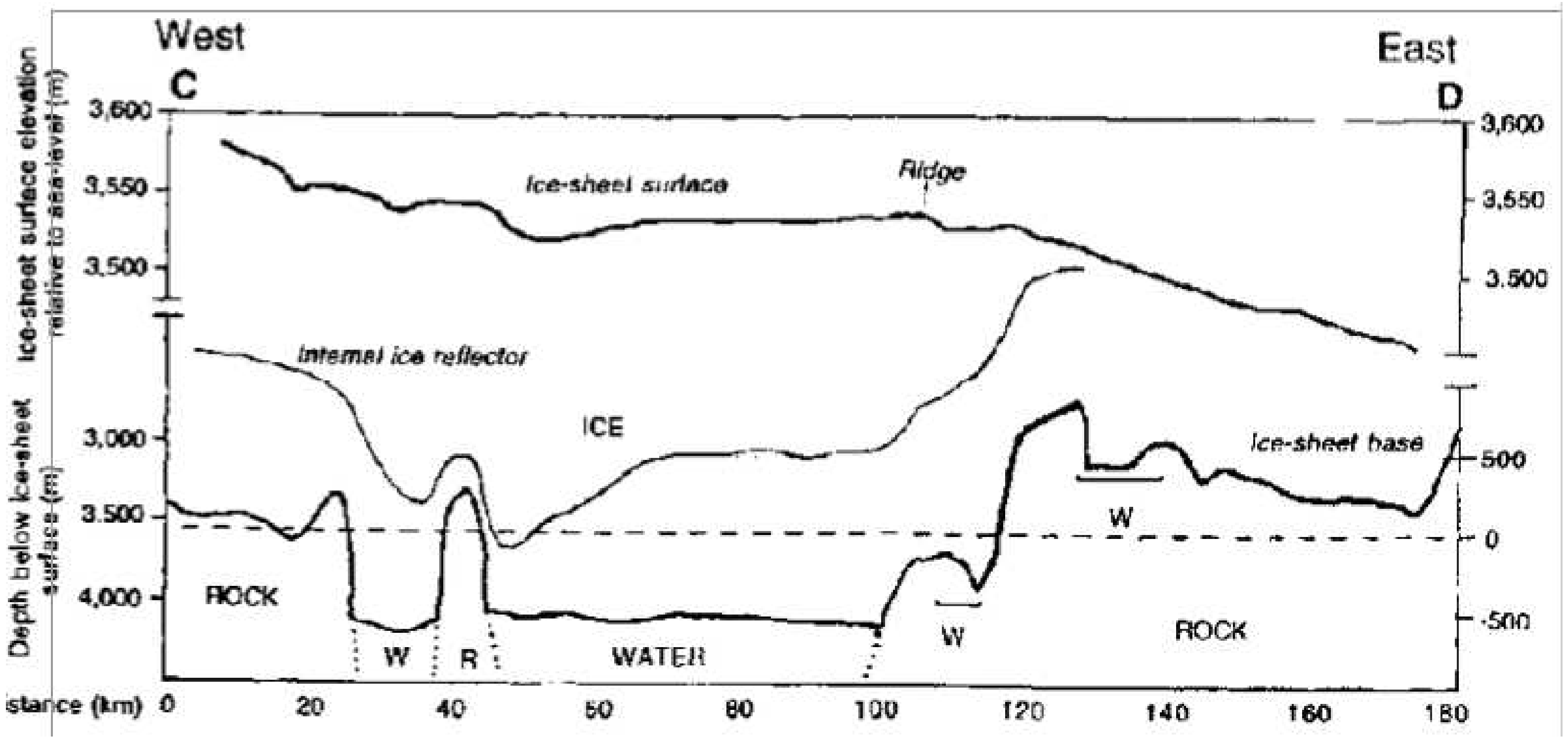
West
East
C
D
Ice-sheet surface elevation relative to sea-level (m)
3,600
3,550
3,500
Ice-sheet surface
Ridge
Internal ice reflector
ICE
Ice-sheet base
Depth below ice-sheet surface (m)
3,000
3,500
4,000
500
0
-500
ROCK
W
R
WATER
W
W
ROCK
Distance (km)
0
20
40
60
80
100
120
140
160
180

Figure 6.9.

6.2.2 Previous Research

Some researches suggest that these profiles are due to downslope and upslope motions produced by a mechanism driven by a change in ice dynamics from grounded, floating, and regrounded ice [60]. Some other researchers think that the trough is caused not by the change in the ice-dynamics from grounded to floated ice but by melting-refreezing mechanism at the ice-water interface [48]. They assume that there is a narrow melting-freezing zone on the eastern part of the lake.

Numerical modeling of ice-flow over subglacial lakes were done by Pattyn [51]. His model shows important aspects of ice-flow over the lake features, such as surface flattening, but has not shown troughs and rises on the sides of the lake due low grid resolution used (the model lake is approximated with only two grid points). Our model is capable of producing all morphological features discussed above; thus supporting the suggestion that the observed profiles are caused by the change in ice dynamics from grounded to floating and regrounded ice.

6.2.3 Modeling

6.2.3.1 2-D Modeling

We solved the problem iteratively starting from a flat frozen bed using a constant accumulation rate. Assumption of a linear flow gives a dome or elliptic analytic profile for the initial data. Starting from this steady state ice sheet conditions, a lake was generated by adjusting the basal boundary conditions for a stress-free surface at the middle of the grid.

The Figure (6.10) shows the evolution of the surface after invoking the stress-free boundary conditions until the steady state was reached.

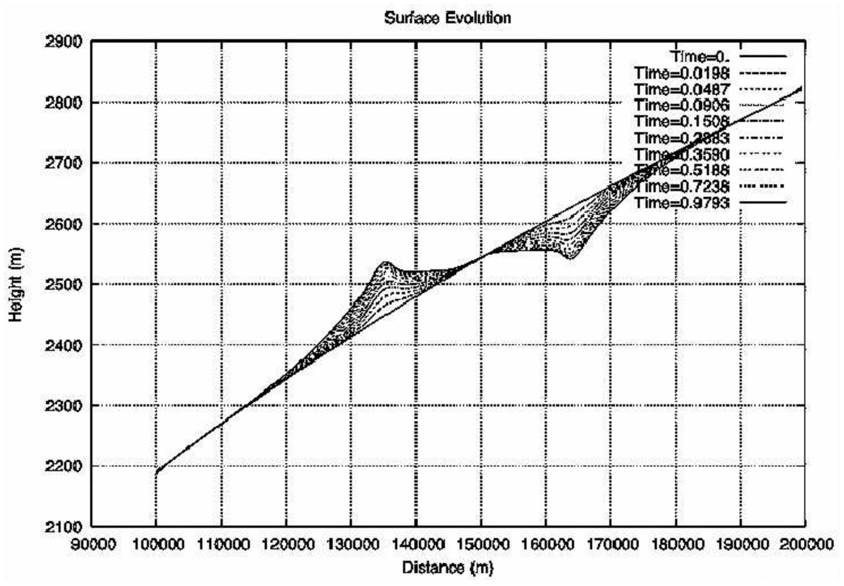

Figure 6.10. Evolution of the ice surface after invoking the stress-free boundary conditions until the steady state is reached. The figure demonstrates the surface flattening over a lake as well as creation of a trench on the onset of the lake and a rise on the down-flow edge of the lake.

As can be seen, the surface topography changes dramatically to become almost flat over the lake. We can also see the trench on the onset of the lake and a rise on the down-flow margin (edge) of the lake, features that have been observed by several radio-echo sounding surveys over West-East flowline above Lake Vostok.

The Figure (6.11) illustrates the velocities magnitude. We can see that velocities are highest over the lake.

6.2.3.2 3-D Modeling

To simulate three-dimensional ice flow over a subglacial lake, we started from the steady state of the radially-symmetric ice sheet with frozen flat bed and elliptic ice surface

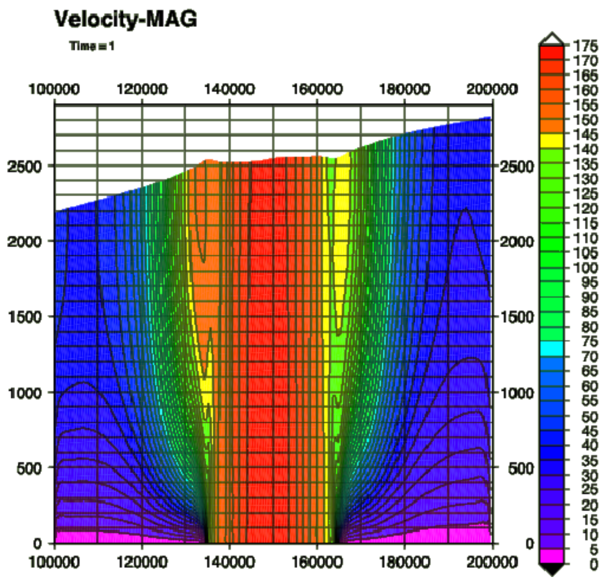

Figure 6.11. Velocity magnitudes. The figure shows a local velocity increase over a lake.

Figure (6.12). Then a subglacial lake was created in the middle of the domain by relaxing velocities constraints.

To solve the problem, we use a three-dimensional, time-dependent model that solves the full momentum (diagnostic) equation and continuity (prognostic) equations to predict the ice thickness distribution and velocity fields in respond to the change in boundary conditions. The model domain is 10x10x4 km which is solved on the 23x23x8 grid (0.5 km grid; time step is 500years). The experiment lasted for 5,000yr.

Figure (6.13) shows the response of the ice thickness to the change in boundary conditions for the first 5,000 years.

The ice velocity rapidly increases and the surface topography changes dramatically to become almost flat over the lake. The model result shows that a trench and a rise are generated at the in-flow and down-flow edges of the lake which are the results of the properly accounting for the longitudinal and transverse shear stresses. Figure (6.14) shows a

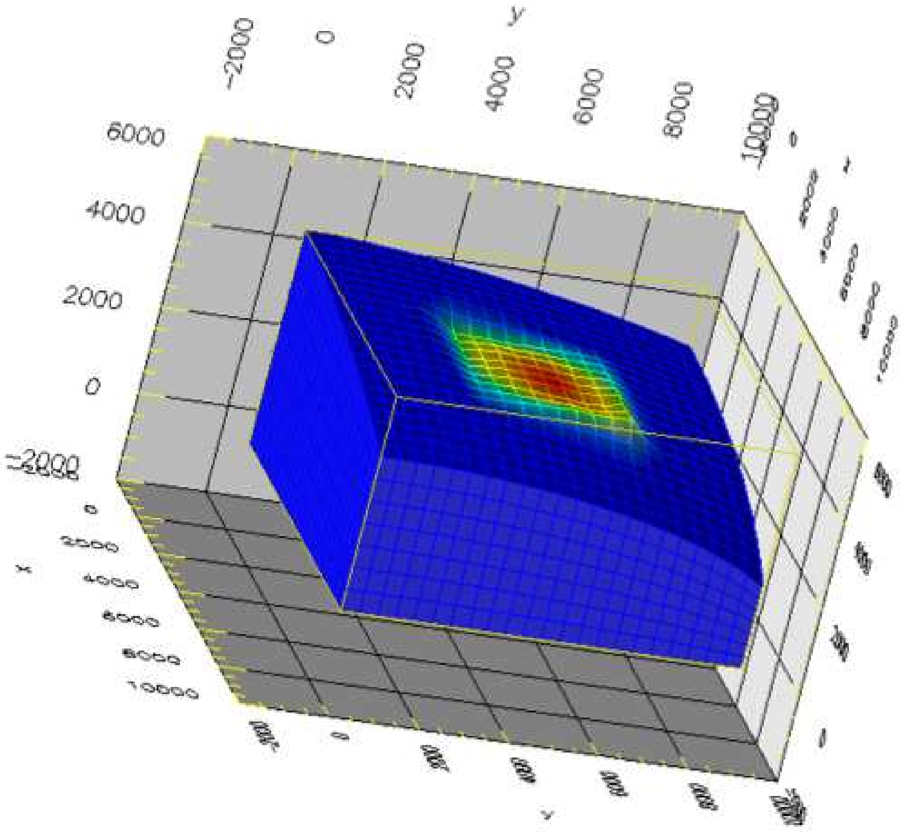

Figure 6.12. The initial steady state of the ice sheet with frozen bed and elliptic ice surface.

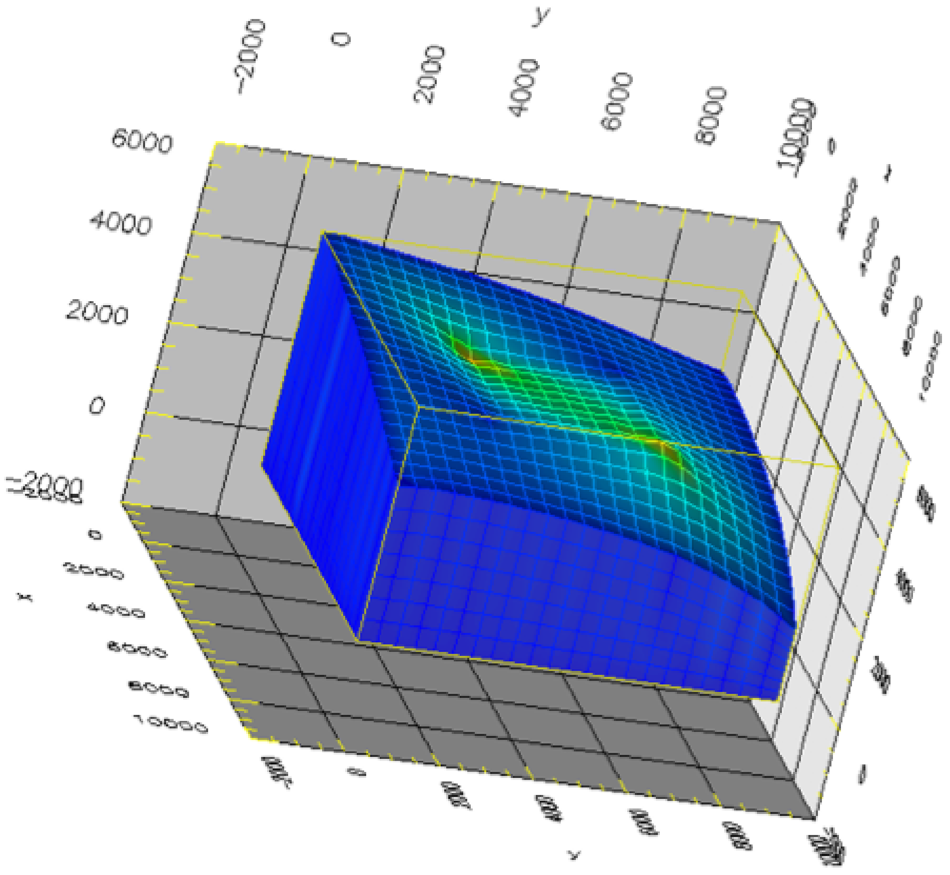

Figure 6.13. The steady-state solution of a subglacial lake simulation.

vertical transect which cuts across the ice sheet over the lake. Thus, our model is able to replicate all the important features of an ice flow over a subglacial lake which are observed at Lake Vostok, Antarctica.

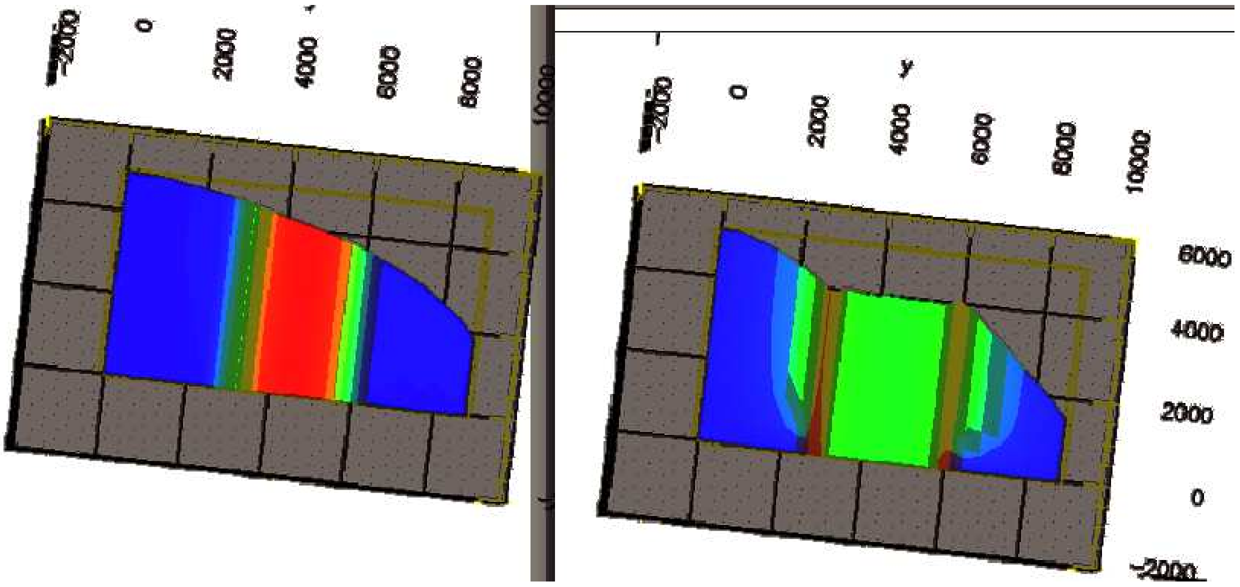

Figure 6.14. A vertical transect through the ice sheet over the lake. Left: initial, prelake, condition. Right: steady-state solution. A trench and a rise are generated at the in-flow and down-flow edges of the lake.

6.3 Simulation of EISMINT Level 1 Ice Shelf Test

To verify shelf/stream model, we have simulated the problem suggested by EISMINT and compared the results with [46]. The test models the flow of an ice-shelf confined by a rectangular (plan view) embayment, into which an ice stream discharges (Figure 6.15). Since the test models an ice-shelf flow, the model's parameters are chosen as $\alpha_x = \alpha_y = 0$.

The two-step numerical procedure is used to perform a time step. The first step is the solution of the diagnostic equations (4.26) - (4.27) to obtain the ice shelf/stream velocity field from the the current ice thickness. This step requires the internal iteration to get

Symbol	Dimensional Value	Constants Definition
n	$= 3$	flow-law exponent
g	$= 9.81\, m\, s^{-2}$	acceleration of gravity
ρ	$= 910\, kg\, m^{-3}$	ice density
ρ_w	$= 1028\, kg\, m^{-3}$	water density
	$31556926\, s\, a^{-1}$	conversion factor for seconds to year
$\dot{a}$	$= 0.3/31556926\, m\, s^{-1}$	ice accumulation rate
B_0	$= 1.4688 \times 10^8\, Pa\, s^{\frac{1}{3}}$	ice stiffness parameter
H_0	$= 1000\, m$	initial ice thickness
u_0	$= \frac{400}{31556926}\, m\, s^{-1}$	velocity of ice-stream input

Table 6.1. Values of constants specified in the intercomparison experiment in Macayeal (1994)].

an accurate solution. The second step is the solution of the prognostic equation (3.28) to specify how ice-thickness h changes with time as a result of the divergence of the ice transport associated with nonzero velocities and the surface and basal accumulation rates.

6.3.1 Initial and Boundary Conditions and Finite Element Mesh

Figure (6.15) displays the finite-element mesh which has been used to simulate the test by [46] and is used in this work. This way we can compare the results of the simulations of shelf flow on the same mesh. The figure also displays the boundary conditions used for the test.

Table 6.3 shows values of constants used for the experiments. They are specified following [46].

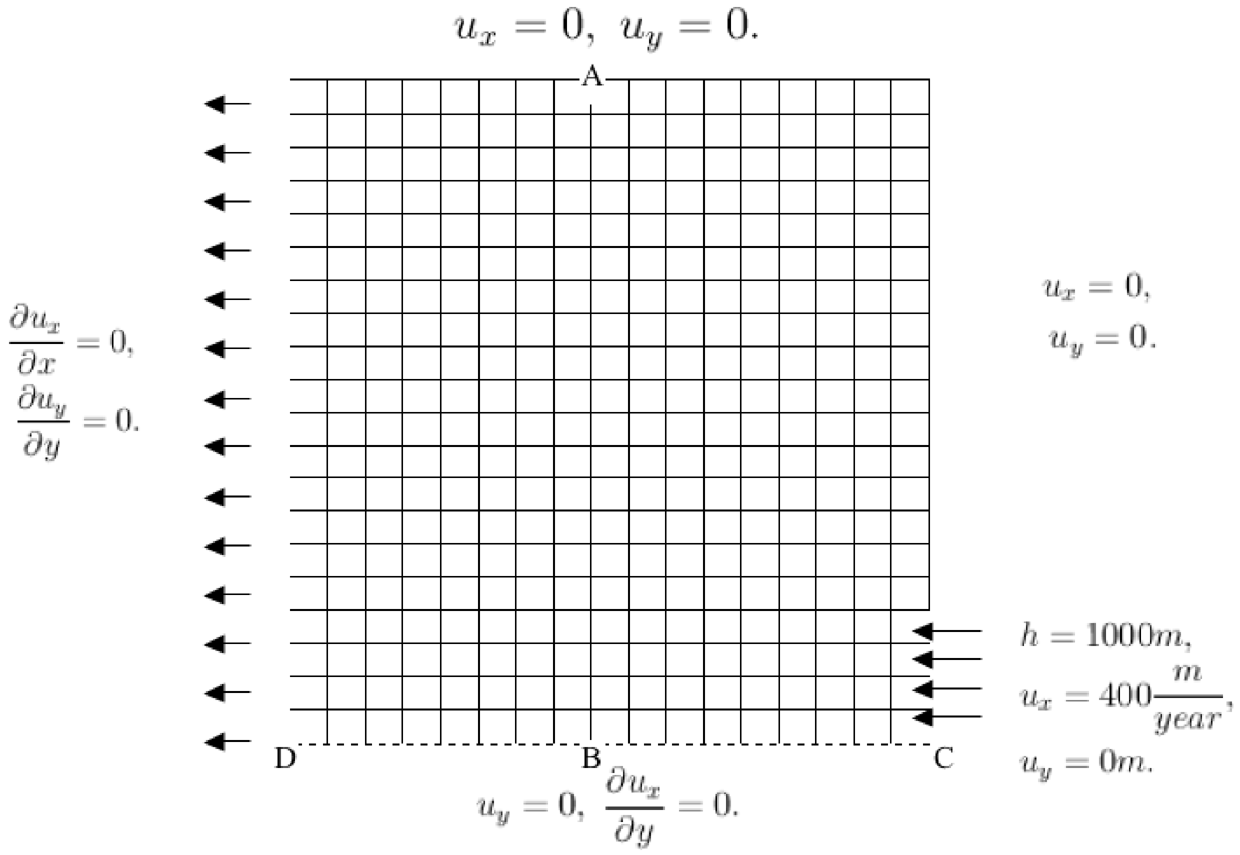
$u_x = 0,\ u_y = 0.$
A
$\frac{\partial u_x}{\partial x} = 0,$
$\frac{\partial u_y}{\partial y} = 0.$
$u_x = 0,$
$u_y = 0.$
$h = 1000m,$
$u_x = 400\frac{m}{year},$
$u_y = 0m.$
D
B
C
$u_y = 0,\ \frac{\partial u_x}{\partial y} = 0.$

Figure 6.15. Finite Element Mesh and Boundary Conditions. Mesh: 17x21 nodes, mesh resolution corresponds to 5 km. Boundary conditions: the kinematic boundary condition associated with ice-stream input is specified on the bottom 4 nodes of the right boundary; The ice front corresponds to the right boundary; The lower boundary, line CD, is an axis of symmetry across which there are no gradients in longitudinal velocity; The top boundary and portion of the right boundary, not corresponding to the inflowing ice stream, have zero velocity (no slip, no normal flow) boundary conditions specified.

6.3.2 Results; Comparison with MacAyeal's EISMINT experiments

We run the coupled ice-shelf model through about 600 years (in dimensional units) starting with a uniform 1000-m ice thickness. The equilibration of the thickness at the ice-front node corresponding to the point D (in Fig. 6.15) where the axes of symmetry (the ice-shelf's longitudinal centerline) intersects the ice front is demonstrated in Figure 6.16. Equilibration is complete at about 400 years.

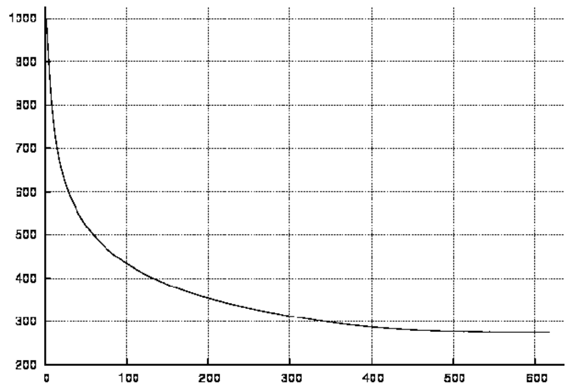

Figure 6.16. Change of ice-front thickness at line of symmetry (point D in Fig. 6.15). Equilibration is complete at about 400 years.

The ice-thickness field, h, and velocity magnitude at the end of the 400-year evolution are displayed as contour maps in Figures 6.17 and 6.18. The figures also display the ice-thickness field and velocity magnitude at the end of 150-years of evolution generated by [46].

Both programs generated very similar results. The minor defect on both maps is the fact that the thickness has grown large at the node point corresponding to the upper right-hand corner where the two stagnant, no-flux boundaries meet.

Figure 6.19 shows velocity vectors after 400 years of evolution.

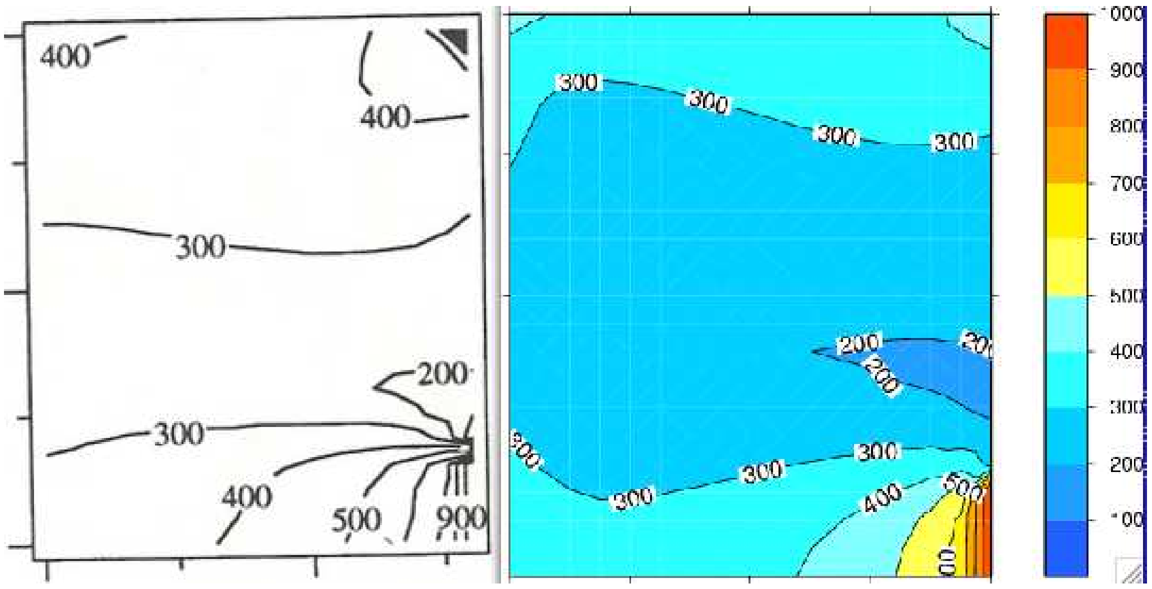

Figure 6.17. Contour map of ice thickness. Left: map generated by Macayeal (1994). Right: map generated by our program. The ice front is on the left-hand side of the diagram; the ice-stream input is on the lower right-hand side.

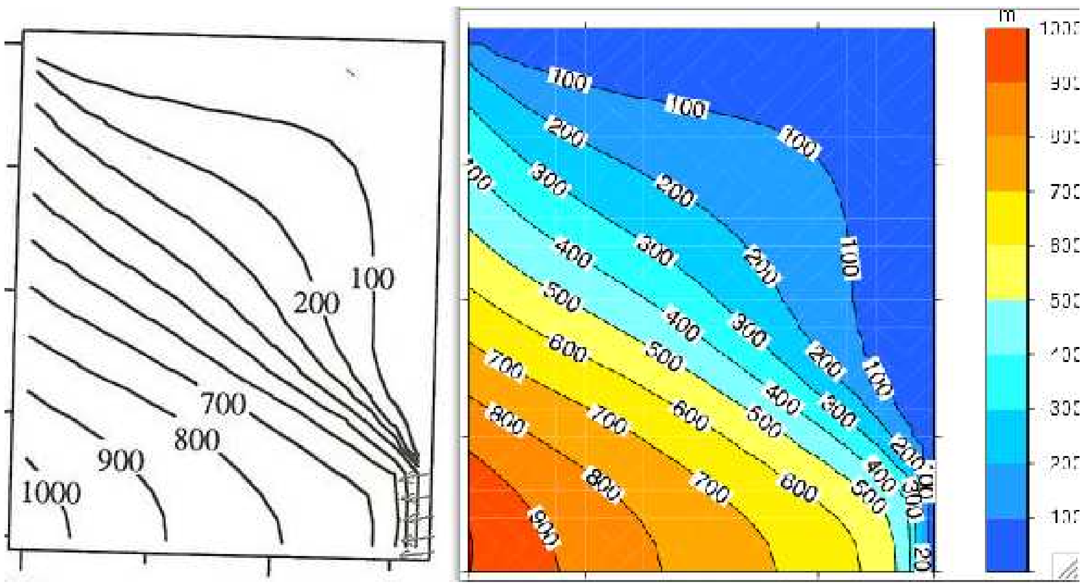

Figure 6.18. Contour map of velocity magnitude. Left: map generated by Macayeal (1994). Right: map generated by our program. The ice front is on the left-hand side of the diagram; the ice-stream input is on the lower right-hand side.

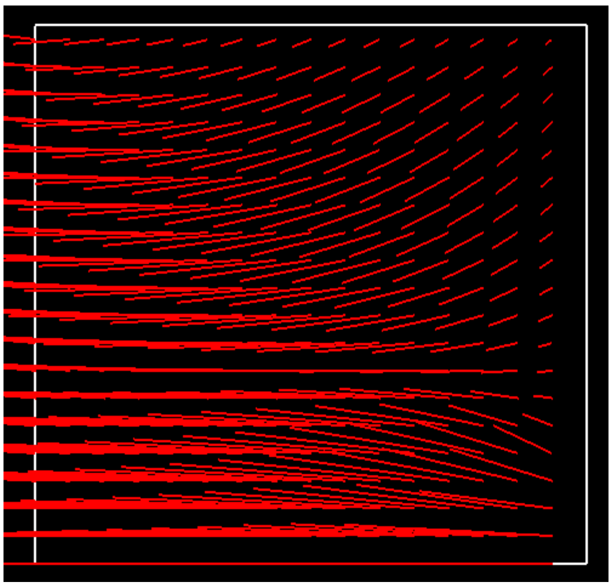

Figure 6.19. Velocity vectors after 400 years.

.

Figures 6.20 and 6.21 show ice thickness along the axis of symmetry (line CD in Figure 6.15 and ice thickness along the transverse, midline axis of the ice shelf (line AB in Figure 6.15 after equilibration has been achieved.

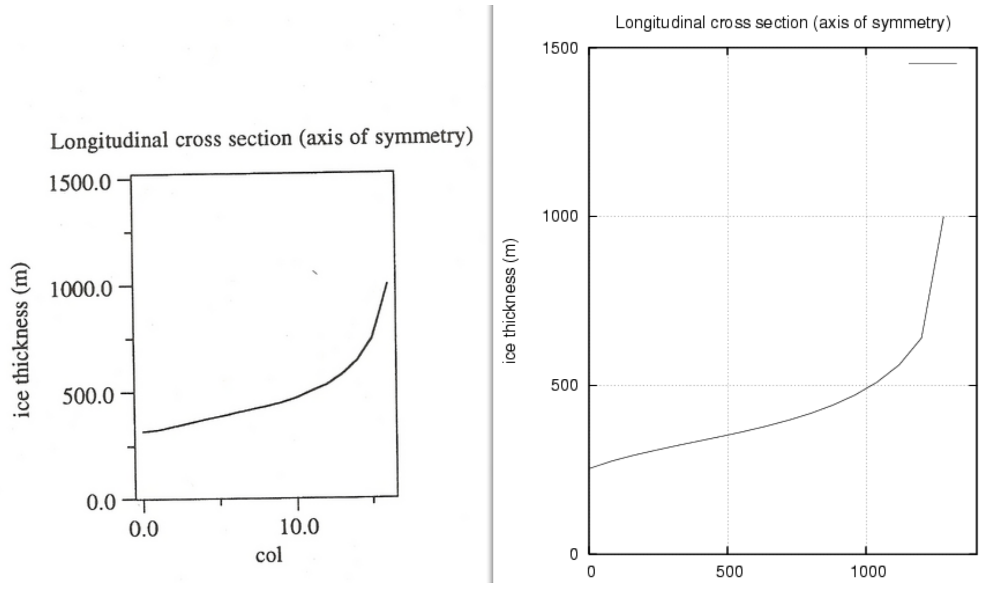

Figure 6.20. Ice thickness (m) along the axis of symmetry. Left: map generated by [46]. Right: map generated by our program.

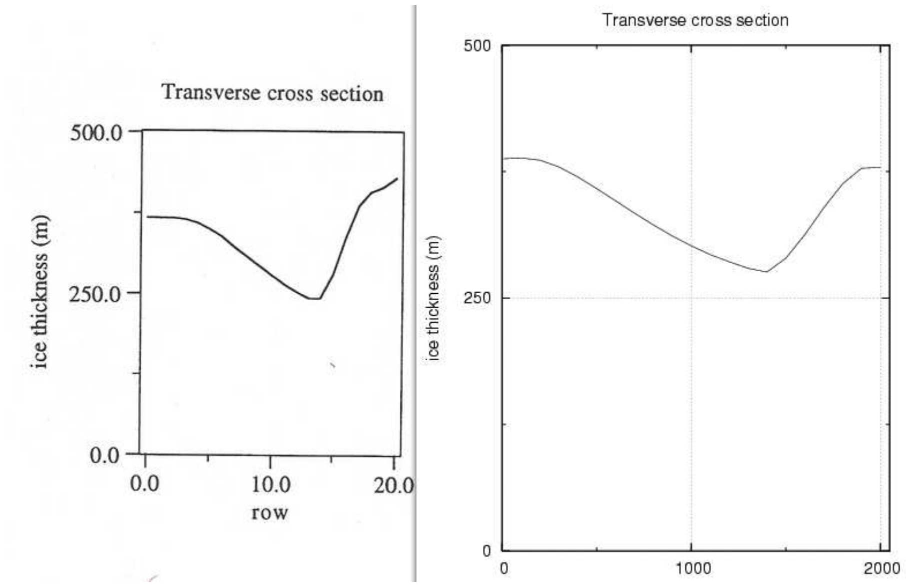

Figure 6.21. Ice thickness (m) along the transverse, midline axis of the ice shelf. Left: map generated by [46] (after 150 years of evolution). Right: map generated by our program (after 400 years of evolution). Notice that the ice thickness at the stagnant side (left) of the ice shelf is slightly higher in our diagram than in the left diagram while the ice thickness in the center of the shelf (right) is slightly lower than in the left diagram. This is the result of depicting the thickness at different time of evolution.

This experiments demonstrates that our ice-shelf model generates results very close to ones generated by [46] which can be considered as a satisfactory verification of the model.

6.4 Modeling Conclusion

This chapter has presented three examples dealing with (1) simulation of iceberg profiles, (2) simulation of ice flow over subglacial lake, and (3) simulation of ice-shelf flow confined by a rectangular embayment into which an ice-stream discharges.

Simulation of ice-shelf flow confined by a rectangular embayment into which an ice-stream discharged demonstrated that the shelf/stream model generates results very similar to ones generated by [46] which can be considered as a satisfactory verification of the model.

The other tests show that the higher-order model replicates important aspects of the iceberg (toe-up and toe-down) profiles, as well as aspects of observed ice stream features, such as the surface flattening over a subglacial lake, a local velocity increase over the lake, and trenches and troughs: features which are observed at Lake Vostok, Antarctica. Thus, they demonstrate the importance of properly accounting for higher-order stress gradients in the model.

Chapter 7

CONCLUSION AND FUTURE WORK

In the first part of this thesis, we presented construction and verification of two models to simulate ice-stream flow, a three-dimensional full-Stokes ice sheet model and a modified MacAyeal-Morland model. Since the 3-D full-Stokes model requires a significant computational effort, in the second part of this thesis, we studied the possibility of using the SuperLU-DIST multiprocessor software package for solving the systems of linear equations generated by the three-dimensional full-Stokes model.

The uniqueness of the modified MacAyeal-Morland equation is in its inclusion of basal shear friction (proportional to the driving stress or to speed) in the derivation of the equation. In the original MacAyeal-Morland equations [46], the basal drag is not included in the fundamental formulation but instead is added as a small correction (proportional to speed) to the final equations. Our approach does two things: first, by including the basal drag in the derivation of the equations, it makes the equations self-consistent. Second, since derived equations contain a term that depends on the bed gradient, our approach gives a formula that accounts for how ice stream flow depends on the bed topography. The basal drag term depends on the heuristic used to approximate the basal shear stress for ice-streams. In this work, we consider two heuristics for the basal shear stress, MacAyeal's assumption that the basal shear stress linearly depends on the velocity, and our assumption that the basal shear stress is some proportion of the driving stress. If we find a dependency formula for the structural form of functional dependency of ice-stream basal shear stress from temperature, velocity, surface elevation, bed-slope, etc. it could be easily substituted in our modified MacAyeal-Morland equations. The search for such formula may be another avenue for future research.

To validate the modified MacAyeal-Morland model, the European Ice Sheet Modeling Initiative 1 intercomparison test is conducted. The test simulates an ice shelf confined by a rectangular embayment into which an ice stream discharges. The results are compared with the results generated by [46]. One of the shortcomings of this chapter is that the constructed model has not been used to simulate the flow of an ice-stream with a non-flat bed; that is, we didn't answer the question: is the effect of basal topography on ice sheet flow small or significant and, if it is significant, can our modified model capture it? This work is left for future research.

The significance of the three-dimensional full-Stokes model is in the inclusion of all higher-order stress gradients in the momentum equation. To validate the three-dimensional higher-order model, experiments demonstrating the importance of the inclusion of all higher order stresses in the model, such as simulation of the evolution of an ice stream within the ice sheet and simulation of iceberg profiles, are conducted. The proper accounting for the higher-order stresses allowed the model to replicate the important features of ice sheet flow observed by glaciologists.

A major deficiency of the higher-order model is that time limitations do not allow using it in large problem domains. One way to solve this problem is to use parallel programming. In this work, the possibility of the application of a distributed SuperLU-DIST software package to solve the model's system of equations is explored, and the performance characteristics of the package are benchmarked. Performed tests indicate that for big-size matrices generated by the three-dimensional model, computations are not stable. However, we have shown that it is possible to improve stability of the algorithm by using a priori knowledge of the matrix and permuting rows prior to applying the algorithm to solving the system.

One of the shortcomings of the work is that the constructed models are isothermal ice-stream flow models – they do not consider effect of temperature flow on ice-stream flow. The models also do not consider the interaction between the ice-stream flow and the underlying bedrock. These steps are left for future work.

In this work we considered possibility of using multiprocessors to overcome the time-constraint problem of solving the 3-D full-Stokes model. Another way to solve this problem may be using embedded modeling. That is, construct a multifaceted embedded model which uses a shallow-ice approximation model (that does not require much computation time) in the entire domain, with the higher-order model in particular subdomains (where longitudinal and lateral stresses play an important role), and the Morland model in the ice-shelf areas. In addition to generating better solutions, this approach would allow us to internalize the generation of boundary conditions at the margins of the higher-order model sub-domains.

Bibliography

[1] ANAHITA, A.T., BELL, R.E., STUDINGER, M. & CLARKE, G. (2002). Ice flow field over Lake Vostok, East Antarctica, inferred by structure tracking.

[2] BAERTSCHY, M. & LI, X.S. (2001). Solution of a three-body problem in quantum mechanics using sparse linear algebra on parallel computers. In *SC2001*, Denver, Colorado USA.

[3] BECKER, E.B., CAREY, G.F. & ODEN, J.T. (1981). *Finite Elements: An Introduction*, vol. 1. Prentice-Hall, Inc.

[4] BELL, R.E., STUDINGER, M., TIKKU, A.A., CLARKE, G.K.C., GUTNERAND, M.M. & MEERTENS, C. (2002). Origin and fate of Lake Vostok frozen to the base of the East Antarctic ice sheet. *Nature*, **416**, 307–310.

[5] BINDSCHADLER, R. (1998). Future of the west antarctic ice sheet. *Science*, **282**, 428–429.

[6] BLATTER, H. (1995). Velocity and stress fields in grounded glaciers: a simple algorithm for including deviatoric stress gradients. *J. Glaciol*, **41**, 333–344.

[7] DEMMEL, J.W., EISENSTAT, S.C., GILBERT, J.R. & ANDJ. W. H. LIU, X.S.L. (1999). A supernodal approach to sparse partial pivoting. *SIAM Journal on Matrix Analysis and Applications*, **20**, 720–755.

[8] DEMMEL, J.W., GILBERT, J.R. & LI, X.S. (2003). SuperLU users' guide. Tech. Rep. LBNL-44289, Computational Research Division, Lawrence Berkley National Laboratory, http://crd.lbl.gov/ xiaoye/SuperLU/.

[9] DONGARRA, J.K., GEIJN, R.A.V.D. & WALKER, D.W. (1994). Scalability issues affecting the design of a dense linear algebra library. *Journal of Parallel and Distributed Computing*, **22**, 523–537.

[10] DUFF, I.S., ERISMAN, A.M. & REID, J.K. (1986). *Direct Methods for Sparse Matrices*. Oxford University Press.

[11] FASTOOK, J. & CHAPMAN, J. (1989). A map plane finite-element model: Three modeling experiments. *Journal of Glaciology*, **35**, 49–52.

[12] FASTOOK, J. & GROSSWALD, M. (1998). Quaternary glaciation of lake baikal and adjacent highlands: modeling experiments. In *International Project on Paleolimnology and Late Cenozoic Climate. No. 11. IPPCCE*, Innsbruck, Austria.

[13] FASTOOK, J. & HOLMLUND, P. (1994). A glaciological model of the younger dryas event in scandinavia. *Journal of Glaciology*, **40**, 125–131.

[14] FASTOOK, J. & HUGHES, T. (1982). *A numerical model for reconstruction and disintegration of the Late Wisconsin glaciation in the Gulf of Maine*, 229–242. Kendall/Hunt Publishing Company, Dubuque, Iowa.

[15] FASTOOK, J. & HUGHES, T. (1982). When ice sheets collapse. *Perspectives in Computing*, **2**, 4–15.

[16] FASTOOK, J. & HUGHES, T. (1988). *A geomorphic method for reconstructing paleo ice sheets part II: Glaciology*, 19–34. Maine Geological Survey, Maine.

[17] FASTOOK, J. & PRENTICE, M. (1994). A finite-element model of antarctica: Sensitivity test for meteorological mass balance relationship. *Journal of Glaciology*, **40**, 167–175.

[18] FASTOOK, J. & SCHMIDT, W. (1982). Finite element analysis of calving from ice fronts. *Annals of Glaciology*, **3**, 103–106.

[19] FASTOOK, J., BRECHER, H. & HUGHES, T. (1995). Derived bedrock elevations, strain rates, and stresses from measured surface elevations and velocities: Jakobshavn glacier, greenland. *Journal of Glaciology*, **41**, 161–173.

[20] FASTOOK, J.L. (1984). *Ice Shelves and Ice Streams: Three Modeling Experiments in Glaciers, Ice Sheets, and Sea Level: Effect of a CO_2- Induced Climatic Change*, 279–300. U.S. Dep. of Energy Rep. DOE/CV/60235-1, Washington D. C.

[21] FASTOOK, J.L. (1987). *The finite element method applied to a time-dependent flow band model*, 321–340. D. Reidel Publishing Company, Dordrecht, Holland, D. Reidel, Boston.

[22] FASTOOK, J.L. (1987). Use of a new finite element continuity model to study the transient behavior of ice stream c and causes of its present low velocity. *Journal of Geophysical Research*, **92**, 8941–8949.

[23] FASTOOK, J.L. (1990). *A map-plane finite-element program for ice sheet reconstruction: A steady-state calibration with Antarctica and a reconstruction of the Laurentide Ice Sheet for 18,000 BP*, 48–80. IBM Scientific and Technical Computing Dept., White Plains, N.Y.

[24] FASTOOK, J.L. (1992). *A map-plane finite-element program for ice sheet reconstruction*, 45–80. The Baldwin Press, The University of Georgia, Athens, Georgia.

[25] FASTOOK, J.L. (1993). The finite-element method for solving conservation equations in glaciology. *Computational Science and Engineering*, **1**, 55–67.

[26] FASTOOK, J.L. (1996). Control methods applied to inverse modeling of ice sheets: The use of finite-element basis functions. In *Third Annual West Antarctic Ice Sheet Initiative Workshop, 25-27 Sept. 1996*, Sterling, Virginia.

[27] FASTOOK, J.L. (1998). Heat generation in the shear margin: A finite-element model. In *Chapman Conference on the West Antarctic Ice Sheet*, University of Maine, ME.

[28] FASTOOK, J.L. & JOHNSON, J.V. (2002). Northern hemisphere glaciation and its sensitivity to basal melt water. In *Quaternary International*.

[29] FASTOOK, J.L. & JOHNSON, J.V. (2002). Predicting the locations of lakes beneath antarctica. *Journal of Glaciology*.

[30] FASTOOK, J.L., HEAD, J.W., MARCHANT, D.R. & SHEAN, D.E. (2005). Ice sheet modeling: Mass balance relationships for map-plane ice sheetreconstruction: Application to tharsis montes glaciation. In *Lunar and Planetary Science XXXVI*, Houston, TX, abstract 1212.

[31] GLEN, J.W. (1955). The creep of polycrystalline ice. In *Proceedings of the Royal Society of London*, 228–519.

[32] HINDMARSH, R.C.A. (2004). A numerical comparison of approximations to the stokes equations used in ice sheet and glacier modeling. *J. Geophys. Res.*, **109**.

[33] HOOKE, R.L. (1981). Flow law for polycrystalline ice in glaciers: Comparison of theoretical predictions, laboratory data, and field measurements. *Rev. Geophys. Space Phys.*, **19**, 664–672.

[34] HOOKE, R.L. (2005). *Principles of Glacier Mechanics*. Cambridge University Press, Cambridge, United Kingdom.

[35] HUGHES, T.J. (2008). *Holistic Ice Sheet Modeling, A First-Order Approach*. University of Maine.

[36] HUGHES, T.J.R. (1987). *The Finite Element Method: Linear static and dynamic finite element analysis*. Prentice Hall, Inc.

[37] HUTTER, K. (1983). *Theoretical Glaciology*. Dordrecht, Kluwer Academic Publishers.

[38] HUYBRECHTS, P., PAYNE, T., ABE-OUCHI, A., CALOV, R., FASTOOK, J., GREVE, R., HINDMARSH, R., HOYDAL, O., JOHANNESSON, T., MACAYEAL, D., MARSIAT, I., RITZ, C., VERBITSKY, M., WADDINGTON, E. & WARNER, R. (1995). The eismint benchmarks for testing ice-sheet models, chamonix, france. igs/eismint international conference on ice sheet modeling. *Annals of Glaciology*, **23**, 1–12.

[39] JACOBS, R. (2005). *Data Structures and Algorithms for Efficient Solution of Simultaneous Linear Equations from 3-D Ice Sheet Models*. Master's thesis, University of Maine, Deptartment of Computer Science.

[40] JACOBS, R., FASTOOK, J. & SARGENT, A. (2006). Applying sparse matrix solvers to a glacial ice sheet model. In *International Conference on Scientific Computing (CSC'06)*, CSC4123, Las Vegas, USA.

[41] JOHNSON, J.V. (2002). *A Basal water model for ice sheets*. Ph.D. thesis, The University of Maine, Orono, Maine.

[42] KAPITSA, A.P., RIDLEY, J.K., DE Q. ROBIN, G., SIEGERT, M.J. & ZOTIKOV, I.A. (1996). A large deep freshwater lake beneath the ice of central East Antarctica. *Nature*, **381**, 684–686.

[43] LI, X.S. (2004). SuperLU: Sparse direct solver. *SIAM Short Course on the CTS Collection*.

[44] LI, X.S. (2005). An overview of SuperLU: Algorithms, implementation, and user interface. *ACM Transactions on Mathematical Software*, **31**, 302–325.

[45] LI, X.S. & WANG, Y. (2003). Performance evaluation and enhancement of SuperLU-DIST 2.0.

[46] MACAYEAL, D.R. (1994). *An Ice-Sheet Modeller's Training Manual: EISMINT Model Intercomparison Exercises*. Dept. of Geophysical Sciences, University of Chicago.

[47] MARTIN, C., NAVARRO, F., OTERO, J., CUADRADO, M. & CORCUERA, M. (2003). Three-dimensional modelling of the dynamics of Johnson Glacier (Livingstone Island, Antarctica). *Ann. Glaciol.*, **39**, 1–8.

[48] MAYER, C. & SIEGERT, M. (2000). Numerical modelling of ice-sheet dynamics across the Vostok subglacial lake, central East Antarctica. *J. Glaciol.*, **46**, 197–205.

[49] MORLAND, L.W. (1987). Unconfined ice shelf flow. In *Dynamics of the West Antarctic Ice Sheet*, 99–116, D. Reidel Publishing Company, proceedings of a Workshop held in Utrecht, May 6-8, 1985.

[50] MORLAND, L.W. & JOHNSON, I.R. (1980). Steady motion of ice sheets. *J. Glaciol.*, **25**, 229–246.

[51] PATTYN, F. (2003). A new three-dimensional higher-order thermomechanical ice sheet model: Basic sensitivity, ice stream development, and ice flow across subglacial lakes. *J. Geophys. Res.*, **108, No. B8**, 2382.

[52] PATTYN, F. (2008). Investigating the stability of subglacial lakes with a full stokes ice-sheet model. *J. Glaciol.*, **54**, 353–361.

[53] PERICHON, F.P.L., ASCHWANDEN, A., BREUER, B., DE SMEDT, B., GAGLIARDINI, O., GUDMUNDSSON, G.H., HINDMARSH, R.C.A., HUBBARD, A., JOHNSON, J.V., KLEINER, T., KONOVALOV, Y., MARTIN, C., PAYNE, A.J., POLLARD, D., PRICE, S., RUCKAMP, M., SAITO, F., SOUCEK, O., SUGIYAMA, S. & ZWINGER, T. (2008). Benchmark experiments for higher-order and full-stokes ice sheet models (ismip-hom). *The Gryosphere*, **2**, 95–108.

[54] REEH, N. (1968). On the calvin of ice from floating glaciers and ice shelves. *J. Glaciol.*, **7**, 215–232.

[55] PRICE, S., WADDINGTON, E. & CONWAY, H. (2007). A full-stress thermomechanical flowband model using the finite volume method. *J. Geophys. Res.*, **F03020, doi:10.1029/2006JF000724**, 112.

[56] SARGENT, A. & FASTOOK, J. (2008). Modified morland-macayeal model for ice-stream flow. In *Fifteenth Annual West Antarctic Ice Sheet Initiative Workshop, Oct. 8 - 10, 2008*, Sterling, Virginia.

[57] SARGENT, A., SCAMBOS, T. & FASTOOK, J. (2005). Better physics in embedded models: iceberg arching and lake-surface profiles. In *Twelfth Annual West Antarctic Ice Sheet Initiative Workshop, Sep. 28 - Oct. 1, 2005*, Sterling, Virginia.

[58] SARGENT, A., FASTOOK, J. & JACOBS, R. (2008). Applying SuperLU-DIST to a glacier ice sheet model. In *International Conference on Scientific Computing (CSC'08)*, CSC3161, 142–148, Las Vegas, USA.

[59] SCAMBOS, T.O., SERGIENKO, O., SARGENT, A., MACAYEAL, D. & FASTOOK, J. (2005). Icesat profiles of tabular iceberg margins and iceberg breakup at low latitudes. *Geophys. Res. Lett.*, **32, L23S09, doi:10.1029/2005GL023802**, 1–4.

[60] SIEGERT, M.J. & RIDLEY, J.K. (1998). An analysis of the ice-sheet surface and subsurface topography above the Vostok Station subglacial lake, central East Antarctica. *J. Geophys. Res.*, **103**, 10195–10207.

[61] TABACCO, I.E., BIANGHI, G., ZIRIZZOTTI, A., ZUCCHERETTI, E., FORIERI, A. & DELLAVEDOVA, A. (2002). Airborne radar survey above Vostok region, east-central Antarctica: ice thickness and Lake Vostok geometry. *J. Glaciol.*, **48**, 62–69.

[62] ZWINGER, T., GREVE, R., GAGLIARDINI, O., SHIRAIWA, T. & LYLY, M. (2007). A full Stokes-flow thermo-mechanical model for firn and ice applied to the Gorshkov crater glacier, Kamchatka. *Ann. Glaciol.*, **45**, 29–37.

Appendix A

SYMBOLS AND CONSTANTS

Symbol	Unit	Definition
x, y	m	horizontal dimensions
z	m	vertical dimension
z_s	m	ice-surface elevation
z_b	m	ice-bed elevation
h	m	ice thickness
u_i	$m\,a^{-1}$	velocity components
U_x, U_y	$m\,a^{-1}$	horizontal components of the depth-averaged velocity of the column of ice
$\dot{a}$	$m\,a^{-1}$	ice accumulation rate
t	yr	time
n	$= 3$	flow-law exponent
A	$Pa^{-n}\,a$	ice-flow parameter
$B = A^{-\frac{1}{n}}$	$Pa\,a^{-\frac{1}{n}}$	resistance to ice-flow, viscosity
P	Pa	hydrostatic pressure
σ_{ij}	Pa	stress components
σ'_{ij}	Pa	deviatoric stress components
σ_e	Pa	effective stress
$\dot{\epsilon_{ij}}$	a^{-1}	strain rate components
δ_{ij}		Kronecker delta
g	$= 9.81\,m\,s^{-2}$	acceleration of gravity
ρ	$= 910\,kg\,m^{-3}$	ice density
ρ_w	$= 1028\,kg\,m^{-3}$	water density
	$31556926\,s\,a^{-1}$	conversion factor (seconds per year)

Table A.1. Symbols and constants used in this work. The following transition is used: $Pa = \frac{N}{m^2} = \frac{kg\,\frac{m}{s^2}}{m^2} = \frac{kg}{m\,s^2}$

Appendix B

SHALLOW ICE APPROXIMATION MODEL

Since solution of a full system of equations is complicated, these equations are solved in a reduced form. A typical ice sheet has a thickness of one or several kilometers while its lateral extend is typically on the order of $1000\ km$. That is, the aspect ratio, or ratio of length to the depth of the ice sheet, is small and this fact can be exploited to derive the reduced form of the ice sheet model. In this model, all stresses are neglected except for the basal shear stress (σ_{xz} and σ_{yz}), that is, gravitational driving stresses are balanced locally by basal traction. This approximation is called a shallow ice approximation (SIA).

Figure B.1 on Page 137 shows the forces acting on a glacier in direction x in the SIA model.

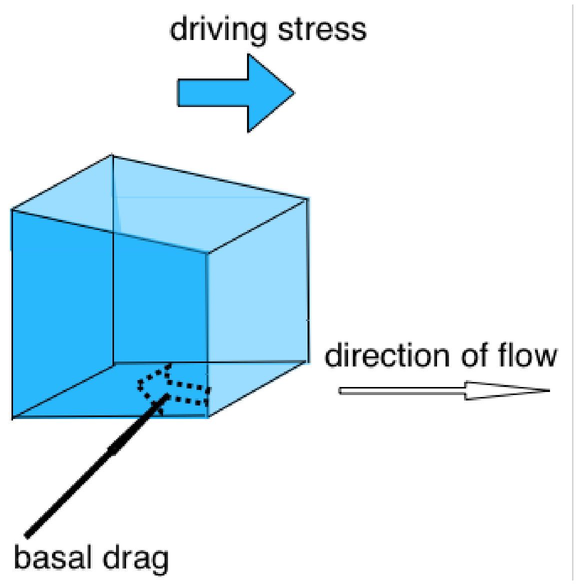

Figure B.1. SIA: major forces acting on ice-sheet in one direction.

The derivation of the model can be found in [[50]] and [[37]]. SIA is a good approximation for regions where creep is the dominant ice flow process.

B Expressions for Velocities

Neglecting all longitudinal deviatoric and lateral shear stresses means that in (2.16) - (2.18), $\frac{\partial \sigma'_{ij}}{\partial x} \to 0, \ \frac{\partial \sigma'_{ij}}{\partial y} \to 0$. In this case, equations (2.16) - (2.18) become

$$\begin{aligned} &\text{x-component:} && \tfrac{\partial \sigma'_{xz}}{\partial z} = -\tfrac{\partial P}{\partial x}, \\ &\text{y-component:} && \tfrac{\partial \sigma'_{yz}}{\partial z} = -\tfrac{\partial P}{\partial y}, \\ &\text{z-component:} && \tfrac{\partial (\sigma'_{zz}+P)}{\partial z} = \rho g. \end{aligned} \tag{B.1}$$

Vertical integration of the $z-$ component of equation (B.1) yields:

$$\sigma'_{zz}(z) + P = \rho g(z - s). \tag{B.2}$$

where s is ice surface.

The disparity between the vertical and horizontal length scales in ice-sheet flow implies that simple shear dominates $\left(\frac{\partial \sigma'_{xz}}{\partial z} > \frac{\partial \sigma'_{zz}}{\partial x}\right)$ and $\left(\frac{\partial \sigma'_{yz}}{\partial z} > \frac{\partial \sigma'_{zz}}{\partial y}\right)$. That is, if we substitute (B.2) into $x-$ and $y-$ components of (B.1) and integrate from $z = z$ to $z = s$, we get that shear stresses balance gravitational driving stresses:

$$\sigma'_{xz}(z) = -\rho g(s - z)\frac{\partial s}{\partial x}, \tag{B.3a}$$

$$\sigma'_{yz}(z) = -\rho g(s - z)\frac{\partial s}{\partial y}. \tag{B.3b}$$

Using ice flow law (2.6) written as $\sigma'_{xz}(z) = B\,(\dot{\epsilon_{xz}})^{\frac{1}{n}}$ and (B.3a), we get

$$B\,(\dot{\epsilon_{xz}})^{\frac{1}{n}} = -\rho g(s-z)\frac{\partial s}{\partial x}. \tag{B.4}$$

Since $\left(\frac{\partial u_x}{\partial z} > \frac{\partial u_z}{\partial x}\right)$, $\dot{\epsilon_{xz}} \approx \frac{1}{2}\frac{\partial u_x}{\partial z}$ and equation (B.4) becomes as follows:

$$\left(\frac{1}{2}\frac{\partial u_x}{\partial z}\right)^{\frac{1}{n}} = -\rho g\, A^{\frac{1}{n}}(s-z)\frac{\partial s}{\partial x}, \tag{B.5}$$

where A is ice flow law rate constant in $Pa^{-3}\,sec^{-1}$ units and $B = A^{-\frac{1}{n}}$.

Integrating (B.5) for u_x (and similar equation for u_y) from $z = z$ to ice surface s generates expressions for horizontal velocities:

$$u_x(z) = u_x(s) - 2(\rho g)^n |\nabla s|^{n-1}\frac{\partial s}{\partial x}\int_s^z A(T)(s-z)^n dz, \tag{B.6a}$$

$$u_y(z) = u_y(s) - 2(\rho g)^n |\nabla s|^{n-1}\frac{\partial s}{\partial y}\int_s^z A(T)(s-z)^n dz. \tag{B.6b}$$

The vertical velocity u_z is found using the divergence of horizontal velocity field from incompressibility condition (2.5a):

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} = 0. \tag{B.7}$$

B Continuity Equation

Conservation of mass equation (2.12) takes the form of the column-averaged flow law that allows to obtain the expression for the change of local ice thickness in space, h:

$$\frac{\partial h}{\partial t} = -\nabla \cdot \left(\vec{\mathbf{U}} h\right) + \dot{a}, \tag{B.8}$$

where $\vec{\mathbf{U}}$ is the vertically averaged horizontal velocity vector, $\nabla \cdot$ is the two-dimensional horizontal divergence operator.

Equations (B.6a) and (B.6b) are integrated from bedrock to ice surface and divided by h to obtain components of the vertically averaged horizontal velocity vector $\vec{\mathbf{U}}$:

$$U_x = \frac{1}{h}\int_b^s u_x(z)dz = u_x(s) + \frac{1}{h}\int_b^s 2\left(\rho g|\nabla s|\right)^n = u_x(s) + \frac{2\left(\rho g|\nabla s|\right)^n A(T)h^{n+1}}{(n+1)(n+2)}, \tag{B.9a}$$

$$U_y = \frac{1}{h}\int_b^s u_y(z)dz = u_y(s) + \frac{1}{h}\int_b^s 2\left(\rho g|\nabla s|\right)^n = u_y(s) + \frac{2\left(\rho g|\nabla s|\right)^n A(T)h^{n+1}}{(n+1)(n+2)} \tag{B.9b}$$

When (B.9a-B.9b) are substituted into equation (B.8), a non-linear parabolic equation results. This equation for h is usually called *”ice-sheet equation”*.

As can be seen from ice-sheet equation, ice-sheet mass flux term in (B.8) is purely a function of the local ice thickness, h, and surface gradient, ∇s.

B Complexity of SIA Equations

In the SIA model the only stress considered is the basal stress. This allows us to reduce 3-dimensional equations to quasi-2-dimensional equations with variables in z directions integrated out, that is, to solve shallow ice approximation model, we have to solve one non-linear parabolic equation (B.8) for h. The velocity field is calculated using formulas (B.6a) and (B.6b). If we are solving a time-dependent problem, the above equations are solved at each time step. The equation for h has one degree of freedom per node and solved in a 2-dimensional grid.

Since SIA equations are solved in a 2-dimensional grid, for a rectangular region that is $50 \times 40 = 2,000$ nodes, the system has only $2,000$ independent variables (ice thickness h).

Since SIA neglects all stresses except the basal drag, it may be a good approximation for inland ice but may be very poor for fast-flowing, low-surface slope ice streams, where longitudinal stresses may not only be important, but may in fact be the dominant stress.

Appendix C

COMPRESSED COLUMN FORMAT SCHEME

In addition to getting an acceptable solution, the goal of solving the system of sparse equations on a computer is minimizing the storage used and minimizing the execution time. Jacobs solved the problem by designing and writing efficient procedures and structures needed to interface the ice sheet model with SuperLU.

To take advantage of the sparsity of the equations, compressed-column format of storing the matrix A of the equations have been used. It is a data format for A that is compatible with SuperLU. Three one-dimensional arrays are used to store a matrix in compressed-column format. One array is used to store the non-zero entries of A. The non-zero entries are stored in column-row order. The second array is used to store the row number of each corresponding non-zero entry in the values array. The matrix column number is the index to the third array. The third array contains the index values of the first and second arrays where the first non-zero entry from each column of A is stored.

However, inserting a new non-zero entry in a compressed-column format data structure is costly. In order to maintain entries in column-row order, all entries in the row number and value arrays above the new entry must be moved and all entries in the column index array above the column number of the new entry must be updated. To alleviate this performance bottleneck, a modified compressed-column format with supporting routines was developed by Rodney Jacobs.The rectangular FEM grid allows us to compute the maximum number of non-zero entries per row of A. This maximum amount of space is allocated for each column in the row number and value arrays. Instead of having a single column pointer array that points to the first entry for each column in the row number and value arrays, two column pointer arrays are used. The first array points to the first row number and value in each column and the second array points to the last row number and value in each column. The free space for each column in the row number and value arrays allows a new entry to be added by moving only entries in the column of the new entry.

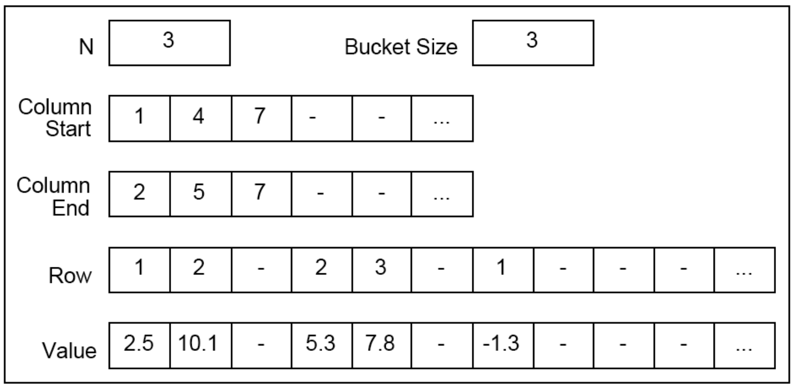

Figure C.1. Modified compressed-column format data structure before squeezing (provided by Rodney Jacobs)

Once FEM has completed the computation of A, the remaining free space is removed from the modified compressed-column data structure by moving entries down in the row number and value arrays and updating the column pointers in the column pointer arrays. Once the free space has been squeezed out of the data structure, the starting column pointer array, the row numbers array, and the values array are in fact a compressed-column format data structure that is compatible with SuperLU.

The FEM computation requires access to entries in A by row and column number as well as the ability to insert new non-zero values in A. Compressed-column format allows efficient access to non-zero entries in A by row and column numbers. The column number can be used to determine the range of index values in the row number and value arrays that contain non-zero entries for the specified column. Since entries in this index range are stored in order by row number, a binary search can be used to quickly find the index value for a specific row. If the row number is located, then the entry value can be read from the values array. If the row number is not found, then the entry value must be zero.The ice sheet model requires performing the FEM calculation for each time step of the model. Once the row and column numbers of non-zero entries produced by FEM is determined in the first iteration, they remain constant in all subsequent iterations. After the first iteration is complete, the values array is zeroed and the column pointer and row number arrays are left unchanged. No new entries are added to the data structure in subsequent iterations, so there is no need for additional free space or the attendant squeeze operation.The ice sheet model is written in Fortran 77. A C interface routine that is callable from the ice sheet model was written to invoke the needed UMFPACK and SuperLU functionality to solve the system of equations.

The software package and interface routines have been integrated with UMISM for this project by Rodney Jacobs [40].

BIOGRAPHY OF THE AUTHOR

Aitbala Sargent was born in Kazakhstan. She attended Lomonosov Moscow State University in Moscow, Russia, and graduated in 1986 with a Kandidat Nauk in Mathematics and Physics degree in Mathematics. After working in the Mathematics Department of Kazakh State University in Almaty, Kazakhstan, for nine years as an assistant professor, she attended the University of Maine in Orono and graduated in 1997 with a Master of Art degree in Economics. After working for three years as an Economic specialist at the United State Agency for International Development in Almaty, Kazakstan, she moved to Orono, Maine. For the last eight years, Aitbala, has been attending University of Maine, which she graduated with the degree of Master of Science in Computer Science in 2006, as well as working as an adjunct instructor in Husson college, Bangor, Maine, and in the University of Maine, Orono, Maine. Aitbala lives in Orono with her husband Lou and their daughter Aiesha.

Aitbala is a candidate for the degree of Doctor of Philosophy in Computer Science from the University of Maine in May, 2009.