Diss. ETH No. 19948

Computational methods for ice flow simulation

A dissertation submitted to

ETH ZÜRICH

for the degree of

Doctor of Sciences

presented by

JEDEDIAH ASPEN KALLEN-BROWN

Master of Science (Mathematics) University of Alaska Fairbanks
Bachelor of Science (Physics) University of Alaska Fairbanks
Bachelor of Science (Mathematics) University of Alaska Fairbanks

born

4 June, 1983

citizen of

United States of America

accepted on the recommendation of

Prof. Dr. M. Funk, examiner

Prof. Dr. W. Kinzelbach, examiner

Dr. M. Lüthi, examiner

Prof. Dr. E. L. Bueler, co-examiner

Dr. B. F. Smith, co-examiner

2011

Contents

Abstract ix
Zusammenfassung xi

1 Introduction **1**

2 Efficient nonlinear solvers for nodal high-order finite element methods in 3D **4**
2.1 Introduction 4
2.2 Newton-Krylov 6
2.2.1 The Newton iteration 6
2.2.2 Krylov methods 6
2.2.3 Performance 7
2.3 High order finite element methods 8
2.3.1 Discretization 8
2.3.2 Residual evaluation 8
2.3.3 Jacobian representation 9
2.3.4 Preconditioning 10
2.3.5 Performance of matrix operations 10
2.4 Numerical examples 11
2.4.1 p-Poisson 11
2.4.2 Stokes 14
2.5 Discussion 17

3 Textbook multigrid efficiency for hydrostatic ice sheet flow **19**
3.1 Introduction 19
3.2 Equations and Discretization 20
3.3 Solver and Implementation 21
3.3.1 Anisotropic Meshes 21
3.3.2 Dirichlet Boundary Conditions 23
3.3.3 Matrices 23
3.4 Numerical Examples 24
3.4.1 Algorithmic Scalability 24
3.4.2 Parallel Scalability on Blue Gene/P 26
3.4.3 Algebraic Methods 27
3.5 Conclusion 30

3.6 Addendum: Implicit free surface and erosion . . . 32

4 Software **35**
4.1 Multiphysics coupling . . . 36
4.1.1 Field-split preconditioning . . . 37
4.1.2 Linear algebraic interfaces to facilitate field-split preconditioning . . . 39
4.2 High throughput on cache-based architectures . . . 41
4.2.1 Sparse matrix kernels . . . 41
4.2.2 Small dense tensor product kernels . . . 45
4.3 Generic finite elements . . . 46
4.3.1 Dual order finite elements . . . 47
4.3.2 Input/output and visualization for high-order mixed spaces . . . 51
4.4 Verification . . . 52
4.5 Time integration . . . 57

5 Discretization issues **59**
5.1 Regularity and approximation . . . 59
5.1.1 Singularities in the continuum formulation . . . 60
5.1.2 Approximation spaces . . . 61
5.2 Implementation of boundary conditions . . . 63
5.2.1 Dirichlet boundary conditions . . . 63
5.2.2 Slip . . . 64

6 Steady-state viscous heat transport at Jakobshavn Isbræ **67**
6.1 Formulation and methods . . . 68
6.1.1 Problem description . . . 68
6.1.2 Numerical solution . . . 73
6.1.3 Verification . . . 77
6.1.4 A simple problem . . . 79
6.2 Jakobshavn Isbræ . . . 82
6.2.1 Scaling . . . 82
6.2.2 Meshing . . . 82
6.2.3 Solutions . . . 84

7 Conclusion and Outlook **89**
7.1 Future directions . . . 89

References . . . 93

List of Figures

2.1 The left panel shows truncation error as a function of total degrees of freedom for a 3D Poisson problem with smooth but rapidly varying solution. The thick horizontal line represents an acceptable accuracy and is only modestly within the asymptotic range for all approximation orders. The right panel shows the total number of nonzeros in the Jacobian which is an optimistic lower bound for the flops required for the solve. 5

2.2 A patch of four Q_5 elements with one Q_1 subelement shaded. 10

2.3 Linear solve time for 3D Poisson with relative tolerance of 10^{-8} using assembled Q_2 elements and unassembled Q_3, Q_5, and Q_7 elements preconditioned by an assembled Q_1 operator. 12

2.4 Two all-hexahedral meshes, `twist` and `random`, containing nearly degenerate elements (element condition number shown). 13

2.5 Linear solve time for 3D Stokes with relative tolerance of 10^{-6}. For $Q_2 - Q_1$ and $Q_3 - Q_2$ elements, convergence is significantly slower than with $Q_k - Q_{k-2}$, but apparently scalable despite being somewhat erratic. 16

3.1 A cutout colored by velocity magnitude for flow over the bumpy bed of test X at $L = 10\,\text{km}$ with $m = 1/10$ nearly plastic basal yield model. The cut on the left shows along-flow velocity as the ice hits the sticky region, the cut on the right shows across-flow shear structure. 25

3.2 Grid-sequenced Newton-Krylov solution of test X. The solid lines denote nonlinear iterations, and the dotted lines with $\times$ denote linear residuals. 26

3.3 Grid sequenced Newton-Krylov convergence for test Y. 26

3.4 Average number of Krylov iterations per nonlinear iteration. Each nonlinear system was solved to a relative tolerance of 10^{-2}. 27

3.5 Strong scaling on Shaheen for different size coarse levels problems and different coarse level solvers (see text for details). The straight lines on the strong scaling plot have slope -1 which is optimal. Grid sequencing is used, but only the nonlinear solve on the finest level is shown since strong scalability is most important when many time steps are needed. 28

3.6 Weak scaling on Shaheen with a breakdown of time spent in different phases of the solution process. Times are for the full grid-sequenced problem instead of just the finest level solve. 28

3.7 A steady-state solution for ISMIP-HOM test C (Pattyn et al., 2008) at 10km computed in 19 iterations. The elevated surface is exaggerated surface height and the color in the solid domain is velocity. 33

3.8 Bed profile eroded from a flat bed after 300 ka with test C slipperiness perturbation. Time steps are 30 ka at this point in the simulation. 33

4.1 Memory and floating point requirements for matrix-free tensor-product application of an operator versus representation as an assembled matrix stored in `BAIJ`(b) format. The same operation $y \leftarrow Ax$ is applied in both cases, the storage is just different. A "result" is a single scalar entry in y, regardless of the block size b. 47

4.2 Solution of the large-deformation nonlinear elasticity problem with manufactured solution on a Q_5 mesh. 56

6.1 The numerical approximation to the manufactured solution (6.1.15) as computed using a $Q_3 - Q_2 - Q_3$ finite element discretization on a $12 \times 12 \times 12$ mesh. The converging streamlines are symmetric from below. Energy isosurfaces are shown, with the phase transition occuring at approximately $E = 0$. The numerical solution is accurate to 4 digits for momentum and energy and 2 digits for pressure, evaluated using the maximum norm and a higher order quadrature rule. The gradients are accurate to 3 digits for momentum and energy, with 3 % error in the pressure gradient. 78

6.2 Convergence rates for $Q_3 - Q_2 - Q_3$ under h-refinement. 79

6.3 Energy isosurfaces and velocity streamlines for the block on an inclined plate. The energy scale has been shifted so that the phase transition occurs at approximately $E = 0$. The warmest region (inside the red isosurface) occurs where the singular viscous heat production (see Figure 6.4) balances advection and combined thermal and moisture diffusion. 80

6.4 Isosurfaces of the viscous heat production rate $\eta D\boldsymbol{u}_i : D\boldsymbol{u}_i$ and velocity streamlines for the block on an inclined plate. Viscous heat production has a $1/r$ singularity at both the upstream and downstream corners, clearly stronger at the downstream corner. 81

6.5 Regions of interest for Jakobshavn Isbræ. The outer black line marks the area on which bed and surface measurements were accurate enough to perform decimation. The inner (thick) black line marks the meshed region. 83

6.6 The volume to be meshed resting on the geometric model for the bed. 84

6.7 Computed momentum density streamlines for the ice stream region at Jakobshavn Isbræ. The highest velocity occurs in a deep, narrow region of the ice stream. 85

6.8 Computed energy density (top) and momentum density (bottom) with flow streamlines for the ice stream region at Jakobshavn Isbræ. 86

6.9 Contours of potential temperature (°C) and velocity streamlines (colored by speed measured in $\mathrm{m\,a^{-1}}$). Variability in layer thickness is clearly caused by tributaries and geometry. The maximum potential temperature is 7.3 °C which corresponds to a maximum melt fraction of 4.8 %. The sharp transition from cold upstream boundary condition is just visible at the top of this plot. 87

List of Tables

2.1 Assembly and solve time (seconds) for a 3D Poisson problem with 121^3 degrees of freedom (120^3 for Q_7), relative tolerance of 10^{-8}. 12

2.2 Iteration counts for four different meshes using ML, BoomerAMG (BMG), and Cholesky (Chol) preconditioning. For Q_1 `random`, BoomerAMG failed, producing NaN. ML on the `flat` mesh was run with stronger smoothers as described in the text. Note that the problem size increases with element order. 14

2.3 Time (seconds) spent in various stages of a nonlinear solve for Q_3 elements preconditioned by ILU(0) and ILU(1) applied to the corresponding Q_1 matrix. The first columns solve the linear system to a relative tolerance of 10^{-4} with a maximum of 60 Krylov iterations per Newton, the latter use the Eisenstat-Walker(1996) method for adjusting solver tolerances with a maximum of 30 Krylov iterations. The events *MF MatMult* (matrix-free Jacobian application), *PCSetup*, and *PCApply* are part of the *Krylov* iteration. Additional *Residual* evaluations are needed when the line search is activated. 15

2.4 Problem size and Krylov iterations to solve the Stokes problem to a relative tolerance of 10^{-6} with elements of different orders using right-preconditioned GMRES(30) and field-split ML on the A block. Each restart caused an approximately 2-iteration stall. . . 16

3.1 Throughput (Mflop/s) for different matrix formats on Core 2 Duo (P8700) and Opteron 2356 (two sockets). `MatSolve` is a forward- and back-solve with incomplete Cholesky factors. The AIJ format is using "inodes" which unrolls across consecutive rows with identical nonzero pattern (pairs in this case). 24

3.2 Average number of GMRES iterations per Newton iteration for one-level domain decomposition with different overlap and fill. 29

3.3 Average number of GMRES iterations per Newton iteration for field-split preconditioners with different ways of combining the splits and different solvers within the splits. BoomerAMG used 7 levels and ML had 3 with the same solver parameters as discussed in the text for the coupled approach. 30

3.4 Average number of GMRES iterations per Newton for different multigrid preconditioners. 30

4.1 Throughput (Mflop/s) and percentage of STREAM Triad bandwidth, assuming optimal vector reuse, for three matrix kernels and different matrix formats. The test machines are a Core 2 Duo (P8700) and an Opteron 2356 (two sockets). The test problem is a Q_1 finite element discretization with block size of 2 on a 3D mesh (essentially a 27-point stencil) in which each process has $64 \times 64 \times 63$ nodes. 44

4.2 Assembly time for the Laplace and p-Bratu problems with different element types and libraries. All cases have the same number of degrees of freedom, $65^3 = 274625$. The structured problem is treated as unstructured by `libMesh`, `Deal.II`, and Dohp, with granularity shown in the Connectivity column. `DMDA` uses a structured grid and is specialized for Q_1 assembly. 51

4.3 Convergence rates for the large-deformation elasticity problem with manufactured solution. 55

6.1 Physical constants used for the viscous heat transport problem. The same constants are used in Aschwanden et al. (2011). 71

6.2 Nonlinear convergence rates. 78

Abstract

A large amount of scientific effort worldwide is focused on understanding and predicting the advance and consequences of climate change, due to its potentially disastrous affects on human society. Numerical models are playing an ever-increasing role in the analysis of complex processes, but current modeling approaches are limiting the scope of problems that can be addressed. With the ever-increasing complexity, it is difficult to verify correctness of the implementation, assess accuracy of the simulation, or distinguish between numerical and modeling errors. The established strategies for model coupling, while generally thought to be necessary in order to manage complexity, cause significant stability and accuracy problems. Efficiency of nonlinear solvers represent a further obstacle to high resolution and advanced analysis techniques such as optimization, uncertainty quantification, and stability analysis. Present implementations also tend to use low-order discretizations which poorly utilize emerging hardware, are low accuracy, and cause numerical artifacts in some cases. This thesis contains contributions to each of these challenges.

A new perspective on high-order methods for finite element analysis is introduced. This formulation is well-suited to advances in linear and nonlinear solvers and offers dramatically better utilization of modern hardware than conventional methods. Dohp, a new general purpose library based on this method is presented, and the performance is shown to be several times faster than other widely used finite element libraries. Through new software interfaces, this performance is achieved while retaining more run-time flexibility in terms of element and preconditioning choice, and drastically better performance as the order of the element increases. The library also retains more geometric information than existing open source libraries, permitting more natural coupling to CAD and geometric models, as well as the implicit solution of equations in which the domain is part of the solution.

A new Newton-Krylov-Multigrid solver for the hydrostatic equations of ice sheet flow is presented. The high cost of solving the hydrostatic equations using conventional methods has been the principle impediment to their use in large-scale ice sheet models, causing existing models to fall back to simpler momentum balance models. In addition to poor algorithms, the community has also suffered from lack of quality parallel implementations, thus further limiting the scope of problems that could be solved. The new solver demonstrates textbook multigrid efficiency on a variety of demanding problems, offering several orders of magnitude speedup for problem sizes of interest, and nearly perfect strong and weak scalability on parallel hardware.

A new algebraic interface for multiphysics coupling is introduced. Robust coupling of multiple interacting physical processes is a challenging problem in which many commonly used methods are fundamentally inadequate. The best methods are highly problem dependent, change as the number of coupled processes grows, and are a highly active area of research. A crucial limitation of earlier software was that trying different methods generally involved a great deal of error-prone software development by the user. This poor software support made it difficult to test the quality and performance of different methods, thus locking projects in to methods that may actually be ill-suited to the problems that are eventually encountered. The new algebraic interface allows an arbitrary number of physical processes to be coupled using a wide range of methods which can be selected and combined at run-time. It permits straightforward reuse of single physics modules with no code modification, thus offering better support for model verification and extensibility. The interface offers higher performance and a great deal more flexibility in choice of methods than previous software. This software, along with implicit time integrators for differential algebraic equations and optimal explicit strong stability preserving integrators for hyperbolic

systems, has been added to PETSc and is in production use by several external groups.

Improvements in throughput on modern hardware are presented. Current methods for solving partial differential equations exhibit very low utilization of modern hardware, often less than 5 percent, due to their overwhelming dependence on memory bandwidth. Part of this under-utilization was due to implementation issues with sparse matrix kernels preventing good reuse of high-level caches. This was rectified within this work by improving PETSc's sparse matrix kernels by 20 to 30 percent, and performance is now close to the theoretical limit of the hardware. The more fundamental limitation of memory bandwidth cannot be overcome by implementation optimization; it requires changing the underlying algorithm. In the context of the finite element library Dohp, this can be achieved by eschewing assembled sparse matrices in favor of a matrix-free representation that has higher arithmetic intensity and uses much less memory for everything beyond lowest order elements. This transformation permits an order of magnitude improvement in hardware utilization and is transparently available to the user in the Dohp library. Improved support for such unassembled representations was integrated into the multi-physics coupling interface.

Robustness and accuracy requirements for ice flow problems place many constraints on the discretization and treatment of boundary conditions. Many of these technical requirements are undocumented in the glaciology literature and hampering current efforts for robust simulation. These technical issues are investigated and conclusions are drawn, with practical consequences to the present work and future development of methods for ice flow.

Current formulations for polythermal ice do not account for density variation caused by melt fraction and thus commit a conservation error of first order in the melt fraction. A new continuum formulation that exactly conserves mass, momentum, and energy independent of the melt fraction is presented. A high order finite element discretization for this system is proposed and numerical accuracy is addressed using manufactured solutions. This formulation treats all terms, including energy transport, implicitly in time, which allows the direct application of Newton-Krylov methods to compute the steady state. Steady state solutions are useful for parameter inversion, "spin up", and stability analysis. They are conventionally computed using direct time integration with a time step size constrained by the CFL stability criterion. With this constraint, they require a mesh-dependent number of time steps, typically very large, to reach steady state. The Newton-Krylov method converges in a small, mesh-independent number of iterations. This steady-state solver is applied to a section of the ice stream channel at Jakobshavn Isbræ. Setting up a model of an outlet glacier using realistic geometry and boundary conditions is a time-consuming task. This is especially true if a geometric model is needed to define slip conditions, or if the mesh needs to conform to the grounding line. Visualization is also complicated by the need to georeference model results. These difficulties have been partially mitigated by having the analysis code work with georeferenced input in any format and any projection, and produce georeferenced output.

The purpose of this thesis is not to make a specific prediction, but rather to improve the methods and process available for future predictive modeling, especially by glaciologists less interested in numerical and computational issues.

Zusammenfasssung

Viele der weltweiten wissenschaftlichen Anstrengungen sind darauf ausgerichtet, das Verständnis des Klimawandels und seiner Konsequenzen zu verstehen und vorherzusagen, dies insbesondere auch im Hinblick auf die möglicherweise katastrophalen Auswirkungen auf die Gesellschaft. Numerische Modelle spielen eine immer wichtigere Rolle bei der Analyse komplexer Prozesse, allerdings ist der Anwendungsbereich der Modelle limitiert durch die gegenwärtigen Modell-Ansätze. Mit zunehmender Modell-Komplexität ist es schwierig, die Güte der Implementation zu verifizieren, die Genauigkeit der Simulation zu bestimmen, und zwischen numerischen und Modellfehlern zu unterscheiden. Die etablierten Strategien der Modellkopplung, die als notwendig zur Bewältigung der Komplexität angesehen werden, verursachen signifikante Stabilitäts- und Genauigkeitsprobleme. Die Effizienz von nichtlinearen Solvern sind ein weiteres Hindernis für hohe Auflösung und fortgeschrittene Analysetechniken wie etwa Optimierung, Quantifizierung der Ungenauigkeit und Stabilitätsanalyse. Heutige Implementierungen verwenden Diskretisierungen niedriger Ordnung welche moderne Hardware nur schlecht ausnutzt, sind von niedriger Genauigkeit, und rufen in gewissen Fällen numerische Artefakte hervor. Diese Arbeit enthält Beiträge zu jeder dieser Herausforderungen.

Eine neue Perspektive auf Finite-Elemente Methoden höherer Ordnung wird eingeführt. Diese Formulierung ergänzt die Fortschritte bei linearen und nichtlinearen Solvern, und nützt die Eigenschaften moderner Hardware viel besser aus als herkömmliche Methoden. Eine neuentwickelte, allgemein verwendbare Programm-Bibliothek mit Namen Dohp wird präsentiert, deren Performance um ein vielfaches besser ist als diejenige von anderen gebräuchlichen Finite-Elemente Programm-Bibliotheken. Die hohe Performance wird erreicht durch neue Software-Interfaces, während gleichzeitig die Flexibilität zur Laufzeit bei der Auswahl von Elementen und Preconditionern grösser ist. Die Performance des Codes steigt drastisch mit höherer Element-Ordnung. Diese Programm-Bibliothek stellt auch mehr geometrische Information bereit als andere Open-Source Codes, und erlaubt dadurch eine natürliche Kopplung mit CAD und geometrischen Modellen, sowie die implizite Lösung von Gleichungssystemen, in denen die Modell-Geometrie Teil der Lösung ist.

Ein neuer Newton-Krylov Multigrid-Solver für die hydrostatischen Eisschild-Gleichungen wird dargestellt. Der hohe Aufwand zur Lösung der hydrostatischen Gleichungen mit konventionellen Methoden war bisher der Hauptgrund für deren Nicht-Gebrauch in grossen Eisschildmodellen – stattdessen werden einfachere Approximationen der Impuls-Bilanz verwendet. Neben den schlechten Algorithmen hat die Modellier-Gemeinde auch keine gute Implementation für Parallelcomputer zur Hand, was den Anwendungsbereich weiter stark einschränkt. Der neue Solver zeigt eine Multigrid-Effizienz wie aus dem Lehrbuch für mehrere anspruchsvolle Probleme, bietet eine Laufzeitverminderung um Grössenordnungen für interessierende Problemgrössen, und zeigt nahezu perfekte starke und schwache Skalierbarkeit auf paralleler Hardware.

Ein neues algebraisches Interface für die Kopplung verschiedener Modell-Codes wird eingeführt. Die robuste Kopplung mehrerer physikalischer Prozesse ist eine Herausforderung für welche viele der normalerweise verwendeten Methoden inadäquat sind. Welche Koppelungsmethode sich am besten eignet ist problemabhängig, ändert sich mit der Anzahl der betrachteten Probleme, und sind ein Gebiet aktiver Forschung. Eine entscheidende Einschränkung bisheriger Software war, dass das Ausprobieren verschiedener Methoden üblicherweise einen grossen Programmieraufwand des Benutzers erforderte. Die mangelnde Unterstützung durch die Software erschwerte das Testen von Qualität und Performance verschiedener Methoden, und führte oft zur Verwendung inadäquater Methoden für die zu lösenden Probleme.

Das neue algebraische Interface erlaubt es, eine beliebige Zahl physikalischer Prozesse mit einer grossen Auswahl an Methoden zu koppeln, welche zur Laufzeit ausgewählt und kombiniert werden können. Es erlaubt die unveränderte Verwendung bestehender Programmmodule für die einzelnen Prozesse, und bietet daher bessere Unterstützung von Verifikation und Extensibilität. Das Interface bietet eine höhere Performance und weitaus mehr Flexibilität bei der Auswahl der Methoden als bisherige Software. Diese Software, zusammen mit impliziten Zeitintegratoren für differentiell-algebraische Gleichungen sowie optimaler expliziter, stark stabilitätserhaltender Integration für hyperbolische Systeme, wurde in PETSc implementiert und ist bei mehreren externen Forschungsgruppen in Gebrauch.

Verbesserungen des Durchsatzes auf moderner Hardware werden dargestellt. Bisherige Methoden zur Lösung partieller Differentialgleichungen zeigen eine schlechte Ausnützung moderner Hardware, oft unter 5 Prozent, wegen der überwiegenden Abhängigkeit von der Memory-Bandbreite. Zum Teil war diese schlechte Ausnützung auf Implementationsprobleme der "Sparse Matrix Kernels" zurückzuführen, die zu schlechter Auslastung des high-level Caches führten. Im Rahmen dieser Arbeit wurden die PETSc Sparse Matrix Kernels um 20 bis 30 Prozent verbessert – die Performance ist nun nahe der theoretischen Grenze der Hardware. Der fundamentaleren Limitierung von Memory-Bandbreite lässt sich nicht mit einer optimierten Implementierung beikommen, dafür sind Veränderungen des Lösungs-Algorithmus notwendig. Im Kontext der Finite-Elemente-Bibliothek Dohp kann das durch Vermeiden von assemblierten "Sparse-Matrices" erreicht werden, indem der Matrix-freien Repräsentation Vorzug gegeben wird, welche eine höhere arithmetische Intensität und stark reduzierte Speicheranforderungen für alle Elemente, ausser jener der niedrigsten Ordnung, hat. Diese Transformation erlaubt eine Verbesserung um eine Grössenordnung im Hardware-Gebrauch, und ist in Dohp für den Benutzer transparent anwendbar. Eine verbesserte Unterstützung für nicht-assemblierte Repräsentationen wurde auch im Multi-Physik-Interface integriert.

Die Robustheits- und Genauigkeitsanforderungen für Eisfliessen schränken die Auswahl der Diskretisierung und die Behandlung der Randbedingungen ein. Viele dieser technischen Anforderungen sind in der Glaziologie-Literatur nicht dokumentiert, und behindern gegenwärtige Anstrengungen einer robusten Simulation. Diese technischen Anforderungen werden untersucht und Folgerungen gezogen, die praktische Konsequenzen für diese Arbeit, sowie auch für künftige Entwicklungen der Methoden für die Modellierung des Eisfliessens haben.

Gegenwärtige Formulierungen für polythermales Eis vernachlässigen die Dichtevariation durch die Schmelze, und verursachen einen Erhaltungs-Fehler erster Ordnung im Schmelzanteil. Eine neue Kontinuums-Formulierung mit exakter Erhaltung von Masse, Impuls und Energie, unabhängig vom Schmelzanteil, wird präsentiert. Für dieses System wird eine Diskretisierung hoher Ordnung vorgeschlagen, und die numerische Genauigkeit wird mit "Manufactured Solutions" untersucht. Diese Formulierung behandelt alle Terme, einschliesslich des Energietransportes, implizit in der Zeit, was die Verwendung von Newton-Krylov Methoden zur Berechnung von stationären Zuständen erlaubt. Solche stationäre Zustände sind nützlich zur Inversion von Parametern, "spin up", sowie zur Stabilitätsanalyse. Sie werden üblicherweise mit direkter Zeitintegration berechnet mit einer Schrittweite die durch die CFL Stabilitätsbedingung begrenzt ist. Diese Bedingung verlangt üblicherweise eine sehr grosse, netzabhängige Anzahl von Zeitschritten zur Erreichung eines stationären Zustandes. Die Newton-Krylov Methode hingegen konvergiert nach einer kleinen, netzunabhängigen Anzahl Iterationen. Dieser Solver wird für einen Ausschnitt des Jakobshavn Isbræ angewendet. Ein solches Modell eines Outlet-Gletschers mit realistischer Geometrie und Randbedingungen zu erstellen ist zeitaufwendig, insbesondere wenn ein geometrisches Modell für die Darstellung der Gleit-Randbedingung benötigt wird, oder wenn das Netz zur Grounding Line konform sein soll. Die Visualisierung der Resultate wird erschwert durch die Notwendigkeit einer Georeferenzierung. Diese Schwierigkeiten wurden teilweise dadurch gelöst, dass der Code mit georeferenzierten Eingabedaten in beliebigem Format und Projektion auskommt, und eine georeferenzierte Ausgabe erzeugt.

Das Ziel dieser Arbeit ist nicht, genaue Vorhersagen zu machen, sondern die Methoden und Prozesse für zukünftige Vorhersage-Modelle zu verbessern – dies insbesondere für Glaziologen die weniger an numerischen und implementations-bedingten Problemen interessiert sind.

Chapter 1

Introduction

A large amount of scientific effort worldwide is focused on understanding and predicting the advance and consequences of climate change, due to its potentially disastrous affects on human society. Increasingly, the results of numerical models are being used to influence decisions regarding energy policy, water rights, property values, geoengineering projects, and many more. Thus climate science, once an academic pursuit, is transforming into an engineering project of the grandest scale, the success of which will affect the entire planet. However, in stark contrast to other engineering disciplines, the "product development" latency for climate is a human lifetime or more, observations are difficult to obtain, and experimental perturbation is nearly impossible. This contributes to an environment in which the people creating and using numerical models never have direct feedback to assess the quality of the numerical results. Additionally, there is no direct financial incentive to produce quality results. Indeed, the quality of the results may never be known within the lifetime of the scientist who creates them.

The situation is very different in the industrial setting. If a computer program is used to design a plane that malfunctions, a bridge or dam that collapses, an engine with poor efficiency, or a reservoir engineering plan that results in poor recovery, there is process of accountability. Poor numerical results have direct financial and/or political consequences for the company or organization responsible. In such fields, it was learned early on that the process of verification and validation (Roache, 1998; Babuška and Oden, 2004) is of paramount importance. Numerical and computational issues cannot be merely an afterthought, and short cuts generally lead to incorrect results and poor understanding of complex phenomena.

This philosophy was summarized well in the 1986 Editorial Policy Statement on the Control of Numerical Accuracy for the Journal of Fluids Engineering (Roache et al., 1986). This statement was unequivocal that the time for less rigorous analysis and testing of methods had long passed, and announced that the jouarnal "will not accept for publication any paper reporting the numerical solution of a fluids engineering problem that fails to address the task of systematic truncation error testing and accuracy estimation." In particular, "a single calculation in a fixed grid will not be acceptable" and "the editors will not consider a reasonable agreement with experimental data to be sufficient proof of accuracy, especially if any adjustable parameters are involved." This policy was strengthened and extended in 1993 to its current form (American Society of Mechanical Engineers, 2004) which defines a list of ten criteria that must be used to assess the accuracy, robustness, efficiency, and proper documentation of a numerical method in order for it to be considered by the journal. Many other engineering journals have since adopted similar editorial policies. It is clear from the list, and has been confirmed by numerous colleagues in each discipline of climate modeling, that there does not exist a single climate component in any discipline that comes close to satisfying these publication criteria. This must change if the results of numerical models are to be taken seriously in the future.

Unfortunately, careful study of spatial discretization and grid convergence is not sufficient to enable the next generation of scientific inquiry for multiphysics systems such as climate or even ice dynamics in isolation. Time discretization and implicit solver performance must also be addressed. As argued by a recent Department of Energy panel (Simon et al., 2007), current models invariably rely on "first-order accurate operator-splitting, semi-implicit and explicit time integration methods, and decoupled

nonlinear solution strategies. Such methods have not provided the stability properties needed to perform accurate simulations over the dynamical time-scales of interest. Moreover, in most cases, numerical errors and means for controlling such errors are understood heuristically at best." This and a related report (Washington et al., 2009) prioritize further research in fast, robust linear and nonlinear solvers because these "will directly determine the scope of feasible problems to be solved" as, inevitably, implicit formulations and advanced analysis techniques such as optimization, uncertainty quantification, and stability and sensitivity analysis assume a central role.

Ice dynamics was identified by the fourth assessment report of the IPCC (Lemke et al., 2007) as a crucial source of uncertainty in sea level rise estimates, with no existing models capable of simulating the physical processes responsible for the large uncertainty. The underlying source of this uncertainty is a dynamical instability identified by Weertman (1974) and made rigorous by Schoof (2007). The problem of grounding line stability in locations such as Jakobshavn Isbræ is fundamentally three dimensional, constant factors are important, and the overall stability is determined by multi-scale behavior such as heat flux from the ocean through thin boundary layers and small bed features that can stabilize an unstable state. The "full" grounding line stability problem is on the frontier of computational science in many ways. It involves coupling physical processes in multiple domains interacting on multiple time scales through boundary layer processes, with material and geometric anisotropy, strong nonlinearity and heterogeneity, mixed characteristic PDEs, four varieties of interacting contact problems, and uncertainty in the geometry, coefficients, and constitutive models.

Advances in geoscience simulations will come from the synergy of

- more accurate physical models,
- more sophisticated mathematical algorithms, and
- more efficient implementations of these models and algorithms that take into account recent advances in computer hardware.

This synergy can only occur within a comprehensive well-thought-out software infrastructure that reflects all three facets of simulation. This thesis contains my work on each of these topics and their synthesis. It attempts to bring a more rigorous understanding of numerical and computational issues in ice flow modeling, with a focus on robust, extensible methods that scale to large problem sizes with efficient use of current and future hardware. An overarching theme is the development of extensible software that can be used to solve increasingly complex problems with minimal development time, while using the best possible methods. Much of this software has been added to the PETSc (Balay et al., 2012a) library[1] and is in production use by many external groups. Components which are not part of PETSc are available under a BSD-style open source license.

Chapter 2 investigates efficient nonlinear solvers for high-order finite element methods in a rather general setting. High order methods have accuracy benefits, but they are also capable of better utilizing modern hardware, and are necessary for certain conservation and stability properties that are needed for robust methods on practical meshes for ice flow problems. Chapter 3 presents an especially robust and scalable multigrid method for the hydrostatic equations of ice flow. These equations are considered to be an important intermediate description of the ice flow and were previously thought to be very expensive to solve. Our implementation improves by several orders of magnitude on the best previous results, both in time to solution and in maximum problem size. The code is available as a tutorial in PETSc.

Chapter 4 discusses several of the software components that were implemented to facilitate efficient solvers, high throughput, flexible and performant finite element methods for multi-physics problems,

[1] The Portable Extensible Toolkit for Scientific computing (PETSc) is an open source parallel nonlinear solvers package with support for many related tasks in scientific computing. It has thousands of users in academia and industry, with uses ranging from development of new iterative and preconditioning methods to computational physics and engineering problems in many fields, and forms the solver infrastructure for many discretization libraries as well as commercial software. I have been an active developer since 2008 and any developments that I felt belonged at PETSc's level of abstraction have been added to the library.

and designing code for easy verification. Chapter 5 investigates several relatively unique discretization requirements for ice flow problems. Finally, Chapter 6 introduces a new conservative formulation for polythermal ice, an implementation using the tools developed in earlier sections, and presents flow solutions for the ice stream channel at Jakobshavn Isbræ. Polythermal ice and refreezing due to the upward sloping bed near the grounding line is believed to play a role in the unique seasonal calving cycle at Jakobshavn.

Chapter 2

Efficient nonlinear solvers for nodal high-order finite element methods in 3D

This chapter is published as:
Brown, J. (2010). Efficient nonlinear solvers for nodal high-order finite element methods in 3D. *J. Scientific Computing*, 45(1):48–63.

Abstract. Conventional high-order finite element methods are rarely used for industrial problems because the Jacobian rapidly loses sparsity as the order is increased, leading to unaffordable solve times and memory requirements. This effect typically limits order to at most quadratic, despite the favorable accuracy and stability properties offered by quadratic and higher order discretizations. We present a method in which the action of the Jacobian is applied matrix-free exploiting a tensor product basis on hexahedral elements, while much sparser matrices based on Q_1 sub-elements on the nodes of the high-order basis are assembled for preconditioning. With this "dual-order" scheme, storage is independent of spectral order and a natural taping scheme is available to update a full-accuracy matrix-free Jacobian during residual evaluation. Matrix-free Jacobian application circumvents the memory bandwidth bottleneck typical of sparse matrix operations, providing several times greater floating point performance and better use of multiple cores with shared memory bus. Computational results for the p-Laplacian and Stokes problem, using block preconditioners and AMG, demonstrate mesh-independent convergence rates and weak (bounded) dependence on order, even for highly deformed meshes and nonlinear systems with several orders of magnitude dynamic range in coefficients. For spectral orders around 5, the dual-order scheme requires half the memory and similar time to assembled quadratic (Q_2) elements, making it very affordable for general use.

2.1 Introduction

High order spatial discretization has significant advantages over low order when high accuracy is required, especially when the solution is smooth. When the solution is only piecewise smooth, hp finite element methods (Demkowicz et al., 1989; Oden et al., 1989; Rachowicz et al., 1989; Schwab, 1998) are capable of exponential convergence, but high order discretization may yield higher quality results before asymptotic convergence rates are realized. Optimal hp meshes are usually defined in terms of minimizing the discretization error for a given number of degrees of freedom, and effective refinement strategies (e.g. Ainsworth and Senior, 1998; Demkowicz et al., 2002) have been developed relative to this metric.

Due to rapid loss of sparsity in the Jacobian under p-refinement, the total number of degrees of freedom is only weakly related to the practical performance metrics, computational time and storage requirements. For linear constant coefficient problems, this can be avoided by choosing special hierarchical bases that preserve sparsity (Karniadakis and Sherwin, 2005). When the spectral element method (a special case of nodal p-FEM which reuses the interpolation nodes for quadrature) is used with (nearly) affine elements,

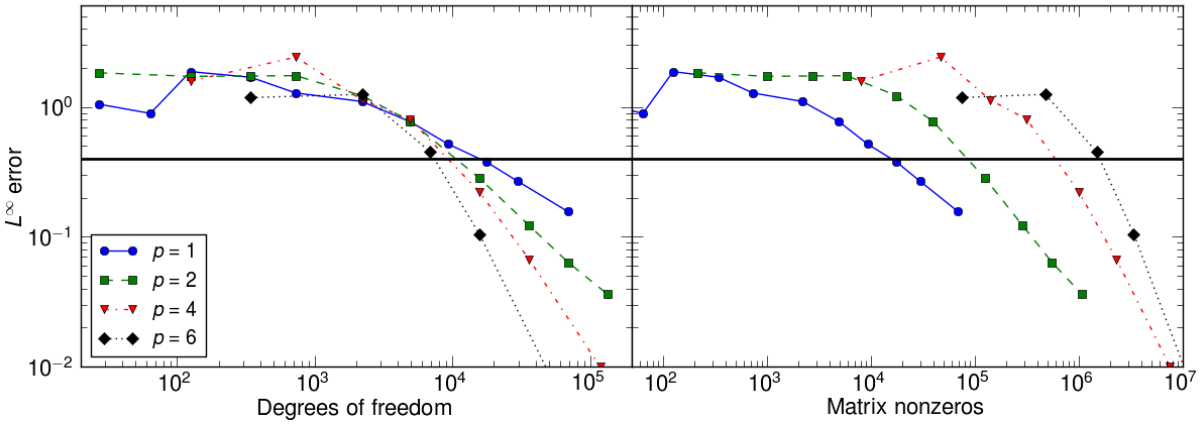

Figure 2.1: The left panel shows truncation error as a function of total degrees of freedom for a 3D Poisson problem with smooth but rapidly varying solution. The thick horizontal line represents an acceptable accuracy and is only modestly within the asymptotic range for all approximation orders. The right panel shows the total number of nonzeros in the Jacobian which is an optimistic lower bound for the flops required for the solve.

linear constant coefficient problems can be very efficiently solved using the fast diagonalization method combined with a multilevel coarse solve (Lottes and Fischer, 2005).

For nonlinear and variable coefficient problems in 3D, the element matrices are necessarily dense and fast diagonalization is not available. For order p tensor product bases on hexahedral elements, there are $n = (p+1)^3$ degrees of freedom per element, but the dense element matrices contribute $(p+1)^6$ nonzeros to the global matrix. Since most sparse matrix operations scale linearly with the number of nonzeros, both time and memory costs scale as $O(n^2)$ under p-refinement. A popular method for reducing linear algebra costs is static condensation which removes the interior degrees of freedom. This method is effective at moderate to high order in 2D because a large portion of the nodes are interior, but in 3D less than half the nodes are interior when $p < 9$. For a scalar problem with $p = 9$, condensation requires manipulating a dense 1000×1000 matrix for each element. This is very expensive in time and memory for only 1000 degrees of freedom, and still contributes a dense 488×488 matrix to the global system. An additional bottleneck of traditional hp-FEM is integration of element matrices. Naïve assembly of the true Jacobian using a p-point quadrature rule is $O(p^9)$ although it can be improved to $O(p^7)$ by exploiting the tensor product structure using sum factorization. The cost incurred by forming and manipulating these dense matrices is the primary reason why high order (even quadratic) elements are not more popular in applications where extremely high accuracy is not needed.

If extremely high accuracy is desired, conventional high-order methods are attractive because the extra cost is more than made up for by improved convergence rate in the asymptotic range. For practical simulations, the asymptotic range is typically only mildly realized because an acceptable accuracy is obtained shortly after the solution is *resolved*. This effect is illustrated in Figure 2.1. High-order methods must have comparable cost *per degree of freedom* to be competitive with low-order methods unless the desired level of accuracy is well into the asymptotic range.

To develop efficient solvers for high order elements, we examine the bottlenecks present in nonlinear solvers, and design the discretization to eliminate as many of these bottlenecks as possible. Our approach to preconditioning involves the method proposed by Orszag (1980) and further investigated by Deville and Mund (1985, 1990); Heys et al. (2005); Kim (2007), resulting in a scheme with storage costs equivalent to Q_1 (piecewise trilinear) elements independent of p and total CPU time which is mesh independent and only weakly dependent on p. The principle contributions of this paper are a formulation of this

preconditioner that is applicable to general nonlinear systems, a matrix-free representation of the Jacobian, and observation of a much lower "crossover" order p at which this class of methods is to be preferred over conventional methods.

2.2 Newton-Krylov

Jacobian-free Newton-Krylov (JFNK) methods combine Newton-type methods for superlinearly convergent solution of nonlinear equations with Krylov subspace methods for solving the Newton step without actually forming the true Jacobian (Knoll and Keyes, 2004). In this section, we introduce the method and highlight the performance bottlenecks.

2.2.1 The Newton iteration

The Newton iteration for the nonlinear system $F(x) = 0$ with state vector x involves solving the sequence of linear systems

$$J(x^k)s^k = -F(x^k), \quad x^{k+1} \leftarrow x^k + s^k, \qquad k = 0, 1, \ldots \tag{2.2.1}$$

until convergence which is frequently measured as a relative tolerance

$$\left\| \frac{F(x^k)}{F(x^0)} \right\| < \varepsilon_{\text{tol}}.$$

The structure and conditioning of the Jacobian $J(x) = \frac{\partial F(x)}{\partial x}$ is dependent on the equations and discretization, but for second order elliptic operators, the condition number $\kappa(J) = \|J\| \left\|J^{-1}\right\|$ scales as $O(h^{-2}p^4)$ where h is the mesh size and p is the approximation order.

2.2.2 Krylov methods

Krylov methods solve $Jx = b$ using the Krylov subspaces

$$\mathcal{K}_j = \text{span}\{\tilde{b}, \tilde{J}\tilde{b}, \tilde{J}^2\tilde{b}, \ldots, \tilde{J}^{j-1}\tilde{b}\}$$

where $\tilde{J} = P^{-1}J$, $\tilde{b} = P^{-1}b_0$ for left preconditioning and $\tilde{J} = JP^{-1}$, $\tilde{b} = b$ for right preconditioning. The convergence rate of various Krylov methods (e.g. CG, MINRES, GMRES, BiCGStab, QMR) depend in diverse ways on the spectral properties of $\tilde{J}$ (e.g. Nachtigal et al. (1992) constructs a linear system for each Krylov scheme listed above, for which that scheme beats all other schemes by a large margin) but a useful heuristic is that the iteration count scales as the square root of $\kappa(\tilde{J})$. When solving the Newton step (2.2.1) we rely on P^{-1} to overcome the $O(h^{-2}p^4)$ conditioning of the spatial discretization. Our preconditioners are obtained by applying an algorithm P to data J_p provided by the discretization of the equations, i.e. $P^{-1} = \mathsf{P}(J_p)$. Concretely, J_p almost always contains one or more assembled matrices, but may contain auxiliary information such as a multilevel hierarchy. The fundamental preconditioners P are relaxation (e.g. SOR) and (incomplete) factorization, which are used to build scalable and problem-specific preconditioners such as multigrid, domain decomposition, and Schur-complement. The notation P^{-1} comes from incomplete factorization in which $P = LU$ is available, but multiplication by P is never needed in the Krylov iteration and indeed is rarely available. A complete solver for $Jx = b$ is $\mathsf{K}(J, \mathsf{P}(J_p))$ which represents a choice of Krylov accelerator K, preconditioning method P, and preconditioning data J_p.

As discussed in the introduction, the Jacobian in (2.2.1) loses sparsity for high order methods so we will never form it explicitly. The finite difference representation

$$J(x)y = \frac{F(x + \varepsilon y) - F(x)}{\varepsilon}$$

for appropriately chosen ε is attractive because it requires no additional storage and no coding beyond function evaluation, however the adjoint J^T is not available, its accuracy is at best $\sqrt{\varepsilon_{\text{machine}}}$, and it performs unnecessary computation if function evaluation involves costly fractional powers or transcendental functions. The inaccuracy can lead to stagnation due to a poor quality Krylov basis, especially when restarts are needed. Automatic differentiation (AD) offers a full accuracy alternative and can produce adjoints in reverse mode, but such evaluations, especially in reverse mode, are usually more expensive than function evaluation due to suboptimal taping[1] strategies. In section 2.3.3 we detail an alternative with several advantages over finite differencing and which is similar to AD with optimal taping.

2.2.3 Performance

The most expensive parts of the Newton-Krylov iteration are assembling matrices for the preconditioner and subsequently solving for the Newton step. Most preconditioners require a setup phase after the matrices (and whatever else goes into J_p) are assembled. Scalable preconditioners for elliptic problems require a globally coupled coarse level solve and we take multigrid as the canonical example. With geometric multigrid the coarse level can be provided by the application by rediscretizing the governing equations while algebraic multigrid must analyze the matrix structure to determine the coarse component and then compute the Galerkin coarse operator (involving relatively expensive matrix-matrix products and producing a denser coarse operator). The coarse operator needs to be factored, a task that is often done redundantly. Finally, the solve involves matrix multiplication and smoothers on each level.

The relative cost of these three phases (assembly, setup, solve) is highly problem dependent, but we offer some general guidelines for perspective; see Gropp et al. (2000b); Knoll and Keyes (2004); Knoll and McHugh (1998) for more detailed analysis of these issues. Matrix assembly in finite element methods involves pointwise operations at quadrature points and is typically limited by instruction level parallelism and scheduling (especially for high-order methods). The setup and solve phases are limited by memory bandwidth on the fine levels and by network latency on the coarse levels. An important issue is simultaneously keeping the number of levels small so that relatively little work is done on the less efficient coarse levels while keeping the coarse operator small enough and sparse enough that it is cheap to factor. When strong preconditioners are used so to keep the iteration count low, setup can be much more expensive than the solve. Direct solvers are an extreme case of this, but multigrid can also have an expensive setup step, especially with many processors and Galerkin coarse operators.

Efficient linear solvers involve a tradeoff between many iterations with cheap preconditioners and fewer iterations with expensive preconditioners. With JFNK, the setup costs can be reduced by *lagging* the preconditioner (not recomputing it on every Newton step) at the expense of additional Krylov iterations. If matrix-free Jacobian application is inexpensive, the cost of these extra iterations may be affordable, making lagging appealing. On the other hand, if matrix assembly and preconditioner setup is cheap, there is no need to work with a stale preconditioner. Note that use of a stale Jacobian compromises quadratic convergence of the Newton method while a stale preconditioner only affects the convergence rate of the Krylov iteration.

Most of the time required to solve strongly nonlinear equations is spent before entering the neighborhood where Newton methods are quadratically convergent. Until the final phase, it is not important to solve the Newton step to high accuracy. In particular, $J(x^k)s^k = -F(x^k)$ may be "solved" so that

$$\left\|J(x^k)s^k + F(x^k)\right\| \leq \eta^k \left\|F(x^k)\right\|.$$

and the method converges q-superlinearly as long as $\eta^k \to 0$, and q-quadratically if $\eta^k \in O(\left\|F(x^k)\right\|)$. Various heuristics have been developed to automatically adjust the "forcing term" η^k to avoid oversolving without sacrificing quadratic convergence in the terminal phase, see Eisenstat and Walker (1996) for further discussion and a particularly useful heuristic which we use in the numerical examples.

[1] This is a slight abuse of terminology, "taping" in the AD context refers to intermediate values stored in memory (as opposed to checkpoints which are typically written to disk) during reverse-mode computation. We use the term more loosely to refer to any intermediate storage that makes multiplication by $J(u)$ and $J^T(u)$ more efficient than only storing the state vector u.

2.3 High order finite element methods

2.3.1 Discretization

Let $\{\hat{x}_i\}_{i=0}^p$ denote the Legendre-Gauss-Lobatto (LGL) nodes of degree p in ascending order on the interval $[-1,1]$, with the corresponding Lagrange interpolants $\{\hat{\phi}_i^p\}_{i=0}^p$. Choose a quadrature rule with nodes $\{r_i^q\}_{i=0}^q$ and weights $\{w_i^q\}$. The basis evaluation, derivative, and integration matrices are $\hat{B}_{ij}^{qp} = \hat{\phi}_j^p(r_i^q)$, $\hat{D}_{ij}^{qp} = \partial_x \hat{\phi}_j^p(r_i^q)$, and $\hat{W}_{ij}^q = w_i^q \delta_{ij}$. In the following, we use index-free notation and suppress the superscripts for clarity.

We extend these definitions to 3D via tensor product

$$\begin{aligned} \hat{\boldsymbol{B}} &= \hat{B} \otimes \hat{B} \otimes \hat{B} \\ \hat{\boldsymbol{D}}_0 &= \hat{D} \otimes \hat{B} \otimes \hat{B} \\ \hat{\boldsymbol{D}}_1 &= \hat{B} \otimes \hat{D} \otimes \hat{B} \\ \hat{\boldsymbol{D}}_2 &= \hat{B} \otimes \hat{B} \otimes \hat{D} \end{aligned} \tag{2.3.1}$$

with the diagonal weighting matrix $\hat{\boldsymbol{W}} = \hat{W} \otimes \hat{W} \otimes \hat{W}$. With isotropic basis order p and quadrature order q, these tensor product operations cost $2(p^3q + p^2q^2 + pq^3)$ flops and touch only $O(p^3 + q^3)$ memory. It should be noted that in the special case of the spectral element method where the same LGL points are reused for quadrature, $\hat{B}$ is the identity and differentiation reduces to $2p^4$ operations. In the present work, we prefer to use more accurate Gauss quadrature and do not find the extra floating point operations in basis evaluation to be prohibitively expensive. Every other required operation will be $O(p^3)$ or $O(q^3)$ so the performance of the operation (2.3.1) is crucial to the success of the method, see §2.3.5 for further discussion.

Let $\hat{K} = [-1,1]^3$ be the reference element and partition the domain Ω into hexahedral elements $\{K^e\}_{e=1}^E$ with coordinate map $x^e : \hat{K} \to K^e$ and Jacobian $J_{ij}^e = \partial x_i^e / \partial \hat{x}_j$. The Jacobian must be invertible at every quadrature point so we have $(J^e)^{-1} = \partial \hat{x} / \partial x^e$ and can express the element derivative in direction i as

$$\boldsymbol{D}_i^e = \Lambda\left(\frac{\partial \hat{x}_0}{\partial x_i}\right) \hat{\boldsymbol{D}}_0 + \Lambda\left(\frac{\partial \hat{x}_1}{\partial x_i}\right) \hat{\boldsymbol{D}}_1 + \Lambda\left(\frac{\partial \hat{x}_2}{\partial x_i}\right) \hat{\boldsymbol{D}}_2$$

where we have used the notation $\Lambda(x)_{ij} = x_i \delta_{ij}$ for expressing pointwise multiplication as a diagonal matrix. Note that forming $\boldsymbol{D}^e$ would ruin the tensor product structure so multiplication by $\boldsymbol{D}_i^e$ is done using the definition (2.3.1) followed by pointwise multiplication and sum. The transpose is defined similarly

$$(\boldsymbol{D}_i^e)^T = \hat{\boldsymbol{D}}_0^T \Lambda\left(\frac{\partial \hat{x}_0}{\partial x_i}\right) + \hat{\boldsymbol{D}}_1^T \Lambda\left(\frac{\partial \hat{x}_1}{\partial x_i}\right) + \hat{\boldsymbol{D}}_2^T \Lambda\left(\frac{\partial \hat{x}_2}{\partial x_i}\right).$$

With the element integration matrix $\boldsymbol{W}^e = \hat{\boldsymbol{W}} \Lambda(|J^e(\boldsymbol{r})|)$, we are prepared to evaluate weak forms over arbitrary elements. The global problem is defined using the element assembly matrix $\mathcal{E} = [\mathcal{E}^e]$ where each $\mathcal{E}^e$ extracts the degrees of freedom associated with element e from the global vector. In the special case of a conforming mesh and equal approximation order on every element, $\mathcal{E}$ will have a single unit entry per row. When the mesh is h- or p-nonconforming, a global basis can be chosen by using minimum order on the largest faces and edges, then the element basis can be constrained by writing it as a linear combination of global basis functions (Demkowicz et al., 1989).

2.3.2 Residual evaluation

We now turn to evaluation of the discrete residual statement $F(u) = 0$ in weak form. Since the weak form is always linear in the test functions, it can be expressed as a pointwise algebraic operation taking values

and derivatives of the current iterate $(u, \nabla u)$ to the coefficients of the test functions and derivatives $(v, \nabla v)$, i.e. the Dirichlet problem is to find u in a suitable space V_D such that

$$\langle v, f(u)\rangle = \int_\Omega v \cdot f_0(u, \nabla u) + \nabla v : f_1(u, \nabla u) = 0 \tag{2.3.2}$$

for all v in the corresponding homogeneous space V_0 where f_0 and f_1 contain any possible sources. For an n-component problem in d dimensions, $f_0 \in \mathbb{R}^n$ and $f_1 \in \mathbb{R}^{nd}$. Inhomogeneous Neumann, Robin, and nonlinear boundary conditions will add similar terms integrated over boundary faces. The fully discrete form is

$$\sum_e \mathcal{E}_e^T \left[(\boldsymbol{B}^e)^T \boldsymbol{W}^e \Lambda(f_0(u^e, \nabla u^e)) + \sum_{i=0}^{d} (\boldsymbol{D}_i^e)^T \boldsymbol{W}^e \Lambda(f_{1,i}(u^e, \nabla u^e)) \right] = \mathbf{0} \tag{2.3.3}$$

where $u^e = \boldsymbol{B}^e \mathcal{E}^e u$ and $\nabla u^e = \{\boldsymbol{D}_i^e \mathcal{E}^e u\}_{i=0}^2$. Note that all physics is contained in the pointwise operation $(u, \nabla u) \mapsto (f_0, f_1)$.

2.3.3 Jacobian representation

Jacobian application $w \mapsto J(u)w$ can be performed in the same way as residual evaluation. The associated bilinear form is expressible as

$$\langle v, \boldsymbol{J}(u)w\rangle = \int_\Omega \begin{bmatrix} v^T & \nabla v^T \end{bmatrix} \begin{bmatrix} f_{0,0} & f_{0,1} \\ f_{1,0} & f_{1,1} \end{bmatrix} \begin{bmatrix} w \\ \nabla w \end{bmatrix} \tag{2.3.4}$$

with the notation $f_{i,0} = \frac{\partial f_i}{\partial u}(u, \nabla u)$ and $f_{i,1} = \frac{\partial f_i}{\partial \nabla u}(u, \nabla u)$. This construction is completely general and applies to any H^1 Galerkin or Petrov-Galerkin method. The application of $J(u)$ and $J^T(u)$ can clearly be performed if $f_{i,j}(u, \nabla u)$ is stored at quadrature points (including similar terms at boundary quadrature points where boundary integrals are required).

There are two extreme cases for the storage of $f_{i,j}(u, \nabla u)$. The first is to simply store $(u, \nabla u)$ which requires $n(d+1)$ storage per quadrature point for an n-component system in d dimensions, but all the physics must be reevaluated in each Jacobian application. This option is slightly faster than standard use of automatic differentiation since the finite element mechanics to reevaluate $(u, \nabla u)$ is not needed. The other extreme is to fully evaluate $f_{i,j}(u, \nabla u)$ which requires $[n(d+1)]^2$ storage per quadrature point. Compare to $n^2(p+1)^d$ per node for the Q_p element matrices and (asymptotically) for globally assembled matrices.

For many physical systems, $f_{i,j}(u, \nabla u)$ possesses structure (e.g. sparsity, symmetry, isotropic plus rank-1 update) such that compared to the naïve representation, it is both cheap to store and cheap to multiply by. Such representations may be constructed from any form between $(u, \nabla u)$ and $f_{i,j}$. It is common for this efficient representation to be available at low cost during residual evaluation, and it may be stored cheaply since the memory bus is relatively unstressed at this time. The process from (2.3.2) to (2.3.4) is mechanical and is easily performed using symbolic algebra. Also note that AD may be used as a complementary technique, well-isolated to pointwise operations so that it doesn't interfere with global software design.

Note that with this representation, the adjoint of (2.3.4) is readily available with no additional programming effort, at exactly the same cost as the Jacobian itself. A worked example is given in §2.4.1, we find that our matrix-free Jacobian application is significantly faster than function evaluation. Since it is full accuracy and provides adjoints, this representation is clearly superior to finite difference Jacobians. Note that naïve automatic differentiation of F would not be aware of this efficient intermediate form, thus an AD Jacobian $J(u)$ would need to re-evaluate the base vector at quadrature points and compute $f_{i,j}\left[\begin{smallmatrix} w \\ \nabla w \end{smallmatrix}\right]$ using $(u, \nabla u)$. The intermediate form is roughly equivalent to an optimal taping strategy.

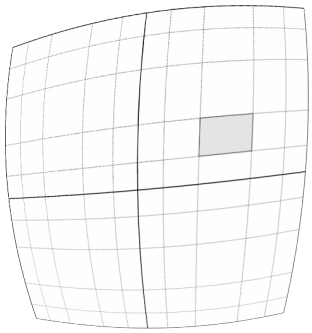

Figure 2.2: A patch of four Q_5 elements with one Q_1 subelement shaded.

2.3.4 Preconditioning

To precondition J, we assemble a matrix by rediscretizing the governing equations on Q_1 sub-elements defined by the LGL interpolation nodes as shown in Figure 2.2. In the notation of §2.2, this matrix is J_p and any preconditioner P for assembled matrices can be used in the Krylov iteration. This could be a direct solve, but that is rarely the most economical choice. When multigrid works well, such as for elliptic problems with smooth coefficients, we find that $\mathsf{P} = MG$ requires very few extra iterations compared to a direct solve $\mathsf{P} = LU$. This preconditioner has received substantial attention in the collocation and spectral element contexts (e.g. Orszag, 1980; Deville and Mund, 1985, 1990; Heys et al., 2005; Kim, 2007), but is not widely used for Galerkin discretization with independent quadrature.

2.3.5 Performance of matrix operations

Our test platform is a 2.5 GHz Intel Core 2 Duo (T9300) which can issue one packed double precision (two 64-bit values packed in each 128-bit register) add and one packed multiply per clock cycle with a latency of 3 and 5 cycles respectively. If there are no data dependencies and both execution units can be kept busy, it is possible to perform 4 flops per clock cycle resulting in a theoretical peak of 10 Gflop/s. The memory controller has a peak read throughput of 5.3 GB/s implying that dual core performance for matrix-vector products cannot exceed 900 Mflop/s, ignoring access costs for the vector (each matrix entry is 8 bytes plus 4 bytes for the column index, and supplies only 2 flops for an arithmetic intensity of 1 flop / 6 bytes). Note that this is less than 5% of the theoretical dual-core peak 20 Gflop/s. Even if the stencil was known so that column indices did not need to be stored explicitly, the operation cannot exceed 4 bytes per flop, despite the system being capable of performing nearly 4 flops for each byte loaded from memory.

For multi-component problems, we can exploit structural blocking to reduce the bandwidth required for column indices. For example, a 3-component problem with full coupling only needs to store one index per 3×3 block, improving arithmetic intensity to almost 1 flop / 4 bytes. In addition, it is advantageous to reorder the unknowns owned by each processor to reduce the matrix bandwidth (e.g. with reverse Cuthill-McKee), thus improving cache reuse by the vector during matrix-vector multiplication and preconditioning kernels (relaxation and solves with factors), and also the effectiveness of these preconditioning kernels. On our 3D elasticity test, these two optimizations combined for a 75% speedup relative to scalar formats in the natural ordering, see Gropp et al. (2000b) for further analysis of blocking and reordering.

High order methods allow much more effective utilization of today's multicore hardware. An implementation of the tensor product operation (2.3.1) operates entirely within L1 cache for practical basis orders and attains 5.7 Gflops for Q_3 on a single core. (This was written using SSE3 intrinsics since the author was unable to find a compiler that produced competitive code for this operation.) Since this kernel puts no pressure on the shared memory bus, all cores are completely independent and limited only by instruction scheduling and data dependence. Although only the tensor-product operation has been vectorized, multiplication by the matrix-free Q_3 Jacobian for a scalar-valued problem is only 2.5 times slower than the assembled Q_2 Jacobian when using one core of an otherwise idle socket.

2.4 Numerical examples

All examples with dual-order preconditioning were implemented as part of a new C library named *Dohp*, available from the author. Dohp is tightly integrated with PETSc (Balay et al., 2012a) and uses the ITAPS (ITAPS, 2011) interfaces for mesh and geometry services. In particular, we use the MOAB (Tautges et al., 2004) and CGM (Tautges, 2001) implementations of the *iMesh* and *iGeom* interfaces. In the examples using algebraic multigrid, we use smoothed aggregation from ML (Gee et al., 2006) and classical multigrid from BoomerAMG (Henson and Yang, 2002), all accessed through the common PETSc interface. The ML interface is designed to only produce aggregates which PETSc uses to construct the multigrid hierarchy and thus exposes all PETSc preconditioners as smoothers, while BoomerAMG is essentially a black-box preconditioner. Smoothed aggregation is known to scale slightly superlinearly, but in our tests ML was always significantly faster and less memory consuming than the typically more robust BoomerAMG. We use PETSc-3.0.0, ML-6.2[2], and BoomerAMG from Hypre-2.4.0b.

The `libMesh` (Kirk et al., 2006) library is used to provide a reference for conventional methods (based on assembling the true Jacobian). Since `libMesh` also uses PETSc for linear algebra, identical solver parameters are used so that the results are representative.

2.4.1 p-Poisson

The inhomogeneous $\mathfrak{p}$-Laplacian is

$$-\nabla\cdot(|\nabla u|^{\mathfrak{p}-2}\nabla u) - f = 0$$

where $1 \le \mathfrak{p} \le \infty$ (see Evans (2007) for discussion of these limiting cases). This equation is singular at $\nabla u = 0$ when $\mathfrak{p} < 2$ and degenerate when $\mathfrak{p} > 2$, so we solve a regularized variant with weak form: find $u \in V_D$ such that

$$\langle v, F(u)\rangle = \int_\Omega \eta \nabla v \cdot \nabla u - fv = 0 \tag{2.4.1}$$

for all $v \in V_0$ where $V_D = H^1_D(\Omega)$ includes inhomogeneous Dirichlet conditions, $V_0 = H^1_0(\Omega)$ is the corresponding homogeneous space, and $\eta(\gamma) = (\varepsilon+\gamma)^{\frac{\mathfrak{p}-2}{2}}$ is effective viscosity with regularization $\varepsilon > 0$ and $\gamma = \frac{1}{2}|\nabla u|^2$. We manufacture the forcing term f so that

$$u(x,y,z) = \cos(ax)\sin(by)\exp(cz)$$

solves (2.4.1) in $\Omega = [-1,1]^3$. Figure 2.1 shows convergence rates for the linear ($\mathfrak{p} = 2$) case $a,b,c = (16,15,14)$ with $1 \le p \le 7$. The expected $O(h^{p+1})$ is indeed observed in the asymptotic range, but in this section we are concerned with the time and memory needed for solution.

The Newton step for (2.4.1) corresponds to the weak form: find $w \in V_0$ such that

$$\langle v, J(u)w\rangle = \int_\Omega \eta\nabla v\cdot\nabla w + \eta'(\nabla v\cdot\nabla u)(\nabla u\cdot\nabla w) = -\langle v, F(u)\rangle$$

for all $v \in V_0$, where $J(u)$ is the Jacobian of F at u. It is informative to rewrite the integrand as

$$\nabla v : \left[\eta\mathbf{1} + \eta'\nabla u\otimes\nabla u\right] : \nabla w$$

to clarify the effect of the Newton linearization. Since $\eta' < 0$ for $\mathfrak{p} < 2$, the (heterogeneous) isotropic conductivity tensor $\eta\mathbf{1}$ is being squished in the direction ∇u. In the singular limit $\mathfrak{p} \to 1$, it flattens completely which has the effect of allowing diffusion only in level sets of u.

To illustrate the taping strategy of §2.3.3 for the $\mathfrak{p}$-Laplacian, first note that it is desirable to store ∇u at each quadrature point in order to avoid it's expensive recomputation. Fractional powers are one of the

[2]We have observed that ML-6.2 produces much higher quality aggregates than ML-5.0 for the assembled Q_2 case, leading to a 50% speedup and significantly lower memory usage.

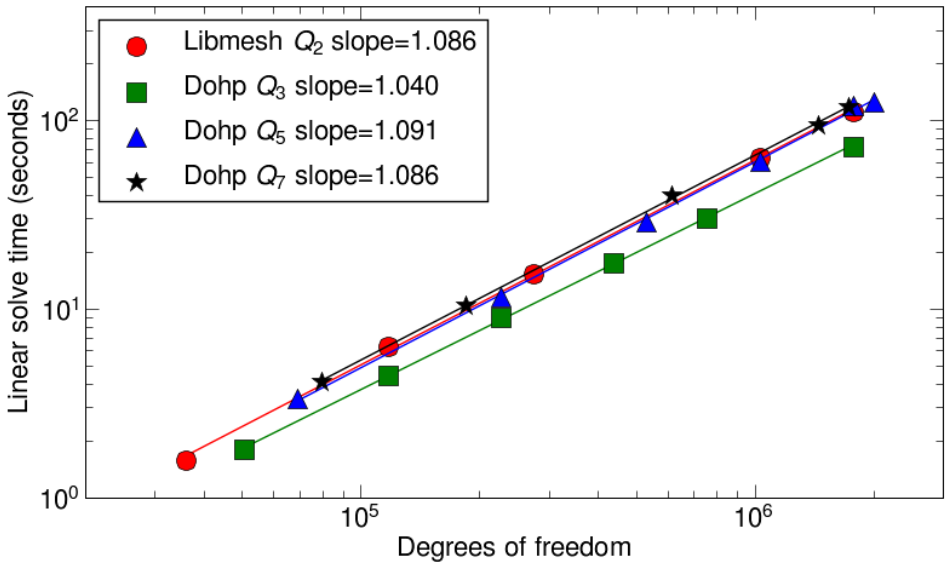

Figure 2.3: Linear solve time for 3D Poisson with relative tolerance of 10^{-8} using assembled Q_2 elements and unassembled Q_3, Q_5, and Q_7 elements preconditioned by an assembled Q_1 operator.

Event	`libMesh` Q_2	Dohp Q_3	Dohp Q_5	Dohp Q_7
Assembly	41	25	24	23
Krylov	111	73	119	117
MF MatMult	—	36	55	55
PCSetUp	16	10	12	9
PCApply	82	27	51	52
CG its	34	23	41	49
Mat nonzeros	111 M	44.7 M	44.7 M	44.3 M

Table 2.1: Assembly and solve time (seconds) for a 3D Poisson problem with 121^3 degrees of freedom (120^3 for Q_7), relative tolerance of 10^{-8}.

more expensive mathematical operations so we would also like to avoid recomputing them. This suggests taping $(\eta, \nabla u)$ which are explicitly available during residual evaluation (2.4.1). If a few Krylov iterations will be required, it becomes beneficial to amortize computation of

$$\eta' = \frac{\partial \eta}{\partial \gamma} = \frac{p-2}{2} \frac{\eta}{\varepsilon + \gamma}$$

which involves one multiplication and one division when taped during function evaluation, very cheap compared to the fractional power, and less expensive than recomputing γ on the first Krylov iteration.

A further enhancement[3] to taping $(\eta, \eta', \nabla u)$ is $(\eta, \sqrt{-\eta'} \nabla u)$ which is both compact and minimizes the operations necessary to apply the Jacobian.

Figure 2.3 shows linear solve performance compared to conventional methods for quadratic elements. The matrices for conventional Q_2 elements were assembled using the `libMesh` library and the solver in all cases was conjugate gradients preconditioned by ML. The time spent in each phase for the largest problem size is shown in Table 2.1. Note the greatly reduced assembly time compared to conventional Q_2 elements.

[3] `sqrt` requires the same number of cycles as division

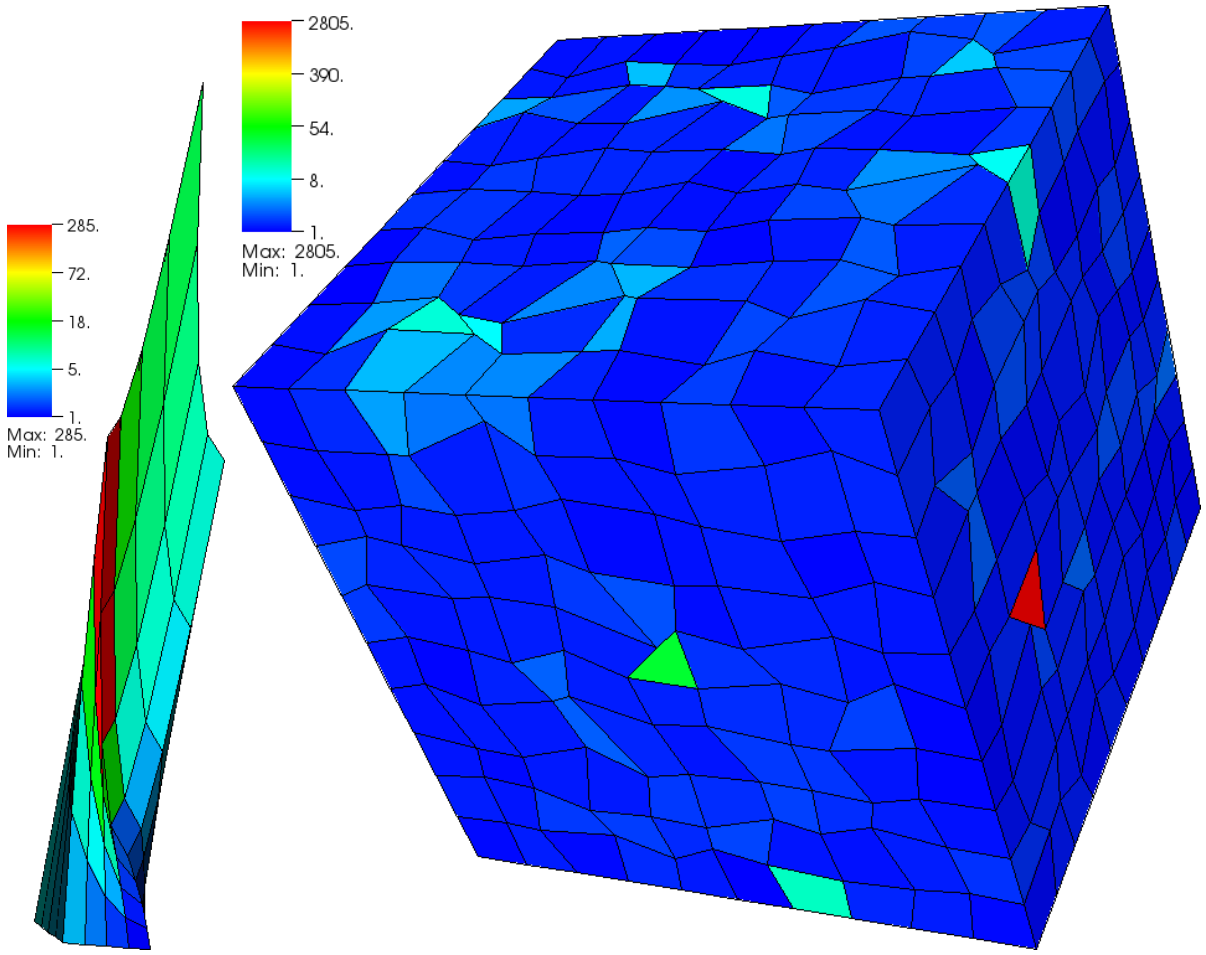

Figure 2.4: Two all-hexahedral meshes, `twist` and `random`, containing nearly degenerate elements (element condition number shown).

Poor-quality meshes

Figure 2.4 shows two low-quality meshes, `twist` and `random`. The first was generated by stretching and twisting a mesh of the reference cube and the second generated using the "random" feature of Cubit (Blacker et al., 1994). Additionally, we consider `flat`, a uniform Cartesian mesh with dimensions $1 \times 1 \times 10^{-5}$. These should be compared to `brick`, an 8^3 mesh of the reference cube.

While the approximation properties of these meshes are poor, the preconditioner is still effective provided a good preconditioner for the low-order system is available. The iteration count to oversolve the second Newton iteration to a relative tolerance of 10^{-8} for a $\mathfrak{p} = 1.5$ case is shown in Table 2.2. With the exception of ML on `twist` and `flat`, the mesh has a limited impact on performance.

Anisotropic meshes such as `flat` require semi-coarsening and/or line smoothers for multigrid performance. Convergence on this mesh is poor with both algebraic multigrid packages. BoomerAMG allows little flexibility in smoothers, and although some semi-coarsening was evident in the hierarchy, coarse spaces suitable for the builtin smoothers were not produced. ML's coarse spaces were less precisely semi-coarsened, and iteration counts could only be marginally controlled through the use of very strong smoothers (8-block overlap-1 additive Schwarz with direct subdomain solves).

Mesh	`brick`		`twist`		`random`		`flat`	
Element	ML	BMG	ML	BMG	ML	BMG	ML	Chol
Q_1	4	4	4	4	8	–	4	1
Q_2	24	24	25	23	27	27	29	26
Q_3	24	24	29	27	28	27	38	34
Q_4	29	28	39	30	34	33	47	37
Q_5	35	27	47	34	42	35	58	40
Q_6	35	29	60	40	43	41	80	44

Table 2.2: Iteration counts for four different meshes using ML, BoomerAMG (BMG), and Cholesky (Chol) preconditioning. For Q_1 `random`, BoomerAMG failed, producing NaN. ML on the `flat` mesh was run with stronger smoothers as described in the text. Note that the problem size increases with element order.

Nonlinearities

For strongly nonlinear problems, the majority of the solve time is spent in the pre-asymptotic range. Since a high-accuracy solve is not needed, it is very important to consider matrix assembly and preconditioner setup time when developing an efficient solver. We consider the $\mathfrak{p} = 1.2, \varepsilon = 10^{-8}$ case with an initial guess of zero (the state at which the system is most nearly singular) which requires a fair number of Newton steps. Table 2.3 shows the time spent in various stages of the solve for 20^3 Q_3 elements (226981 degrees of freedom). In the most efficient configuration, roughly one third of the time is spent in matrix-free Jacobian application with a similar amount spent in assembly. This problem demonstrates a situation where low-accuracy linear solves slow the nonlinear convergence, but it is still not cost effective to perform high accuracy linear solves. Lagging the preconditioner was not found to be effective for this problem, therefore only limited returns would be provided by a method with higher assembly costs, even if that setup enabled the use of a stronger preconditioner. We have computed a fourth order accurate solution with essentially the same memory requirements and modest cost increase over a second order scheme (Q_1). This solution is less expensive in both space and time than a conventional third order (Q_2) scheme, the reduced assembly costs offer more flexibility in nonlinear solver, and the sparser matrix offers more preconditioning choices.

2.4.2 Stokes

The weak form of the Dirichlet Stokes problem is: find $(u,p) \in \boldsymbol{V}_D \times P$ such that

$$\int_\Omega \eta Dv : Du - p\nabla\cdot v - q\nabla\cdot u - f\cdot v = 0$$

for all $(v,q) \in \boldsymbol{V}_0 \times P$ where $Du = \frac{1}{2}\left(\nabla u + (\nabla u)^T\right)$ is the symmetric gradient, $\boldsymbol{V}_D = \boldsymbol{H}^1_D(\Omega)$ is the inhomogeneous velocity space with $\boldsymbol{V}_0$ the corresponding homogeneous space, and $P = \{p \in L^2_0(\Omega) : \int_\Omega p = 0\}$ is the pressure space. Stability requires satisfaction of the discrete inf-sup condition

$$\inf_p \sup_u \frac{\int_\Omega p\nabla\cdot u}{\|p\|_0 \|u\|_1} \geq \beta > 0.$$

The finite element space $Q_k - Q_{k-2}$ is stable with $\beta \in O(k^{-(d-1)/2})$ in d dimensions (see Schötzau et al., 1998) and is thus quite usable for the modest orders we propose.

Indefinite preconditioning

Standard preconditioners perform poorly or fail completely when applied to indefinite problems. Block factorization provides a general framework for constructing effective preconditioners for the Stokes

Tolerance	10^{-4} relative tolerance		Eisenstat-Walker	
Event	ILU(0)	ILU(1)	ILU(0)	ILU(1)
Residual	35	29	57	51
Assembly	78	68	127	110
Krylov	295	274	187	166
MF MatMult	259	190	155	110
PCSetup	2	11	5	17
PCApply	26	67	27	39
Total time	413	374	377	333
Newton #	15	13	24	21
Residual #	25	20	40	36
Krylov #	911	667	545	386

Table 2.3: Time (seconds) spent in various stages of a nonlinear solve for Q_3 elements preconditioned by ILU(0) and ILU(1) applied to the corresponding Q_1 matrix. The first columns solve the linear system to a relative tolerance of 10^{-4} with a maximum of 60 Krylov iterations per Newton, the latter use the Eisenstat-Walker(1996) method for adjusting solver tolerances with a maximum of 30 Krylov iterations. The events *MF MatMult* (matrix-free Jacobian application), *PCSetup*, and *PCApply* are part of the *Krylov* iteration. Additional *Residual* evaluations are needed when the line search is activated.

problem. They are based on factoring the Jacobian as

$$J(u) = \begin{bmatrix} A(u) & B^T \\ B & \end{bmatrix} = \begin{bmatrix} 1 & \\ BA^{-1} & 1 \end{bmatrix} \begin{bmatrix} A & \\ & S \end{bmatrix} \begin{bmatrix} 1 & A^{-1}B^T \\ & 1 \end{bmatrix} \tag{2.4.2}$$

where $S = -BA^{-1}B^T$ is the Schur complement which is dense and must be preconditioned by other means. When GMRES is used with left (right) preconditioning, the upper (lower) block is typically dropped (since the resulting exactly preconditioned operator has minimal degree 2, see Murphy et al. (2000)), and all occurrences of A^{-1} are replaced by a suitable preconditioner. There are numerous ways to precondition S (e.g. Benzi et al., 2005; Elman et al., 2008), here we use only the classic pressure mass matrix M_p, but more sophisticated methods are needed for strongly heterogeneous or anisotropic problems (see May and Moresi, 2008), for Navier-Stokes with non-vanishing Reynolds number, and for short time steps in a time-dependent simulation. With the dual-order method, it is only necessary to assemble matrices to precondition A and S (the latter via a diagonal approximation to the mass matrix M_p), all "matrix multiplies" are performed matrix-free.

For the Dirichlet Stokes problem, constant pressure is in the null space of J and S. This does not impact solver performance as long as the right hand side is consistent and the solver removes the null space from the Krylov basis. Table 2.4 shows iteration counts for a variety of elements and meshes using right-preconditioned GMRES and the right-triangular preconditioner resulting from the factorization (2.4.2) with S^{-1} approximated by the diagonal of M_p and A^{-1} approximated by one V-cycle of ML with inter-component coupling dropped. The iteration count is scalable with resolution, but deteriorates much more rapidly with element order than for the definite problem. The corresponding solve times, shown in Figure 2.5, confirm the asymptotics under h-refinement with each element type. More sophisticated preconditioners for S were able to reduce the iteration count, but did not reliably reduce solve time across the range of element orders. Further investigation of indefinite preconditioner performance under p-refinement is needed.

	$Q_3 - Q_1$		$Q_5 - Q_3$		$Q_7 - Q_5$	
Mesh	dofs	its	dofs	its	dofs	its
4^3	4118	38	22774	79	68310	92
8^3	37230	38	193582	75	568046	92
12^3	130822	38	666790	76	1942342	95
16^3	316382	40	1596766	77		
20^3	625398	40				
24^3	1089358	40				
30^3	2144698	41				

Table 2.4: Problem size and Krylov iterations to solve the Stokes problem to a relative tolerance of 10^{-6} with elements of different orders using right-preconditioned GMRES(30) and field-split ML on the A block. Each restart caused an approximately 2-iteration stall.

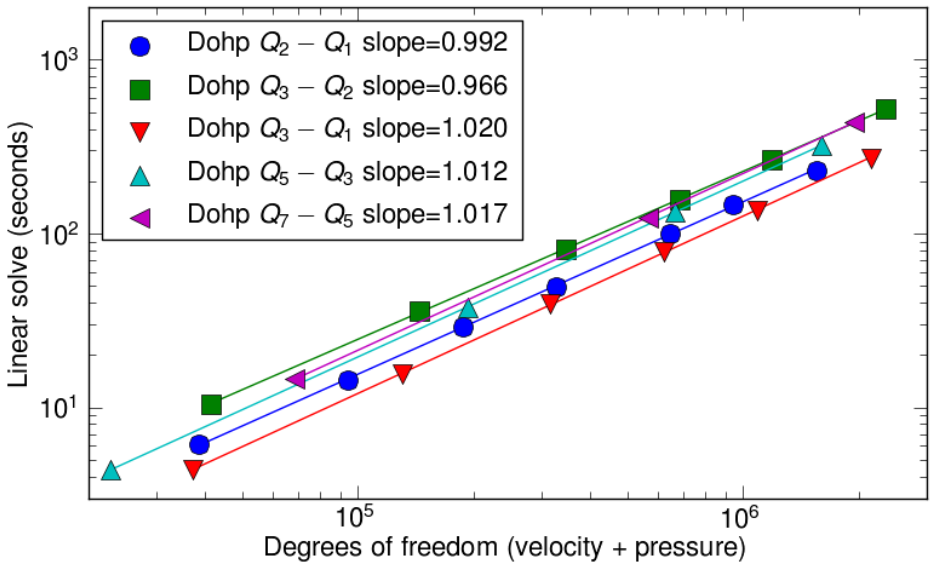

Figure 2.5: Linear solve time for 3D Stokes with relative tolerance of 10^{-6}. For $Q_2 - Q_1$ and $Q_3 - Q_2$ elements, convergence is significantly slower than with $Q_k - Q_{k-2}$, but apparently scalable despite being somewhat erratic.

2.5 Discussion

We have presented a practical method of obtaining high-order accuracy with cost similar to conventional methods for lower order accuracy. It is robust on highly deformed meshes and nonlinear problems provided an effective preconditioner is available for the associated Q_1 matrix. The computational kernels remove significant pressure on the memory bus enabling more effective use of floating point units, especially in multicore environments.

The extra structure imposed by high-order methods provides natural coarsening which suggests the use of one or more levels of geometric multigrid. This avoids the cost of computing interpolation operators in algebraic multigrid and enables the use of rediscretized coarse operators which preserve sparsity in comparison to Galerkin operators. Integration with PETSc's geometric multigrid framework is underway.

Since solve time is only weakly dependent on element order, the dual-order scheme is well-suited for use in an *hp*-adaptive simulation where existing refinement strategies for minimizing degrees of freedom will more accurately represent computational cost. Relative to conventional Q_2 and higher elements, the dual-order scheme significantly reduces the cost of matrix assembly and preconditioner setup, while maintaining competitive iteration counts when used with preconditioners such as algebraic multigrid.

Chapter 3

Textbook multigrid efficiency for hydrostatic ice sheet flow

This chapter is in review at the SIAM Journal of Scientific Computing: Brown, J., Smith, B., and Ahmadia, A. (2011) Achieving textbook multigrid efficiency for hydrostatic ice sheet flow.

Abstract. The hydrostatic equations for ice sheet flow offer improved fidelity compared to the shallow ice approximation and shallow stream approximation (SSA) popular in today's ice sheet models. Nevertheless, they present a serious bottleneck because they require the solution of a 3D nonlinear system, as opposed to the 2D system present in SSA. This 3D system is posed on high-aspect domains with strong anisotropy and variation in coefficients, making it expensive to solve using current methods. This paper presents a Newton-Krylov multigrid solver for the hydrostatic equations that demonstrates textbook multigrid efficiency (an order of magnitude reduction in residual per iteration and solution of the fine-level system at a small multiple of the cost of a residual evaluation). Scalability on Blue Gene/P is demonstrated, and the method is compared to various algebraic methods that are in use or have been proposed as viable approaches.

3.1 Introduction

The dynamic response of ice streams and outlet glaciers is poorly represented using the shallowness assumptions inherent in the present generation of ice sheet models. Accurate simulation of this response is crucial for prediction of sea level rise; indeed, the inability of available models, based on the shallow ice approximation (SIA) (Hutter, 1983) and shallow stream approximation (SSA) (Morland, 1987; Weis et al., 1999), to simulate these processes was cited as a major deficiency in the Fourth Assesment Report of the Intergovernmental Panel on Climate Change (Bernstein et al., 2008).

The hydrostatic equations were introduced by Blatter (1995) as a model of intermediate complexity between the full non-Newtonian Stokes system and the vertically-integrated SIA and SSA models. A more precise analysis, including the limiting cases of fast and slow sliding, was given in Schoof and Hindmarsh (2010). Well-posedness was proven in Colinge and Rappaz (1999), and approximation properties of finite-element methods were analyzed in Glowinski and Rappaz (2003); Chow et al. (2004). Hindmarsh (2004) conducted a perturbation analysis in one horizontal dimension to study validity of different continuum models including SIA, hydrostatic, and Stokes. Pattyn et al. (2008) contains a model intercomparison using several implementations of the hydrostatic equations for periodic benchmark problems with large amplitude basal topography and stickyness variability. The hydrostatic equations were used for transient simulation in Pattyn (2002) and in 3D models in Pattyn (2003), as well as subsequent work. These equations have been known by a variety of names in the literature including "Blatter", "Pattyn", "Blatter-Pattyn", "LMLa" (Hindmarsh, 2004), and "first order" (Greve and Blatter, 2009).

The use of hydrostatic equations in current models has been limited, however, by the cost of solving the 3D nonlinear system for velocity. This cost comes from both slow convergence on the nonlinearities (rheology and slip) and expensive linear solves using standard preconditioners such as incomplete factorization and one-level domain decomposition. The poor linear solve performance is attributable to the strong anisotropy and heterogeneity imposed by the rheology and geometry.

In the present work, we introduce a Newton-Krylov multigrid solver that demonstrates textbook multigrid efficiency, characterized by convergence in a small multiple of the cost of a single, fine-level residual evaluation, and typically involving an order of magnitude reduction in residual per multigrid (V or F) cycle. The scheme converges quadratically on the nonlinearities, is rapidly globalized by using grid sequencing, is robust to parameters and geometry, coarsens rapidly in almost all cases, and exhibits excellent parallel scalability. Our code is freely available as part of the Portable Extensible Toolkit for Scientific computing (PETSc) (Balay et al., 2012b).

Section 3.2 presents the equations and discretization, Section 3.3 describes the solver, and Section 3.4 demonstrates performance and scalability with numerical examples. Section 5 summarizes our conclusions.

3.2 Equations and Discretization

The hydrostatic equations are obtained from the non-Newtonian Stokes equations in the limit where horizontal derivatives of vertical velocity are small. Neglecting these terms allows incompressibility to be enforced implicitly by eliminating pressure and vertical velocity, leaving a system involving only horizontal components of velocity. See Schoof and Hindmarsh (2010) for a rigorous derivation and asymptotic analysis.

Consider an ice domain $\Omega \subset \mathbb{R}^3$ lying between a Lipschitz continuous bed $b(x,y)$ and surface $s(x,y)$ with thickness $h = s - b$ bounded below by a positive constant. [1] The velocity $\boldsymbol{u} = (u,v) \in \mathcal{V} = \boldsymbol{W}^{1,1+1/n}(\Omega)$ satisfies conservation of momentum, which, omitting inertial and convective terms as is standard for ice sheets, is given by

$$-\nabla\cdot\left[\eta\begin{pmatrix} 4u_x+2v_y & u_y+v_x & u_z \\ u_y+v_x & 2u_x+4v_y & v_z \end{pmatrix}\right] + \rho g \nabla s = 0, \tag{3.2.1}$$

where

$$\eta(\gamma) = \frac{B}{2}\left(\varepsilon^2/2+\gamma\right)^{\frac{1-n}{2n}} \tag{3.2.2}$$

is the nonlinear effective viscosity with regularizing strain rate ε and

$$\gamma = u_x^2 + v_y^2 + u_x v_y + \frac{1}{4}(u_y+v_x)^2 + \frac{1}{4}u_z^2 + \frac{1}{4}v_z^2$$

is the second invariant. Ice sheet models typically take $n = 3$ as the power law exponent. Equation (3.2.1) is subject to natural boundary conditions at the free surface and either no-slip $\boldsymbol{u} = 0$ or power-law slip with friction parameter

$$\beta^2(\gamma_b) = \beta_0^2\left(\varepsilon_b^2/2+\gamma_b\right)^{\frac{m-1}{2}},$$

where $\gamma_b = \frac{1}{2}(u^2+v^2)$, ε_b is regularizing velocity, and $m \in (0,1]$ is the exponent that produces Navier slip for $m = 1$, Weertman (Weertman, 1957) sliding for $m = 1/n$, and Coulomb slip as $m \to 0$. In the present work, we define ε and ε_b using a strain rate of $10^{-5}\,\mathrm{a}^{-1}$ and a slip velocity of $1\,\mathrm{m\,a}^{-1}$, respectively.

[1] The singular limit $h \to 0$ is important in the case of grounded margins, but the present work does not pursue it.

To discretize this system with a finite-element method, we introduce the test functions $\phi \in \mathcal{V}$ and integrate by parts to produce the weak form: Find $\boldsymbol{u} \in \mathcal{V}$ such that

$$\int_\Omega \nabla\phi : \eta \mathbf{1} : \begin{pmatrix} 4u_x + 2v_y & u_y + v_x & u_z \\ u_y + v_x & 2u_x + 4v_y & v_z \end{pmatrix} + \phi \cdot \rho g \nabla s + \int_{\Gamma_{\text{bed}}} \phi \cdot \beta^2 (|\boldsymbol{u}|^2/2)\boldsymbol{u} = 0 \tag{3.2.3}$$

for all $\phi \in \mathcal{V}$, where Γ_{bed} is the slip portion of the bed.

For our numerical studies, equation (3.2.3) was discretized on a topologically structured hexahedral grid using Q_1 finite elements and standard 2^3-point Gauss quadrature. Length, time, and mass units were chosen so that thickness, velocity, and driving stresses are of order 1. See Section 3.3.2 for details on the enforcement of Dirichlet boundary conditions.

The source code for our implementation is distributed as an example in PETSc (Balay et al., 2012b) versions 3.1 and later. [2] Several generalizations of the tests from ISMIP-HOM (Pattyn et al., 2008) are implemented, run with the option `-help` for a complete list of options. Incidentally, our results for "5 km test *C*" in that paper (variable slipperiness on a flat bed) agree to three significant figures with the "Stokes" results therein. This is consistent with the asympotic analysis of Schoof and Hindmarsh (2010), which shows that slipperiness perturbations are not present in the leading-order error terms for the hydrostatic equations (which are purely geometric); c.f. Pattyn et al. (2008, Table 4 and Figure 8), where the ensemble range is nearly as large as the mean.

3.3 Solver and Implementation

We begin by writing the discretization of (3.2.3) as an algebraic system $F(U) = 0$ with Jacobian $J(U)$. This nonlinear system is solved with a Newton iteration that requires an approximate solution δU of

$$J(U)\delta U = -F(U). \tag{3.3.1}$$

Newton methods are quadratically convergent in the terminal phase but may converge slowly or not at all in early phases. Many applications of the present solver are in a time-stepping code where the initial iterates start within the region of quadratic convergence; thus globalization would rarely be a concern. But since a good initial iterate is not available in the present tests, we use grid sequencing (solving the problem on a sequence of coarser grids) to produce an initial iterate on the fine grid. Globalization is also a critical issue when solving steady-state problems. Grid sequencing requires a geometric hierarchy of meshes with interpolation operators to move the solution to the next finer level. Managing this hierarchy is often seen as a programming burden, but it exposes more robust algorithms than are available otherwise.

The Newton step (3.3.1) is solved by a Krylov method such as GMRES, for which the iteration count is highly dependent on the quality of the preconditioner. Since $J(U)$ is symmetric positive definite (SPD), methods such as conjugate gradients could be used. This work always uses GMRES, however, because it allows the use of nonsymmetric preconditioners, and the iteration counts are always kept low so that storage and restarts are not an issue. As an SPD system, it has a wide variety of preconditioners to choose from; however, viscosity contrasts and strong anisotropy cause most preconditioners to perform poorly. The rest of this section describes the methods used to produce a scalable algorithm in spite of these difficulties.

3.3.1 Anisotropic Meshes

The ratio of width to thickness for outlet glaciers (the regions of ice sheets with greatest physical interest) ranges from order 1 to over 100. The nonlinear constitutive relation (3.2.2) produces three to four orders of

[2] Upon unpacking the source, which can be downloaded from `http://mcs.anl.gov/petsc`, see `src/snes/examples/tutorials/ex48.c`.

magnitude variation in viscosity (usually with fastest variation in the vertical) and the Newton linearization of (3.2.3) produces additional anisotropy, effectively collapsing the conductivity tensor in the direction of the velocity gradient.

For systems with a priori known anisotropy, semi-coarsening has been successful for attaining satisfactory multigrid performance even with weak smoothers like SOR, but semi-coarsening is unattractive for two reasons. First, semi-coarsening in the vertical direction would necessitate many levels because it reduces the problem size by only a factor of 2 on each coarsening (instead of 8 for isotropic coarsening), and the fully coarsened problem would still be far too large for a direct solver, necessitating further coarsening in the horizontal direction. The presence of many levels leads to more synchronization in parallel, which is detrimental to scalability and makes performance more sensitive to network latency. Second, viscosity contrasts and anisotropy unaligned with the grid arise when the friction parameter β^2 is not smooth, as is the standard case when studying the migration of ice stream margins as the bed transitions from frozen (no-slip or very high friction) to temperate and very slippery depending on subglacial hydrology.

To coarsen the system isotropically even on high-aspect domains, we order the unknowns so that columns are contiguous with a natural block size of 2 (i.e. $\{u_{i,j,k}, v_{i,j,k}, u_{i,j,k+1}, v_{i,j,k+1}, \dots\}$ where k is the index that is increasing in the vertical direction) and not decomposed in parallel. This decomposition is reasonable since the number of vertical levels used in simulations is typically between 10 and 100; it also is convenient since it is compatible with decompositions used by other climate model components.

With this ordering, zero-fill incomplete factorization effectively performs an exact solve of the column since all the neglected fill (relative to an exact factorization) is caused by the coupling with adjacent columns. Pure line smoothers were also tried as a smoother on the finest level, but robustness was significantly impacted, and the memory benefits were deemed insufficient to pursue further.

In scenarios with little sliding and gentle geometry compared to the ice thickness, the "shallow ice approximation" (SIA) is valid, allowing the velocity field to be determined locally from the surface slope and a column integral. If horizontal derivatives were discarded from the linearized hydrostatic equations, the corresponding matrix would have only uncoupled tridiagonal systems for each column and the solution of these systems would be equivalent to linearized SIA. These tridiagonal systems become singular as basal friction $\beta^2 \to 0$, and furthermore, SIA is physically inconsistent if β^2 is discontinuous even when it is bounded below by a sufficiently large constant due to discontinuous velocities (Fowler, 2001). The contribution to the hydrostatic matrix from discrete horizontal derivatives is small (order h_z^2/h_x^2) relative to that from vertical derivatives when the element aspect ratio (h_z/h_x) is small, and therefore high energy modes of the hydrostatic matrix, corresponding to velocity variation primarily in the vertical, become well approximated by the block tridiagonal system representing SIA. When the bed is very slippery, there are medium to long wavelength modes with vanishing energy in the norm induced by the (nearly singular) SIA equations, but non-vanishing energy in the norm induced by the hydrostatic equations. These modes are problematic if the tridiagonal SIA system is used to precondition hydrostatic because it will take many iterations for the Krylov solver to locate all of these modes not controlled by the preconditioner.

Incomplete factorization with column ordering provides nearly exact coupling in the vertical, effectively containing the tridiagonal SIA plus some coupling with nearby columns, and is an effective stand-alone preconditioner when the bed is sticky and horizontal grid spacing is large, despite lack of a coarse level. Indeed, with a typical ice thickness of 1 km resting on a frozen bed and elements 5 km on a side, block Jacobi with zero-fill incomplete Cholesky converges to relative tolerance of 10^{-5} in about 10 Krylov iterations independent of the horizontal extent of the domain (number of elements in the horizontal), independent of the number of elements in the vertical, and independent of the number of subdomains (provided they do not get too small: there is some degradation when subdomain size approaches a single column). However, none of these favorable performance characteristics remain when the elements become small relative to the ice thickness or when the bed becomes slippery since the usual $O\big((L/H)^2\big)$ condition number for second-order elliptic problems preconditioned by one-level Schwarz methods with subdomains of size H, see Smith et al. (1996), becomes apparent. Indeed, we have found low-fill incomplete factorization to be nearly unusable as part of a one-level additive Schwarz method for problems

with slippery beds or steep geometry, even at low resolution, as investigated in Section 3.4.3.

3.3.2 Dirichlet Boundary Conditions

Multigrid is often sensitive to the enforcement of boundary conditions. Ideally, Dirichlet conditions would be completely removed from the solution space, but doing so complicates grid management on structured grids, so instead we leave these degrees of freedom in the system but decouple them from the other equations. During residual evaluation in the finite-element context, this strategy corresponds to evaluating integrals with the Dirichlet condition satisfied exactly and setting the residual on the Dirichlet nodes to be equal to a multiple of the current solution. With this scheme, all rows and columns of the Jacobian corresponding to Dirichlet nodes are zero except for a single diagonal entry. Thus the system retains symmetry, and satisfaction of the Dirichlet conditions does not interfere with solving the other equations. For good multigrid performance, the diagonal entry should be similar to the diagonal entry of the Jacobian for nearby nodes. To ensure this, we set the residual at Dirichlet nodes to

$$\begin{aligned} f_u &= 2\eta h_x h_y h_z (4/h_x^2 + 1/h_y^2 + 1/h_z^2) u \\ f_v &= 2\eta h_x h_y h_z (1/h_x^2 + 4/h_y^2 + 1/h_z^2) v, \end{aligned} \tag{3.3.2}$$

where h_x, h_y, h_z are the local element dimensions. This scaling produces the same diagonal entries that would appear if the domain was extended so that constant viscosity momentum equations appeared at the formerly Dirichlet nodes.

3.3.3 Matrices

The most expensive operations are Jacobian assembly and sparse matrix kernels such as matrix-vector multiplication and solves with incomplete triangular factors. The former involves evaluation of fractional powers and finite element quadrature loops. While fractional powers take most of the time for residual evaluation, they are less significant than quadrature loops for assembly. The quadrature loops were explicitly vectorized by using SSE2 intrinsics which led to a 30% speedup on Core 2 and Opteron architectures using both GNU and Intel compilers. There was no manual vectorization for the Blue Gene/P results quoted in Section 3.4.2.

Assembly costs could be further mitigated by recomputing the matrix less frequently, either by using a modified Newton method (degrades nonlinear convergence rate) or by applying the current operator matrix-free by finite differencing the residual or using automatic differentiation in which case only the preconditioner is lagged. These are runtime options in the present code; but since they do not offer a clear benefit they have not been pursued in the present work. If matrix-free application of the true Jacobian is used, several other preconditioning options become available without impacting the nonlinear convergence rate. One could assemble only the block-tridiagonal column coupling, ignoring horizontal coupling, thus saving the memory for the finest level(s). Additionally, a truly 2D coarse problem can be defined by using the shallow stream equations (Morland, 1987; Weis et al., 1999; Schoof, 2006a) and restriction operators defined by integrating the entire column. These possibilities are also runtime options, but have not exhibited a level of robustness comparable to that of the more conventional methods pursued here.

The Jacobian is always symmetric positive definite and has a natural block size of 2, so we use a symmetric block format (PETSc's SBAIJ). This format stores one column index per 2×2 block in the upper triangular part of the matrix and therefore uses about half the storage of the nonsymmetric BAIJ format, which in turn uses 25% less memory than a scalar format (AIJ). Multiplication for symmetric storage requires twice as much parallel communication as nonsymmetric storage, albeit with the same number of messages. In return, the diagonal part of a parallel decomposition does twice as much work per matrix entry and thus achieves higher throughput, as shown in Table 3.1.

There are two ways to construct matrices on the coarse levels of multigrid methods. The first, which we use in almost all our numerical examples, is to rediscretize the system on the coarse mesh. In

Table 3.1: Throughput (Mflop/s) for different matrix formats on Core 2 Duo (P8700) and Opteron 2356 (two sockets). `MatSolve` is a forward- and back-solve with incomplete Cholesky factors. The AIJ format is using "inodes" which unrolls across consecutive rows with identical nonzero pattern (pairs in this case).

Kernel	Core 2, 1 process			Opteron, 4 processes		
	AIJ	BAIJ	SBAIJ	AIJ	BAIJ	SBAIJ
`MatMult`	812	985	1507	2226	2918	3119
`MatSolve`	718	957	955	1573	2869	2858

our implementation, this involves re-evaluating nonlinearities on each level of the hierarchy, although restricting fine-level coefficients of the linearized problem would also be possible. This procedure produces coarse operators that are as sparse as possible on each of the levels. The Galerkin procedure is an alternative that is mandatory for algebraic multigrid. Given an interpolation operator $P : \mathcal{V}_{\text{coarse}} \to \mathcal{V}_{\text{fine}}$ and assembled fine-level matrix A_{fine}, the Galerkin coarse operator is $A_{\text{coarse}} = P^T A_{\text{fine}} P$. These operators work well for some problems, but computing the sparse matrix product in parallel involves significant communication and irregular memory access, so it is relatively expensive. Additionally, second-order-accurate interpolation operators cause a loss of sparsity in the coarse-level operators, an effect known as stencil growth. Stencil growth tends to blur regions where the solution has local structure and usually reduces the effectiveness of inexpensive smoothers.

3.4 Numerical Examples

We present several numerical examples that demonstrate the algorithmic and parallel scalability of the Newton-Krylov multigrid approach.

3.4.1 Algorithmic Scalability

We consider three model problems inspired by the periodic domain ISMIP-HOM (Pattyn et al., 2008) tests. All use surface $s(x,y) = -x\sin\alpha$, where α is the surface slope (the coordinate system is not rotated) and a bed similar to $b_A(x,y) = s(x,y) - 1000\,\text{m} + 500\,\text{m}\cdot\sin\hat{x}\sin\hat{y}$ for $(x,y) \in [0,L)^2$ with $\hat{x} = 2\pi x/L, \hat{y} = 2\pi y/L$. Test X uses bed $b_X = b_A$ and stickiness parameter

$$\beta_X^2(x,y) = \begin{cases} 2000\,\text{Pa}\,\text{a}\,\text{m}^{-1}, & \text{if } r = |(\hat{x},\hat{y}) - (\pi,\pi)| < 1 \\ 0, & \text{otherwise} \end{cases}$$

which is free slip except for a sticky circle at the center of the domain, which is not aligned with the grid. A solution cutout for $L = 10\,\text{km}$ is shown in Figure 3.1. This problem exhibits shear localization at the edges of the sticky region and is most extreme at high aspect ratio. We choose $L = 80\,\text{km}$ and $\alpha = 0.05^\circ$ which produce velocities from $0.9\,\text{km}\,\text{a}^{-1}$ to $47\,\text{km}\,\text{a}^{-1}$. [3] To demonstrate algorithmic scalability as the problem size is increased, this test was run on 8 processors starting from a coarse grid of $16\times16\times1$, refining twice in the horizontal by factors of 2 in both x and y, then three times in the vertical by factors of 8 each to reach a fine mesh of $64\times64\times513$, which has elements of nominal dimension $1250\times1250\times1.95$ meters.

A visual representation of the nonlinear solve process is shown in Figure 3.2. In this example and the next one, Luis Chacón's variant of the Eisenstat-Walker (Eisenstat and Walker, 1996) method was used to automatically adjust linear solve tolerances as the nonlinear solve converges. Solving the linear systems to higher tolerance would have little impact on the number of nonlinear iterations and would be visible in

[3] This problem may be run with the options `-thi_hom X -thi_L 80e3 -thi_alpha 0.05`, the other cases can be selected with similar options.

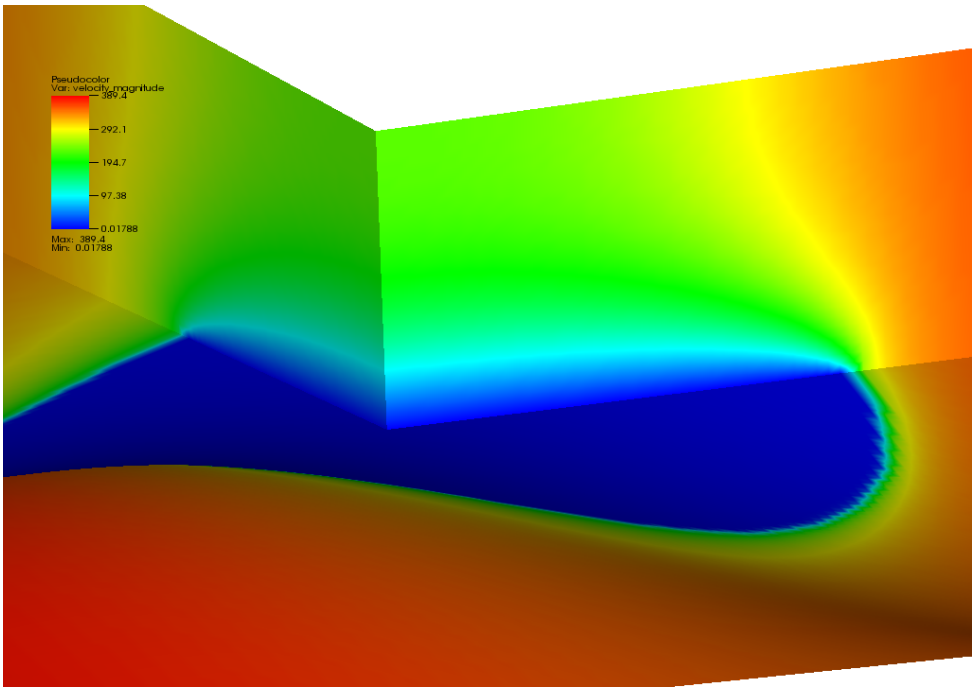

Figure 3.1: A cutout colored by velocity magnitude for flow over the bumpy bed of test X at $L = 10\,\text{km}$ with $m = 1/10$ nearly plastic basal yield model. The cut on the left shows along-flow velocity as the ice hits the sticky region, the cut on the right shows across-flow shear structure.

the form of more $\times$ marks below the solid lines. That most $\times$ marks lie on the solid line for nonlinear residual is an indication that effort is well balanced between linear and nonlinear solves. Note that approximately 10 linear V-cycles on the fine level are required to reduce the residual by 10 orders of magnitude. We remark that Picard iteration takes at least 50 iterations to reach this tolerance (cf. De Smedt et al. (2010) in which hundreds or thousands of iterations were needed for an easier problem). When used in a time-stepping code, the solution at the last time step is available so typically only the terminal convergence rate is important which further favors Newton methods. For example, an initial guess might have a residual 10^3 higher than the required tolerance which takes 15 Picard iterations to solve, but only one Newton iteration. Additionally, each linear solve for this fine-level problem requires hundreds or thousands of iterations with a one-level additive Schwarz method; see Section 3.4.3, which considers a smaller problem.

Test Y places a 200 m tower with vertical walls on the top of each bedrock bump and uses an uncorrelated but smoothly varying stickiness resembling a dimpled sombrero:

$$b_Y(x,y) = \begin{cases} b_A(x,y), & \text{if } b_A(x,y) < -700\,\text{m} \\ b_A(x,y) + 200\,\text{m}, & \text{otherwise} \end{cases}$$

$$\beta_Y^2(x,y) = 1000\,\text{Pa a m}^{-1} \cdot (1 + \sin(\sqrt{16r})/\sqrt{10^{-2} + 16r}\,\cos\frac{3\hat{x}}{2}\cos\frac{3\hat{y}}{2}.$$

This tests the quality of the coarse grids even when large geometric errors are committed. Note that the hydrostatic equations cannot be considered valid in this regime since the topography is too abrupt. Such topography is present in reality, however, so we may still desire an efficient solver. Figure 3.3 depicts the solve for this problem in a 10 km square domain. Because of the successively better resolution of the "cliff," performance deteriorates on each level, as can be seen by the closer spacing of linear solve marks ($\times$). It is entirely acceptable up to level 3, however, where the elements are approximately 12 m thick and stretch to reach over a 200 m vertical cliff in 125 m horizontal, a slope of 58°. On the finest level, they are 6 m thick and stretch over the cliff in 62 m horizontal, a slope of 73°. The approximation properties of

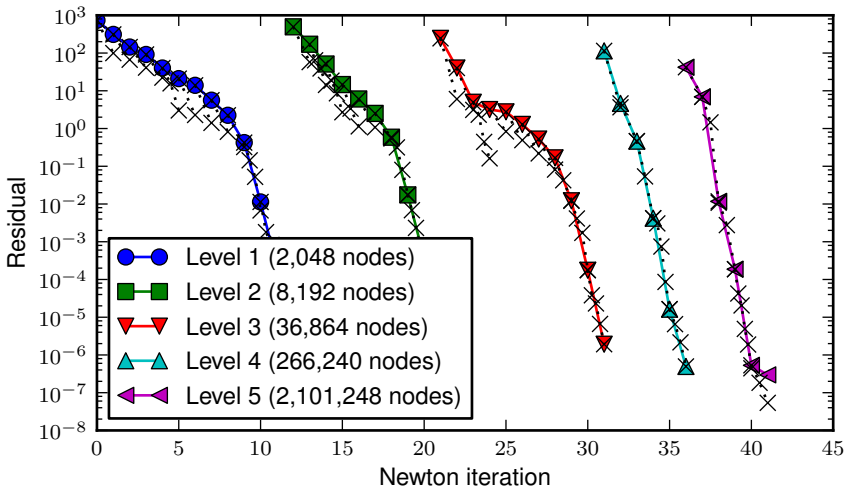

Figure 3.2: Grid-sequenced Newton-Krylov solution of test X. The solid lines denote nonlinear iterations, and the dotted lines with $\times$ denote linear residuals.

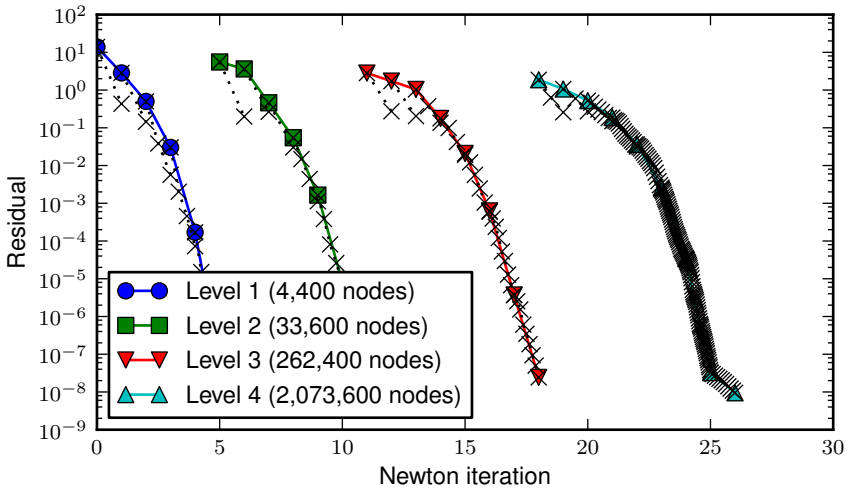

Figure 3.3: Grid sequenced Newton-Krylov convergence for test Y.

such elements are poor and, considering that the continuum equations are invalid here, we believe this resolved topography is significantly rougher than will needed in applications.

Test Z sets $b_Z = b_A$, $\beta_Z^2 = \beta_X^2$, and nonlinear sliding with exponent $m = 0.3$. It is a regime where the hydrostatic equations are valid, provided the wavelength L is not too small. We use this case to explore linear solve performance in Figure 3.4.

3.4.2 Parallel Scalability on Blue Gene/P

Two types of parallel scalability are considered. Strong scalability is the ability to solve a fixed problem size faster by using more processors. It is represented by a log-log plot of time versus number of processors where a slope of -1 is optimal. Weak scalability is the ability to solve larger problems in constant time by using more processors with the problem size per processor remaining fixed. It is represented by a plot of time versus number of processors where a slope of 0 is optimal. Strong scalability is important in transient simulations with many time steps while weak scalability is more important for attaining high

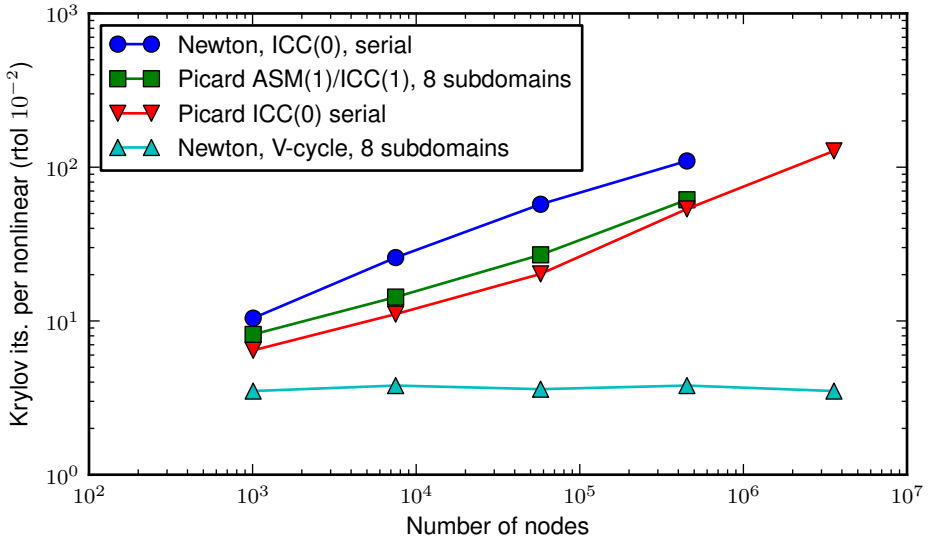

Figure 3.4: Average number of Krylov iterations per nonlinear iteration. Each nonlinear system was solved to a relative tolerance of 10^{-2}.

resolution in steady-state simulations or transient simulations with fewer time steps. Optimal strong and weak scalability together would imply that a problem with N elements could be solved in $O(N/P)$ time on P processors.

We investigate strong scaling using test Z at 80 km with basal friction exponent $m = 0.3$ on Shaheen, a Blue Gene/P at the KAUST Supercomputing Laboratory. Figure 3.5 shows strong scalability for four fixed global problem sizes with coarse meshes of $16 \times 16 \times 3$, $32 \times 32 \times 3$, $64 \times 64 \times 3$, and $64 \times 64 \times 1$. The first three use five levels of isotropic refinement to reach target meshes of $256 \times 256 \times 48$, $512 \times 512 \times 48$, and $1024 \times 1024 \times 48$, the latter with nominal element sizes of $78 \times 78 \times 21$ meters. The last coarse mesh is refined anisotropically and more aggressively to reach $1024 \times 1024 \times 64$ with only four levels. The smallest two coarse problems are solved redundantly and the second is also solved using the "XX^T" direct solver of Tufo and Fischer (2001) (TFS) which exhibits significantly better scalability but the coarse-level problem still presents a bottleneck. The largest two coarse problems are only solved approximately using a V-cycle of the algebraic multigrid package BoomerAMG (Henson and Yang, 2002), which removes the severe scalability bottleneck of a direct coarse-level solve. These coarse problems have been designed so that BoomerAMG is effective, see Section 3.4.3 for BoomerAMG applied to fine-level problems.

Figure 3.6 shows weak scalability. The size of the coarse grid was held constant, and additional levels were added as the number of processes was increased, such that the subdomain sizes remain approximately constant. As the resolution increases, the nonlinearity strength and linear stiffness also changes, explaining the relatively expensive solve with four processes. This effect is also apparent in the Level 3 convergence of Figure 3.2. Four phases of the solution process dominate the runtime and are identified in Figure 3.6. Multigrid setup costs account for less time than nonlinear residual evaluation for the larger problem sizes.

3.4.3 Algebraic Methods

Building a geometric hierarchy with rediscretization on coarse levels adds software complexity that many developers of numerical models do not want to deal with. In this section, we summarize the performance characteristics of several popular algebraic methods. We consider test X, $L = 80\,\text{km}, \alpha = 0.03$, with $40 \times 40 \times 12$ elements distributed over four processes. We compare our multigrid method with several one-level domain decomposition methods, two algebraic multigrids, and field-split approaches. This problem is challenging for the standard Newton iteration which requires 37 iterations and should be

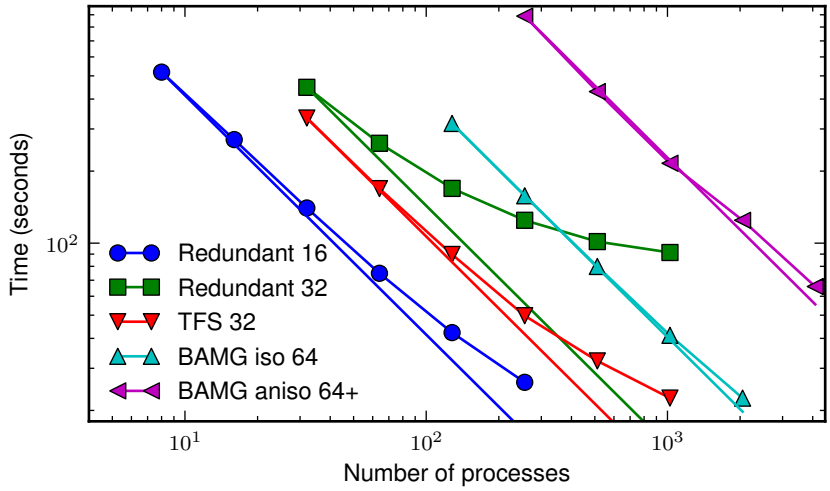

Figure 3.5: Strong scaling on Shaheen for different size coarse levels problems and different coarse level solvers (see text for details). The straight lines on the strong scaling plot have slope -1 which is optimal. Grid sequencing is used, but only the nonlinear solve on the finest level is shown since strong scalability is most important when many time steps are needed.

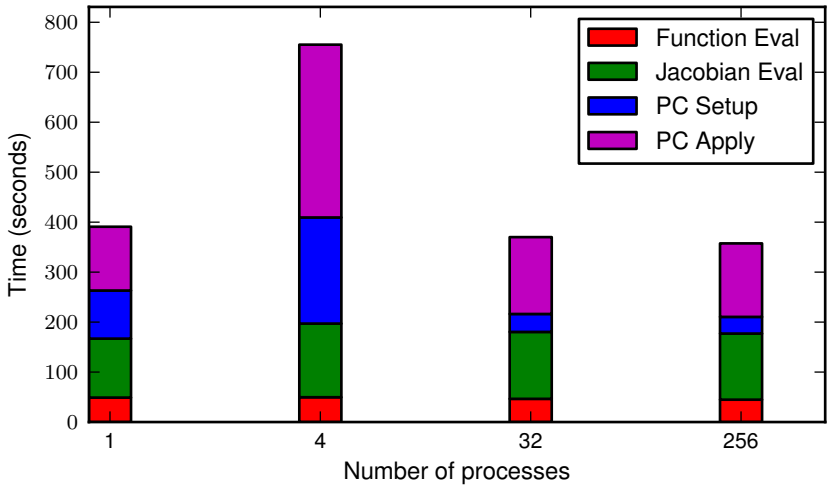

Figure 3.6: Weak scaling on Shaheen with a breakdown of time spent in different phases of the solution process. Times are for the full grid-sequenced problem instead of just the finest level solve.

Table 3.2: Average number of GMRES iterations per Newton iteration for one-level domain decomposition with different overlap and fill.

Decomposition	Subdomain solver			
	ICC(0)	ICC(1)	ICC(4)	Cholesky
Block Jacobi	367	315	220	97
ASM(1)	508	441	296	59
RASM(1)	368	306	190	52
ASM(2)	521	445	316	44
RASM(2)	365	305	189	38

accompanied by grid sequencing for efficiency. It takes the problem through a range of nonlinearities, however. Thus the number of Krylov iterations to solve with a relative tolerance of 10^{-5}, presented below, is a good test of the linear solver.

We first consider one-level domain decomposition methods with incomplete factorization, which are currently used to solve the hydrostatic equations by (Evans et al., 2010; Larour et al., 2010) among others. To keep iteration counts representative, we use full GMRES (no restart) with modified Gram-Schmidt orthogonalization (note that neither is practical for production use). Conventional symmetric additive Schwarz is denoted ASM(k), where k is the overlap, restricted additive Schwarz (Cai and Sarkis, 1999) is denoted RASM(k). The average number of GMRES iterations per Newton iteration is shown in Table 3.2. Note that increasing overlap has no benefit when incomplete subdomain solvers are used.

The parallel algebraic multigrid packages ML (Gee et al., 2006) and BoomerAMG (Henson and Yang, 2002) provide potentially scalable alternatives. ML is based on smoothed aggregation, tends to coarsen very rapidly, and provides its restriction and coarse-level matrices to PETSc so that that elaborate smoothers can be used. ML does not converge for this problem with standard options; but with FGMRES on the outside of the V-cycle and GMRES(1) with RASM(1) as the smoother, using ICC(0) for the subdomain solve except on level 1, where a direct solve was used, we see 34 V-cycles per Newton iteration. ML needs only three levels to reach a coarse level with 144 degrees of freedom. BoomerAMG is a classical algebraic multigrid, which tends to coarsen slowly on anisotropic problems and does not expose the internal details, so smoother choices are limited. BoomerAMG needs seven levels to reach a coarse grid with 663 degrees of freedom and averages 76 iterations per Newton iteration. There were other, still somewhat challenging, problems for which BoomerAMG was competitive with geometric multigrid in terms of iteration count, but the setup costs and required number of levels was always large. ML never exhibited low iteration counts, even for easy problems.

Another approach to solving multicomponent problems is to split the components and solve scalar problems for each in hopes that the scalar problems can be more readily handled by available software such as algebraic multigrid. The split problems can be combined additively, multiplicatively, or symmetric multiplicatively. Unlike most Schwarz methods, additive methods are not typically implemented to expose concurrency, but it is simpler to implement in a matrix-light way because only the "self-coupling" terms need to be made available. Multiplicative methods need to apply the off-diagonal submatrix, and the most efficient way to do so is usually by assembling it; but the submatrix can also be applied by finite differencing the full residual. The results, shown in Table 3.3, are uninspiring, especially when considering that field-split creates additional synchronization points and attains lower throughput since it works with scalar matrices instead of block matrices (see Table 3.1).

A good geometric multigrid for this problem uses four-levels with a coarse grid of $10 \times 10 \times 2$ elements, which is semi-refined twice by a factor of 2 in the horizontal, then by a factor of 6 in the vertical to reach the target $40 \times 40 \times 12$ grid. The smoothers consist of a domain decomposition method and a subdomain solver that may be exact or inexact. Direct solves for the subdomain problems on level 1 are inexpensive and tend to improve robustness so we always use a direct solve. A different refinement

Table 3.3: Average number of GMRES iterations per Newton iteration for field-split preconditioners with different ways of combining the splits and different solvers within the splits. BoomerAMG used 7 levels and ML had 3 with the same solver parameters as discussed in the text for the coupled approach.

Solver in Splits	Method of combining splits		
	Additive	Multiplicative	Sym. Multiplicative
Cholesky	19	9.9	9.3
ML	41	34	30
BoomerAMG	89	83	78
RASM(1)+Cholesky	186	173	84

Table 3.4: Average number of GMRES iterations per Newton for different multigrid preconditioners.

Coarse Problem	Level 1		Level 2 and 3		Its
	Decomp.	Subdomain	Decomp.	Subdomain	
$10 \times 10 \times 2$ Redisc	BJacobi	Cholesky	BJacobi	Cholesky	8.9
$10 \times 10 \times 2$ Redisc	BJacobi	Cholesky	BJacobi	ICC(0)	9.6
$10 \times 10 \times 2$ Redisc	BJacobi	Cholesky	ASM(1)	ICC(0)	11.9
$10 \times 10 \times 2$ Redisc	BJacobi	Cholesky	RASM(1)	ICC(0)	6.9
$10 \times 10 \times 2$ Redisc	ASM(1)	Cholesky	ASM(1)	ICC(0)	20.2
$10 \times 10 \times 2$ Redisc	RASM(1)	Cholesky	RASM(1)	ICC(0)	5.9
$10 \times 10 \times 2$ Galerkin	RASM(1)	Cholesky	RASM(1)	ICC(0)	54.
$10 \times 10 \times 1$ Redisc	BJacobi	Cholesky	BJacobi	ICC(0)	10.3
$10 \times 10 \times 1$ Redisc	ASM(1)	Cholesky	BJacobi	ICC(0)	10.1
$10 \times 10 \times 1$ Redisc	RASM(1)	Cholesky	BJacobi	ICC(0)	6.9

involves a $10 \times 10 \times 1$ coarse grid and semi-refines twice by a factor of 2 in the horizontal, then by a factor of 12 in the vertical to reach the same target grid. The coarse levels are smaller in this case so the refinement is more efficient provided the iteratation counts are similar. Table 3.4 explores a variety of multigrid preconditioner configurations with each coarse level. A distinct effect is that inexact subdomain solvers cause minimal performance degradation; cf. Section 3.2 where a factor of 5 to 10 degradation is visible. Use of Galerkin coarse operators has a catastrophic effect on the iteration count and may help explain the poor robustness exhibited by the algebraic multigrids. We cannot explain why symmetric additive Schwarz performs so poorly on this problem, but the other numbers are robust to changes in resolution, spatial domain, and number of processes.

We remark that when using grid sequencing and the method in the last row of Table 3.4, the problem can be solved in seven Newton iterations on the fine level and an average of 5.4 V-cycles per Newton iteration. If the Eisenstat-Walker method is used to avoid oversolving, the problem takes eight Newton iterations with a total of 12 V-cycles.

3.5 Conclusion

Several projects (Lemieux et al., 2011; Larour et al., 2010; Johnson and Staiger, 2007; De Smedt et al., 2010; Pattyn, 2003; Lemieux et al., 2011) have developed software to simulate ice sheets using the hydrostatic equations. All of these projects solve the resulting algebraic equations using one-level domain decomposition (if parallel) and incomplete factorization, and all but (Lemieux et al., 2011) use Picard linearization. These methods are not scalable in the strong or weak sense so resolution and simulation turn-around time has been severely limited by the cost of the solver.

We have presented a grid-sequenced Newton-Krylov multigrid algorithm for solving the algebraic equations resulting from a finite element discretization of the hydrostatic equations. This geometric multigrid method demonstrates textbook multigrid efficiency for extreme topography and basal conditions and offers thousandfold speedups relative to the methods currently in use. Algebraic multigrid and field-split preconditioners were not found to be competitive in terms of robustness or efficiency.

The present velocity solver can be immediately incorporated in a semi-implicit ice sheet evolution model, but the resulting scheme is only conditionally stable (time steps must satisfy a CFL condition). Fully implicit time integration offers an alternative that has no such constraint and is thus suitable for simulations over full ice ages and longer, but requires the solution of more difficult algebraic equations. A prototype using the present velocity discretization has been developed, but scalable linear solvers for velocity-surface coupled problems is the subject of future investigation.

Several models involving only a vertically-integrated elliptic solve have been proposed as less costly alternatives to solving the hydrostatic equations, e.g. (Bueler and Brown, 2009; Goldberg et al., 2009; Goldberg, 2011). A scalable solver has not yet been proposed for these vertically-integrated elliptic problems so the present model is faster than existing implementations at sufficiently high resolution. To make a useful comparison, we estimate relative costs with the assumption that a similarly optimal solver exists for the vertically-integrated system, in which case the cost difference should be a constant factor independent of problem size and number of processors. The "hybrid" model of Goldberg (2011) requires evaluating effective viscosity in 3D and integrating to produce depth-averaged viscosity which appears in the vertically-integrated equations. Evaluating effective viscosity accounts for most of the cost of residual evaluation for the hydrostatic equations. Therefore the cost of such an evaluation is comparable to the cost of the nonlinear viscosity evaluation required in a Picard iteration for the hybrid model of Goldberg (2011). Our solver produces high-accuracy solutions in about 30 times the cost of a residual evaluation, less if a good initial guess was available as when time stepping. Goldberg (2011) uses a Picard iteration to solve the nonlinear system, requiring 50 iterations per nonlinear solve, therefore the hybrid solver is only likely to have similar cost to our hydrostatic solver when the linear solves require negligible time. If the hybrid equations were instead solved by Newton iteration and the linear solves required similar or less time than evaluating viscosity, then the hybrid equations might be solved as much as 5 times faster than the hydrostatic equations. Bueler and Brown (2009) factors depth dependent velocity and temperature contributions out of the vertically-integrated nonlinear problem. The process only applies if the constitutive relation can be factored and the resulting continuum equations have a lower formal order of accuracy with respect to shallowness and stickyness distribution than the model of Goldberg (2011). In exchange, the nonlinear elliptic solve can be completed without revisiting the 3D velocity field so the model cost has weaker dependence on the number of nonlinear iterations and thus potential to be a few times faster than the model of Goldberg (2011). Vertically-integrated models do not involve linear algebra operating on the 3D velocity field so they require significantly less memory than the hydrostatic equations when all operators are assembled.

The hydrostatic equations are well-suited to investigation of ice streams and outlet glaciers with gentle slope and arbitrary stickyness distribution. For example, 3D ice stream thermodynamics (see Raymond (2000) for analytic and Truffer and Echelmeyer (2003) for numerical investigation with along-flow symmetry) could be reliably investigated using velocity defined by the hydrostatic equations. The methods in this paper can efficiently solve the discrete hydrostatic equations in regimes with extreme basal topography and stickyness distributions. The solution method is efficient even in regimes which are far in excess of where the continuum hydrostatic approximation is valid as an approximation to the Stokes equations. The latter issue of continuum validity has been partially addressed using asymptotic analysis (Schoof and Hindmarsh, 2010) and numerical comparison (Hindmarsh, 2004; Pattyn et al., 2008). Despite being more complete than models involving only a vertically-integrated elliptic solve, the hydrostatic equations are not suitable for steep outlet glaciers like Jakoshavn Isbræ or Pine Island Glacier where solution of Stokes problems appears to be unavoidable.

Comprehensive log files and the scripts used to produce the figures in this paper are publicly

available. They are currently hosted along with the source for this document at `https://github.com/jedbrown/tme-ice`.

3.6 Addendum: Implicit free surface and erosion

This addendum is not part of the submitted paper, but represents an extension to a stiffly-coupled multiphysics problem with very little changes to the code, made possible by the software tools to be introduced in Section 4.1.

Solving the transient and steady-state problems with an implicit free surface is the next step. As a proof of concept, we consider the hydrostatic equations from Section 3.2 augmented with surface evolution and erosion

$$-\nabla\cdot\left[\eta\begin{pmatrix}4u_x+2v_y & u_y+v_x & u_z\\ u_y+v_x & 2u_x+4v_y & v_z\end{pmatrix}\right]+\rho g\nabla s=0 \tag{3.6.1}$$

$$h_t+\nabla\cdot\int_b^s(u,v)=0 \tag{3.6.2}$$

$$b_t+k_e\left|(u,v)\right|^l=0 \tag{3.6.3}$$

where (u,v) is horizontal velocity, η is the nonlinear effective viscosity, h is the thickness, b is the bed elevation, and $s=b+h$ is the surface elevation. The power law erosion model with nondimensional erosion parameter $k_e=10^{-3}$ and exponent $l=1$ was used by Tomkin and Braun (2002); Herman and Braun (2008); Egholm et al. (2011). The velocity equation is solved on the domain between b and h with the basal sliding parameter held constant. This is an Arbitrary Lagrange-Eulerian (ALE) formulation (Donea et al., 2004) in which the mesh always conforms to the current surface. The ISMIP test C (Pattyn et al., 2008) initial geometry, periodic horizontal boundary conditions, and basal sliding parameter distribution are used. The thickness is discretized using a cell-centered upwind finite volume method and fluxes are computed by splitting the faces of the continuous finite element solution for the velocity and performing numerical quadrature which is exact for affine elements. The steady-state solution with erosion turned off is shown in Figure 3.7, computed in 19 Newton iterations (without time stepping). There is no steady state when eroison is turned on, but the transient simulation can still take time steps dictated by the physical process of interest instead of a CFL condition that may not be interesting. Figure 3.8 shows the bed elevation after 300 ka of evolution, with time steps of 30 ka. This time step length corresponds to a CFL number of nearly half a million. In practice, the climate changes somewhat faster than 30 ka so a smaller time step size would be used, but it is still orders of magnitude slower than the CFL constraint for explicit advection. Since the short time-scale transient response is not of scientific interest for long-term erosion modeling (e.g. mountain range formation), this implicit free surface is much more efficient.

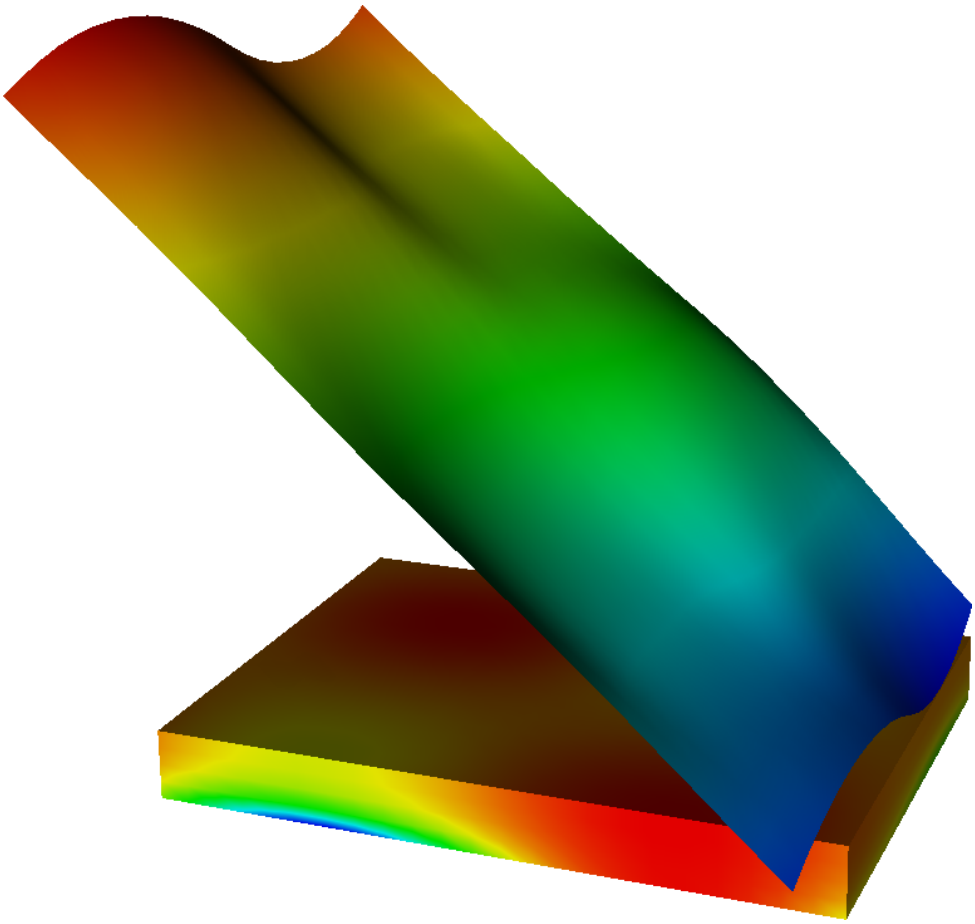

Figure 3.7: A steady-state solution for ISMIP-HOM test C (Pattyn et al., 2008) at 10km computed in 19 iterations. The elevated surface is exaggerated surface height and the color in the solid domain is velocity.

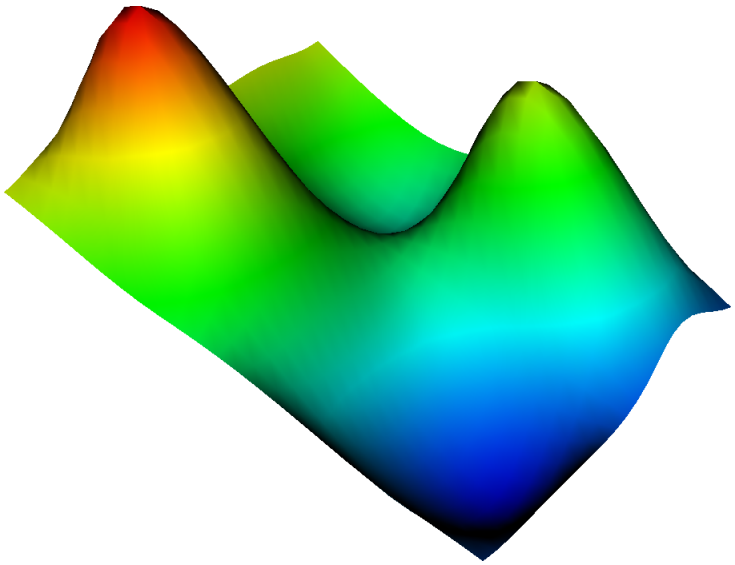

Figure 3.8: Bed profile eroded from a flat bed after 300ka with test C slipperiness perturbation. Time steps are 30ka at this point in the simulation.

Chapter 4

Software

Implementing convergent numerical models for complex physical processes such as ice streams and grounding line dynamics is a time consuming task. If the model must also be accurate, achieve high performance, exhibit good parallel scalability, be portable to different operating systems and hardware, and be flexible enough to add new physics or couple to other models as part of a larger problem, the complexity explodes. Well-designed software libraries help to manage the complexity and to greatly reduce development time for new models. Perhaps more important than reducing initial development time, libraries enable experimentation with different methods at extremely low effort. The best way to experiment is entirely with run-time options, such that no code modifications or recompilation is necessary.

Using different algorithms, each of which may be necessary for different parameter ranges or to obtain high performance on different architectures, require the use of different data structures. A fundamental tenet of object oriented design, and PETSc in particular, is that users should not interact directly with the underlying data structures and should not depend on on any specific implementations of an interface. The requirement to interact through a generic interface without impacting performance places hard restrictions on the design of those interfaces. As a general rule, the interface should have a granularity that is large enough to offer high performance, but is not so large that it becomes restricted to certain data structures. All major classes in PETSc have a plugin architecture with object creation managed by a service locator pattern Fowler (2004). This enables, for example, a vendor to compile a matrix format and/or preconditioner that performs well on their hardware into a shared library such that an end user of any PETSc program can select the vendor's plugins at run-time without access to any source code.

There is a balance between "frameworks" that take control of the model (often at `main()`) and offer a high-level interface to the user versus libraries that leave the user in control. Frameworks make many assumptions about the equations and methods to be used, which allow a high level of abstraction and rapid development of a new models, provided the equations and discretization are supported by the framework. On the other hand, these assumptions limit the ability to solve problems and use methods that are unnatural for the framework, as well as the ability to couple with other numerical models as part of something larger, or to perform analysis that is not explicitly supported by the framework (e.g. bifurcation, sensitivity, optimization). Libraries, on the other hand, take more effort (which we try to minimize) to build a working model, but the user has much more control and the resulting code is more amenable to reuse in a larger scope. PETSc's solvers are purely algebraic and therefore make no assumptions about the discretization or coupling to external systems. The scope of my new library, Dohp, is strictly limited to the task of turning a continuum formulation and input data including a computational mesh into discrete algebraic equations. Unlike many discretization libraries, Dohp makes no attempt to also manage the solution of algebraic equations. PETSc's algebraic interfaces are used exclusively. Since the data types are extensible, their use places no restriction on functionality or performance, but having a single common interface substantially reduces the "impedance mismatch" of mixing code from different packages.

This section discusses the design decisions and implementation of several software components that I

have developed to improve the capability, performance, and verifiability of numerical simulations. These contributions have broader scope than glaciology, but each component provides significant benefit to the overall capability of numerical models in glaciology.

4.1 Multiphysics coupling

Ice flow and more generally, earth system models, involve many physical processes occurring on multiple spatial domains and possessing multiple time scales. An important challenge in computational science is addressing how to couple models of these processes in a maintainable way without sacrificing accuracy and analysis capability. The traditional method of splitting in time and integrating each process independently has been shown to result in low-order splitting errors (Knoll et al., 2003; Mousseau et al., 2002) for many problems of interest, as well as fundamentally different results regarding the stability of the system (Estep et al., 2008) and existence of steady states versus limit cycle behavior (Jardin et al., 2008). Tightly coupled multiphysics software should enable the use of robust IMEX [1] or fully implicit methods while permitting different physics modules to be managed independently. There are essentially two scalable approaches to tightly coupled multiphysics: field-splitting and coupled multilevel methods such as domain decomposition or multigrid. Field splitting is most effective when it decomposes the coupled problem into separate 'blocks' of equations that are well-understood and for which known methods perform well. Coupled multilevel methods are favored when local spectral structure and compatibility conditions of the equations are well-understood so that effective smoothers and grid transfer operators can be defined.

In many cases (Rannacher, 2000; Jameson and Caughey, 2001; Adams et al., 2010), the monolithic coupling approach has potential to offer the best possible performance. The general guidelines are to define the smoother in terms of low-order discretization and to smooth all components at any node or element of the grid at the same time, usually either by block Gauss-Seidel relaxation or by block incomplete LU factorization. The former permits a nonlinear smoother which has higher arithmetic intensity, but the latter tends to be more robust, especially to strong anisotropy. The purpose of the low-order discretization (first-order upwind for hyperbolic terms) is to retain h-ellipticity, a necessary and sufficient condition for the existence of a pointwise smoother (Brandt and Dinar, 1979; Trottenberg et al., 2001). When an operator fails to satisfy h-ellipticity, there are high-frequency error components that are poorly corrected by the smoother.

For multi-component problems, the relative scaling between fields is important for good solver performance. This can be difficult to achieve when different parts of the domain are in different regimes. When the problem is indefinite, the smoother and restriction operators need to be chosen to be compatible with the inf-sup condition. Such restriction operators are usually defined geometrically, though they have also been defined algebraically for contact problems (Adams, 2004). The associated smoothers (Vanka, 1986) are relatively more expensive than for definite problems, and it is difficult to handle anisotropy because incomplete factorization as a smoother becomes problematic due to zero or negative pivots (Higham, 2002; de Niet, 2007).

It is difficult to write generic software for coupled multigrid because many fine-grained operations, especially the definition of the local smoother, require physics- and discretization-dependent choices. Additionally, very little theory is available for coupled multi-physics and it is unclear how to define smoothers for multi-domain problems or surface-volume coupled problems.

A different approach is to apply a Newton iteration on the coupled problem and solve the linear system with a Krylov method using a preconditioner defined in terms of some operator splitting such that the physics of each part of the decomposition can be understood and efficient solvers can be provided. This approach (Knoll and Keyes, 2004) has proven successful at reusing software, with many papers accessing

[1] IMplicit-EXplicit (IMEX) methods treat problems with well-separated time scales by solving slow processes explicitly and fast processes implicitly. IMEX methods are designed so as not to suffer from the splitting errors that plague first-order operator splitting methods.

such methods through PETSc (Balay et al., 2012b). However, until my recent additions to PETSc, it was cumbersome to package separate physics separately without compromising which methods are available, and decisions about using field-split versus monolithic methods were typically required up-front. The enhancements to PETSc allow each physics to be packaged independently without sacrificing performance or runtime flexibility in choice of methods. We first summarize a variety of field-split methods and their requirements from the user, then we examine the new linear algebraic interface by which the user can generically provide this information.

4.1.1 Field-split preconditioning

Suppose the Jacobian of the original coupled problem has block structure

$$J = \begin{pmatrix} A & B \\ C & D \end{pmatrix}. \tag{4.1.1}$$

We explain the methods for 2×2 block systems, but $n \times n$ block systems can be treated similarly. Field split methods can be classified as block relaxation or factorization.

Relaxation methods are inspired by the classical stationary iterative methods the preconditioners taking the forms

$$\begin{aligned} P^{-1}_{\text{Jacobi}} &= \begin{pmatrix} A & \\ & D \end{pmatrix}^{-1} \\ P^{-1}_{\text{GS}} &= \begin{pmatrix} A & \\ C & D \end{pmatrix}^{-1} \\ P^{-1}_{\text{SGS}} &= \begin{pmatrix} A & \\ & \mathbf{1} \end{pmatrix}^{-1} \left(\mathbf{1} - \begin{pmatrix} A & B \\ & \mathbf{1} \end{pmatrix} \begin{pmatrix} A & \\ C & D \end{pmatrix}^{-1} \right) \end{aligned}$$

which can be accessed at runtime using the PETSc option `-pc_fieldsplit_type additive`, `multiplicative`, and `symmetric_multiplicative` respectively. Relaxation is simple to use and expected to perform well when the diagonal blocks control the dynamics of the system or in which the coupling is one-way. Heuristically, this occurs when there are few modes y in the second space for which $\|By\|$ is large compared to $\|Dy\|$. We expect problems if A or D is nearly singular or "small" compared to B and C on a sizable subspace. More precisely, relaxation splitting will perform well if D is a good approximation to the Schur complement $S = D - CA^{-1}B$.

Stiff hyperbolic waves and systems with constraints are two common cases where relaxation breaks down. The first is somewhat more benign and we use the shallow water equations as an illustrative example. In conservative non-dimensional form with thickness h and momentum uh, the flat-bed shallow water equations are

$$\begin{aligned} (uh)_t + \nabla \cdot \left(u \otimes uh + \frac{g}{2} h^2 \mathbf{1} \right) &= 0 \\ h_t + \nabla \cdot uh &= 0 \end{aligned}$$

where g is the gravitational acceleration. When the gravity wave speed $\sqrt{gh}$ is much faster than the velocity u, we are in the low Mach limit which is an especially interesting case for global circulation models. Semi-discretizing in time using implicit Euler with step size Δt, the Jacobian takes the form

$$\hat{J}(uh, h) = \begin{pmatrix} \Delta t^{-1} \mathbf{1} & gh\nabla \\ \nabla \cdot & \Delta t^{-1} \mathbf{1} \end{pmatrix} \tag{4.1.2}$$

where several "slow" terms in the second row have been suppressed. The off-diagonal blocks carry the energy of the gravity wave and capture the stiffness that appears for large time steps, which explains

why relaxation performs poorly for (4.1.2). The alternative is to define a factorization preconditioner for (4.1.1) as

$$P^{-1} = \begin{pmatrix} A & B \\ & S \end{pmatrix}^{-1} \begin{pmatrix} 1 & \\ CA^{-1} & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & A^{-1}B \\ & 1 \end{pmatrix}^{-1} \begin{pmatrix} A & \\ C & S \end{pmatrix}^{-1} \tag{4.1.3}$$

where $S = D - CA^{-1}B$ is the Schur complement. When used as a right (left) preconditioner for GMRES, the lower (upper) triangular blocks of (4.1.3) can be dropped without changing the eigenvalues of the preconditioner operator (Murphy et al., 2000; Ipsen, 2001) which saves one solve with A per Krylov iteration. The resulting preconditioned operator has minimal polynomial degree 2 so that GMRES converges in two iterations if the preconditioner is applied exactly. When the problem is symmetric, it is possible to precondition MINRES with the positive definite

$$P^{-1} = \begin{pmatrix} A & \\ & -S \end{pmatrix}$$

which produces a preconditioned operator with minimal polynomial degree 3 so that MINRES converges in three iterations. In practice, the inverses appearing in (4.1.3) are only applied approximately using some scalable method such as a V-cycle of multigrid so that the outer iteration converges in a constant number of iterations independent of grid resolution. Block factorization methods in this family are available in PETSc through `PCFieldSplit` and can be accessed at run time using `-pc_fieldsplit_type schur -pc_fieldsplit_schur_factorization_type lower`, `upper`, `diag`, and `full`.

In the case of the semidiscrete shallow water equations (4.1.2), S is similar to the Helmholtz differential operator $\Delta t^{-1} - \Delta t g \nabla \cdot h \nabla$. This "good Helmholtz" structure is characteristic of stiff wave problems and discussed at length in the review (Knoll et al., 2005) and a variety of applications (Mousseau et al., 2002; Chacón, 2008; Park et al., 2009). This Helmholtz operator is our first example of an "auxiliary matrix" not explicitly present in the continuum equations, but needed by the preconditioner.

The situation becomes more delicate for problems with constraints such as contact problems and incompressible flow (Elman et al., 2008). In these problems, the "important" part of the matrix A is an elliptic operator so the Schur complement becomes dense. For isotropic variable-viscosity Stokes problems, the Schur complement is spectrally equivalent to a mass matrix defined with respect to the inverse-viscosity weighted inner product (Olshanskii and Reusken, 2006). This weighted mass matrix is another example of an auxiliary matrix. Time dependence in a generalized Stokes problem with variable viscosity and density can be accommodated by using both the weighted mass matrix and a density-weighted Neumann Laplace operator (Olshanskii et al., 2006). For anisotropic viscosity, Navier-Stokes, and other indefinite systems, we no longer have provable spectral equivalence and instead turn to heuristics based on approximate commutator arguments. The idea is to find $\tilde{A}$ operating on the dual space (e.g. pressure) such that

$$CA^{-1}B \approx L\tilde{A}^{-1}M$$

where $L = CM_1^{-1}B$. Here, M_1 is the mass matrix in the primal space (e.g. velocity) and M is a mass matrix in the dual space. For many problems, L is a discrete Laplacian with Neumann boundary conditions that can be assembled independently to preserve sparsity. Applying such preconditioners requires solves with M and L, but only multiplication by $\tilde{A}$. There are three common approaches to constructing such an approximate commutator.

The first approximate commutator approach is to use physical arguments to define $\tilde{A}$ by considering the continuum operators without boundary conditions (though see Elman and Tuminaro, 2009). An example of this is the "pressure convection-diffusion" preconditioner of Silvester et al. (2001); Kay et al. (2002) for the Navier-Stokes equations. Such preconditioners, if one accounts for the action of the mass matrix, require three auxiliary operators $(M, L, \tilde{A})$ to be provided by the user (although approximations in terms of diagonals of existing matrices are possible). This approach has been shown to produce nearly mesh independent convergence rates (Elman, 2005; de Niet and Wubs, 2007; Elman et al., 2008) for Navier-Stokes problems, with modest dependence on Reynolds number.

The second approximate commutator approach is to use a least squares argument to define

$$\tilde{A} = ML^{-1}CM_1^{-1}AM_1^{-1}B$$

where M_1 is usually approximated by its diagonal (Elman, 1999; Elman et al., 2006). Application of this preconditioner requires an additional solve with L and consequently has higher computational cost per iteration than the first alternative, but anisotropy and other terms are naturally accommodated with no additional effort. This method shows near mesh independence for solving the Navier-Stokes equations, and while slightly less robust, typically has better performance than the first method when Newton linearization is used (Elman et al., 2008). It has been used successfully for challenging variable viscosity Stokes problems discretized using $Q_1 - P_0^{\text{disc}}$ finite elements. The "least squares commutator" is available in PETSc using `PCLSC`.

The third method is to define $\tilde{A}$ using sparse approximate commutators (Elman et al., 2006), a similar method to the sparse approximate inverse (Grote and Huckle, 1997). This approach has the attractive property of requiring only one solve per iteration without the need for additional user-provided auxiliary matrices.

There are several important variants of the methods above including the augmented Lagrangian (Awanou and Lai, 2005; Dohrmann and Lehoucq, 2006; de Niet and Wubs, 2007) which accelerates convergence by penalizing A using a multiple of $BM^{-1}C$ which is singular (thus making the inner problem more difficult to solve with most methods) as well as the use of pivoting for preconditioning full-space iteration in PDE-constrained optimization (Biros and Ghattas, 2005a,b; Akçelik et al., 2006). See Benzi et al. (2005) for a review of methods for saddle point problems.

As a practical and architectural concern, we warn that that excessive splitting leads to lower arithmetic intensity and more synchronization points which reduces the utilization of modern hardware. Therefore, it usually only makes sense to split when there is a clear reward in superior algorithmic performance such as many fewer Krylov iterations or substantially reduced storage requirements (e.g. by not needing to assemble inter-field coupling or by taking advantage of symmetric block storage for part of a larger problem).

4.1.2 Linear algebraic interfaces to facilitate field-split preconditioning

One ideal governing the PETSc approach is to isolate the user's specification of the discretization and physics from the solution methods and underlying data structures so that solver choices can be delayed until run time. The user should not need to write any special code to support a given solver (in some cases auxiliary matrices may still be needed), thus new solvers will automatically be available when added to the library or present in a plugin. The seemingly simple matter of interlacing fields for high throughput (see Section 4.2) versus splitting them for certain preconditioners motivates an interface in which "blocks" are addressed more abstractly than by number. Instead of addressing a field by number, PETSc uses the `IS` class which represents an arbitrary index set. An `IS` has an MPI communicator and indices which may be represented more compactly than arrays if they have structure such as a regular stride.

There are two classes of matrix storage format relevant to field-split versus monolithic preconditioners. The traditional compressed sparse row formats, known as `AIJ` (plus symmetric and node-blocked variants), are fully assembled and stored in row-partitioned form. The `Nest` format does not store entries directly, instead it stores nested matrices of any format with the action of the whole matrix defined in terms of its nested blocks. Each nested block can use a different format, enabling optimization for symmetry and constant block size, even for mixed discretizations or multi-domain problems where it would otherwise not be possible.

`MatGetSubMatrix()` is the primary interface used by `PCFieldSplit` to extract blocks from the Jacobian. This function extracts a parallel submatrix with distribution specified by the distribution of the row and column index sets. For monolithic matrix formats like `AIJ`, `MatGetSubMatrix()` necessarily requires the matrix entries to be copied into a new data structure that as much as doubles the memory needed by an

application. In the generic parallel setting, this operation requires complex communication patterns that are not memory scalable as currently implemented. For matrices in the `Nest` format, `MatGetSubMatrix()` returns the submatrix without making a copy or any parallel communication.

While Schur complements needed by block factorization could be constructed from submatrices, it would be cumbersome for the user to provide problem-specific approximations to the Schur complement this way. Instead, `PCFieldSplit` uses

```
PetscErrorCode MatGetSchurComplement(Mat mat,
                  IS isrow0,IS iscol0,IS isrow1,IS iscol1,
                  MatReuse mreuse,Mat *newmat,MatReuse preuse,Mat *newpmat);
```

to extract the Schur complement of `(isrow0,iscol0)` restricted to `(isrow1,iscol1)` and an approximation to be used for preconditioning. The default implementation returns a matrix of type `SchurComplement` which applies S according to its definition $D - CA^{-1}B$ by storing the four blocks, and if requested by the caller, the SIMPLE (Patankar and Spalding, 1972) approximation $D - C\text{diag}(A)^{-1}B$. This function can be overridden by the user to return arbitrary problem-specific data.

The `MatGetSubMatrix()` and `MatGetSchurComplement()` interfaces are perfectly adequate for use by `PCFieldSplit`, and `Nest` matrices provide an efficient storage format, but since the assembly interface is different for creating separate blocks from a monolithic matrix, it seems to require the user to decide up-front whether to assemble a `Nest` or a `AIJ` matrix. Additionally, if `AIJ` is used, assembly for a single physics would need to "know" about the other fields in order to set the indices correctly for insertion. These are both highly undesirable because they limit the algorithmic choices that can be made at run time. To overcome this unsatisfactory performance and interface complexity, I introduced a new interface for modular and generic assembly.

Before describing the interface, we need to define "local" spaces in the multi-physics context. We first assume a non-overlapping partition of owned nodes across subdomains, where typically subdomains are identified with MPI processes. We also assume a possibly overlapping partition of the integration domain. In the continuous finite element context, assembly is done by integration over elements, with each element typically integrated on exactly one process (thus the partition is non-overlapping). Note that finite difference and finite volume methods can also be interpreted as integration with appropriate quadratures. The local space is then defined as the set of all nodes with support on the integration subdomain. This is also exactly the domain on which state variables need to be defined to evaluate a subdomain's contribution to the residual and contains the residual contribution from any given subdomain (identical for continuous finite element methods). The local space comes with an independent ordering (starting at zero for each subdomain) and a local-to-global mapping that translates local indices to global indices. Matrices and vectors typically carry a local-to-global mapping so that entries can be referenced by local index. A sub-physics local space is defined as the subset of a multi-physics local space that contributes to a particular physics (or field or other "split") and is uniquely represented by an index set (`IS`) containing those multi-physics local indices appearing in the sub-physics local space. When `DMComposite` is used to manage a multi-physics problem, each sub-physics local index set is a contiguous range.

For the purpose of matrix assembly, the local space defines the part of the global vector and matrix into which entries can be contributed by the method outlined below. This is a restriction relative to the standard method of inserting any entry by global index, but the user can still define the local-to-global mapping to allow insertion wherever they desire. In return for this restriction, partially assembled matrices required by non-overlapping domain decomposition methods like FETI-DP (Farhat et al., 2001, 2000; Klawonn and Widlund, 2006; Klawonn and Rheinbach, 2007b,a) and the closely related BDDC (Dohrmann, 2003; Mandel and Dohrmann, 2003; Li and Widlund, 2006) can be assembled with no changes to user code.

Assembly of "sub-physics" blocks as well as off-diagonal coupling blocks is achieved using

```
PetscErrorCode MatGetLocalSubMatrix(Mat A,IS rows,IS cols,Mat *submat);
```

which returns a submatrix with local indices defined by the index sets. This function is not collective and makes only weak guarantees about the functionality implemented by the returned submatrix. In particular, the communicator is not specified and collective operations like `MatMult` may not be defined. If the matrix storage format is `Nest`, this function simply returns the appropriate submatrix, which usually lives on a parallel communicator and has full functionality. For matrix formats that do not implement `MatGetLocalSubMatrix()`, a proxy matrix is set up on `PETSC_COMM_SELF` that implements `MatSetValuesLocal()` and similar functions by translating sub-physics local indices to coupled local or global (best choice depending on what is explicitly supported) indices and setting values in the parent matrix. If the row and column index sets have matching block size attribute, `MatSetValuesBlockedLocal()` is also implemented regardless of whether the underlying storage format uses blocks. This permits sub-physics modules having constant block size to always speak the most specific language (blocked and/or symmetric) regardless of the underlying format. When the underlying format specifically supports blocks, we reap the benefits of faster insertion due to fewer searches and moves, otherwise there is negligible performance penalty. When `MatSetValuesBlockedLocal()` is used recursively and the matrix implementation has no specific support, the proxy matrices are flattened so that index translation is never done more than once. This interface allows separate physics modules to be packaged independently and gives a run-time choice of monolithic matrix formats or `Nest` with the best possible format in each block and high performance in all cases.

4.2 High throughput on cache-based architectures

The easiest way to make software scalable is to make it sequentially inefficient. (Gropp, 1999)

4.2.1 Sparse matrix kernels

This section briefly describes some optimizations I made to improve PETSc's matrix kernels by about 30% on Intel and AMD architectures, bringing them quite close to the memory bandwidth limits. This was implemented subsequent to the new data structures for factored matrices described in Smith and Zhang (2010) and thus represents an additional improvement.

The following hardware descriptions generally hold across Intel Core, Core 2, and Core i7 as well as AMD K8 and K10 microarchitectures. Most variation is due to different prefetch and TLB semantics as well as different latencies for vector instructions. Details can be found in the optimization reference manuals (Intel, 2011; AMD, 2011) as well as Agner Fog's excellent resources Fog (2011b,a). See Drepper (2007) for a more accessible introduction to the memory hierarchy.

Most of my optimizations for sparse matrices involve maximizing reuse of caches and attained fraction of peak streaming bandwidth. Each floating point unit can issue one packed add and one packed multiply per clock cycle, with latencies of 3 to 5 cycles. SSE instructions may have both operands in registers or have one operand in level 1 data cache (L1D) with no throughput penalty, although a concurrently issued add and multiply cannot both have a memory operand. AMD has a 2-cycle latency penalty for memory operands, Intel has no latency penalty. This means that L1D is almost as good as a register, compared to L2 with roughly 10-cycle latency and DRAM with approximately 250-cycle latency.

Assuming a 2.5 GHz clock, one packed load and store per cycle means a bandwidth to L1D of 24 GB/s each way (48 GB/s each way for newer architectures supporting 32-byte AVX registers). We can compare this to per-core memory bandwidth ranging from 1.75 GB/s (6-core with 2-channel DDR2-667, found on Cray XT-5) to 9.6 GB/s (4-core with 3-channel DDR3-1600, found on Intel's newest "Sandy Bridge" processors). With four (SSE2) or eight (AVX) double precision floating point operations possible per cycle, it becomes clear that memory will be the bottleneck unless many floating point operations can be performed per value loaded. The ratio of flops per byte for a computational kernel is known as arithmetic intensity. The previously mentioned architectures achieve balance in computation and memory accesses when the arithmetic intensity is between 2 and 6 flops/byte.

The hardware prefetch unit can recognize 16 forward-moving streams and 4 backward-moving streams, but only one stream per 4 kB page and not across page boundaries. Additionally, the hardware prefetcher does not resolve minor TLB misses. This is a significant issue because the TLB can only address 1 or 2 MB and TLB misses produce roughly 60 cycles of latency plus DRAM latency if the resulting address is not in high-level cache. Software prefetch can hide these latencies and resolve TLB misses in advance, providing roughly 20% improvement in my optimized implementations of the STREAM benchmarks (McCalpin, 2007) discussed later in this section. In addition to improving overall bandwidth, software prefetch instructions can set a "non-temporal access" (NTA) policy such that lines evicted from L1D will not subsequently reside in higher level caches.

We consider four matrix formats with increasing amounts of structure: `AIJ`, `AIJ/Inode`, `BAIJ`, and `SBAIJ`. `AIJ` is a standard compressed row format that stores a column index for each nonzero entry. A matrix-vector product $y \leftarrow Ax$ is characterized by the `MatMult` kernel,

```
for (i=0; i<m; i++) {
    y[i] = 0.0;
    for (j=ai[i]; j<ai[i+1]; j++)
        y[i] += aa[j] * x[aj[j]];
}
```

Assuming perfect cache reuse with m rows and an average of n nonzeros per row, `MatMult` has an arithmetic intensity of

$$I_{\text{AIJ}} = \frac{2n}{(n+1)\texttt{sizeof(Scalar)} + (n+1)\texttt{sizeof(Int)}} \xrightarrow{n\to\infty} 0.167\text{flops/byte}$$

where the long-row limit is calculated for `double` precision and 32-bit integers. In practice, the vector will not reuse cache perfectly, so some entries of x will need to be loaded into cache more than once. Choosing a low-bandwidth ordering such as Reverse Cuthill-McKee (Cuthill and McKee, 1969) (RCM) improves cache reuse. This reduces end-to-end run time substantially, speeding up matrix-free operations such as residual evaluation in addition to sparse matrix kernels, e.g. by more than a factor of 2 (Gropp et al., 2000b). My optimization for `AIJ` consists of inserting software prefetch instructions to initiate loads of every cache line holding entries in the row after the current one. The L1D cache line length is detected during configuration so only one prefetch instruction per line is issued (it is 64 bytes on the x86-64 microarchitectures considered here). Sparse matrix-vector products for PDEs can be thought of as a stencil operation combined with a high-volume stream of matrix entries and column indices (the weights and shape of the stencil). Without the NTA policy, the high-volume stream flushes the vector x out of caches, even though the vector will be reused and the matrix entries and indices will not be.

The `AIJ/Inode` format simply augments the `AIJ` format by marking "Inodes" (clusters of consecutive rows that have the same nonzero pattern). The matrix kernels unroll over the Inodes which reduces the memory demands for column entries and performs more work per entry loaded from the vector. Use of the `AIJ/Inode` format is automatic if the matrix has consecutive rows with the same nonzero pattern. The column indices are *not* copied out of the standard `AIJ` storage so matrix kernels skip over $b-1$ rows of identical column indices in each Inode of size b. Skipping over these rows of column indices is especially inefficient with hardware prefetch because entries are eagerly brought into cache and then skipped, followed by a full DRAM stall as the kernel jumps past the prefetched indices in `aj` and starts $b-1$ new streams from `aa` that are not predicted by the prefetch unit. Software prefetch for `AIJ/Inode` avoids bringing in the redundant column entries and prevents the stalls due to jumping over column indices `aj` and starting new `aa` streams. The prefetch logic assumes that the next Inode size and row lengths will be approximately the same size as the current one since this is the typical case for PDE-like problems. If no redundant column entries are brought in, the arithmetic intensity is

$$I_{\texttt{AIJ/Inode}} = \frac{2nb}{(n+1)b\texttt{sizeof(Scalar)} + (n+2)\texttt{sizeof(Int)}} \xrightarrow[b=3]{n\to\infty} 0.214\text{flops/byte}$$

where the example Inode size of 3 is representative of 3D elasticity or the viscous part of a Stokes problem. We expect this performance model to be somewhat less sharp than for `AIJ` because it is not possible to load less than a cache line, therefore redundant column indices are unavoidable unless all rows naturally align with cache line boundaries.

The `BAIJ` format optimizes for constant block size by storing only one column index per $b \times b$ block. With this format, the problems of skipping ahead in `aj` and `aa` go away, but issues of peak bandwidth and enabling the vector to reuse cache remain. The arithmetic intensity

$$I_{\texttt{BAIJ}} = \frac{2nb}{(n+1)b\texttt{sizeof(Scalar)} + (n/b+1)\texttt{sizeof(Int)}} \xrightarrow[b=3]{n\to\infty} 0.237\text{flops/byte}$$

is very close to the dense limit of 0.25. In practice, `BAIJ` achieves a higher fraction of the bandwidth peak due its more regular memory access and unrolled kernels.

The symmetric block format `SBAIJ` stores only the upper-triangular part, in compressed row format, and has a more involved memory access pattern. Each matrix entry is only loaded once and is used for both upper and lower triangular contributions. The contribution from the lower-triangular part has multiple destinations, effectively doubling the number of stencil-like streams that must be sustained. All the new streams are read-write as opposed to the read-only streams of nonsymmetric row formats. In return, the storage requirement is halved and arithmetic intensity is doubled compared to nonsymmetric formats,

$$I_{\texttt{SBAIJ}} = \frac{4nb - 2b^2}{(n+2)b\texttt{sizeof(Scalar)} + (n/b+1)\texttt{sizeof(Int)}} \xrightarrow[b=3]{n\to\infty} 0.47\text{flops/byte.}$$

The data structure change described in Smith and Zhang (2010) caused both forward and back solves with (incomplete) LU factors to traverse matrix entries by moving forward in memory. Even though the matrix entries are fetched moving forward, the stencil for "back solve" still moves backward through memory so there are much fewer hardware prefetch streams (4 instead of 16 forward-moving on Intel/AMD; note that there is no hardware prefetch for backward-moving streams on Blue Gene/P).

For (incomplete) Cholesky, the factors are only stored once, so it is not possible for both forward and back solves to traverse matrix entries by moving forward. Because the primary backward-moving streams are prefetched by software, this causes relatively little performance degradation compared to ILU.

As an estimate of the peak usable bandwidth, we use the STREAM Triad benchmark with my moderately tuned implementation (software prefetch, SSE2 arithmetic, non-temporal stores, unrolled over cache lines). We consider two test systems, a Core 2 Duo (P8700) clocked at 2.53 GHz with a single DDR2-800 memory channel (theoretical bandwidth of 6.4 GB/s) and a two-socket Opteron 2356 (quad core) clocked at 2.3 GHz with two channels of DDR2-677 memory per socket (theoretical peak of 5.3 GB/s per channel). The achieved bandwidth for Triad on the Intel system is 5.4 GB/s for a single thread, there is no further improvement for multiple threads or processes. On Opteron, the single-threaded Triad achieved 5.8 GB/s, increasing to 11.1 GB/s for 2 processes (1 per socket), 14.8 GB/s for four processes (2 per socket), and 13.1 GB/s for 8 processes. Note that a single process very nearly saturates the usable bandwith of an entire socket, regardless of whether one or two memory channels are available. Within a single socket, 2 processes achieve 7.6 GB/s while 4 processes see only 6.6 GB/s. Although these Opteron results are The memory architecture of It is clear that the memory architecture within and between sockets limits the realizable bandwidth to well below the theoretical peak.

Table 4.1 shows the performance of the three primary sparse matrix kernels with each storage format on both test machines. With prefetch enabled, the sparse matrix kernels obtain a remarkably high fraction of theoretical peak bandwidth, especially on Intel processors. Indeed, the scalar format performance is competitive with the best `MatMult` results of Williams et al. (2007) and block formats are consistently faster by a factor of 20 to 30 percent. Block formats provide greater benefit for `MatSolve` than for `MatMult`, with `BAIJ MatSolve` frequently outperforming the corresponding `MatMult`. We are not aware of previous reports of this effect and speculate that it is due to `MatSolve` having fewer read streams and using only one vector at a time (`MatMult` reads from one vector and writes to another, `MatSolve` is in-place).

	Core 2, 1 process			
	AIJ	AIJ/Inode	BAIJ	SBAIJ
MatMult	916(103%)	855(81%)	1013(86%)	1429(62%)
MatSolve/ILU	799	742	1078	—
MatSolve/ICC	741	737	1011	1007
	Opteron, 1 process			
	AIJ	AIJ/Inode	BAIJ	SBAIJ
MatMult	506(53%)	592(52%)	735(58%)	744(30%)
MatSolve/ILU	460	572	845	—
MatSolve/ICC	426	427	815	814
	Opteron, 4 processes			
	AIJ	AIJ/Inode	BAIJ	SBAIJ
MatMult	1673(69%)	1978(68%)	2450(76%)	3089(49%)
MatSolve/ILU	1621	1896	2766	—
MatSolve/ICC	1519	1522	2700	2710
	Opteron, 8 processes			
	AIJ	AIJ/Inode	BAIJ	SBAIJ
MatMult	2374(111%)	2408(94%)	2897(102%)	4331(77%)
MatSolve/ILU	2321	1954	2913	—
MatSolve/ICC	2214	1854	2892	2877

Table 4.1: Throughput (Mflop/s) and percentage of STREAM Triad bandwidth, assuming optimal vector reuse, for three matrix kernels and different matrix formats. The test machines are a Core 2 Duo (P8700) and an Opteron 2356 (two sockets). The test problem is a Q_1 finite element discretization with block size of 2 on a 3D mesh (essentially a 27-point stencil) in which each process has $64 \times 64 \times 63$ nodes.

4.2.2 Small dense tensor product kernels

A fundamental operation for computing with high order finite element and spectral element methods is application of a 3D tensor product kernel operations $y \leftarrow (A \otimes B \otimes C)x$, or, in index notation,

$$y_{ijk} \leftarrow A_{i\alpha} B_{b\beta} C_{c\gamma} x_{\alpha\beta\gamma}. \tag{4.2.1}$$

To evaluate a state vector at quadrature points, the Greek indices run over the basis functions in each Cartesian direction and the Latin indices run over the number of quadrature points in that direction. When using a Gauss quadrature that integrates the mass matrix exactly (a typical choice in finite element computations), the range is the same. Additionally, when the approximation order is isotropic (the most common case), all the ranges are the same. The tensor product operation can be written in various ways as dense matrix multiplication (albeit highly non-square) so we expect it to achieve very close to the floating point peak for sufficiently large sizes (e.g. by calling a tuned BLAS3). Increasing the number of fields also increases the problem size which improves the efficiency of BLAS3 calls. The attainable performance for smaller sizes is much less clear, hence we examine the scalar Q_3 case. This is a very practical approximation order: it is high enough to reap some benefits from the regular structure, but not so high as to make resolving geometry overly difficult or to have highly non-uniform resolution (due to clustering of interpolation nodes near the edges of elements as the approximation order increases).

In the scalar Q_3 case, each of A, B, C is 4×4 and the entire operation produces 64 entries in y from 64 entries in x. We have considered a variety of unroll-and-jam optimizations as well as different register blocking strategies. The SSE3 horizontal add instruction `HADDPD` has increased latency and reduced throughput compared to the standard vector addition and multiplication instructions `ADDPD` and `MULPD`, therefore we should try to avoid it whenever possible. Assuming lexicographic ordering, applying the first two parts A and B of the tensor product kernel perform the same operation for all values of γ and thus provide balanced adds and multiplies (which can be issued concurrently as long as the kernel has been unrolled enough to cover the latency) without needing any horizontal operations. This is not possible when applying C because packed loads (and vector arithmetic with memory operands) retrieve consecutive entries. The input could be transposed to avoid this constraint, but the cost for small sizes overwhelms the benefit, therefore we are forced to use one `HADDPD` instruction per packed result $y_{i,j,k:k+1}$ and the reduction also creates some unavoidable (without more registers) pipeline stalls. Application of $A \otimes B \otimes \mathbf{1}$ achieve nearly 80% of floating point peak on the Core 2 Duo while C is reduced to near 60%. The full tensor product contraction $A \otimes B \otimes C$ achieves more than 70% of floating point peak on Intel and about 60% on AMD. This performance is comparable to or better than the best autotuned results of Shin et al. (2010), though they achieve slightly higher performance for larger sizes (which have less overhead). Note that large size tensor contractions are important for quantum chemistry and have thus received more optimization attention (e.g. Kaushik et al., 2008; Hirata, 2003). Contrast the tensor product performance with sparse matrix kernels that attain less than 10% of floating point peak due to low arithmetic intensity.

An old (McCalpin, 2007) but continuing trend in computer architecture is for peak floating point performance to continue to improve via the use of longer registers (Intel's AVX and the Blue Gene/Q architecture have packed 32-byte floating point registers), more cores, and GPU-style vectorization while memory bandwidth increases much more slowly (Keyes, 2011). An additional feature of upcoming high-performance computing architectures is the prevalence of in-order execution; high power requirements no longer justify the luxury of sophisticated out-of-order execution units common on today's commodity hardware (Seiler et al., 2008; Pham et al., 2006). Achieving high performance on such systems requires decomposing the computation into kernels with high arithmetic intensity (so as not to overload the memory subsystem) and sufficient local structure to utilize vector registers and effectively schedule instructions at compile time. We optimized streaming stencil kernels for Blue Gene/P in Malas et al. (2012), including implementation of a static scheduler that we used to rapidly perform loop optimizations with a small assembly-language building block that exploited the available SIMD floating point unit. With this approach, we were able to effectively use all the floating point and general purpose registers to hide instruction latency, delivering 93% of theoretical FPU peak for the 27-point stencil with problem sizes

that fit in level 1 cache and 91% of bandwidth for larger problem sizes (72% of FPU peak). These results are 80% better than the best previously published for Blue Gene/P.

Since the tensor product kernel has sufficient local structure to utilize vector registers, we can estimate its performance on future architectures by computing its arithmetic intensity. For simplicity, suppose that $n = p+1$ is both the number of interpolation nodes and quadrature points on an element in each Cartesian direction, and that b is the number of degrees of freedom per node. Then direction of a contraction performs bn^4 multiplies and $b(n-1)n^3$ additions on the reference element. We approximate the total across all three dimensions $3bn^4$ fused multiply-add (FMA) operations because FMA units are increasingly popular. When evaluating both interpolation and gradient (most general form), it is possible to share partial results so that both can be evaluated in $9bn^4$ FMAs. Translating the gradient from reference to physical element costs another $9bn^3$ FMAs. The Galerkin procedure evaluates these contractions twice per residual or matrix-free Jacobian (once to evaluate the trial function, once more in transpose for the test functions), leading to a total of $18(bn^4 + bn^3) + Qn^3$ FMAs, where Q is the number of operations per quadrature node. Typically we do not store the coordinate transformation, so it must also be computed at a cost of $9 \cdot 3n^4 + 31n^3$ (assume block size 3 and held in an equal order basis, evaluate only on the reference element, then invert the local Jacobian at every quadrature node at a cost of 30 FMA or multiplies and one division). Thus, for a problem with b degrees of freedom per node and no sharing between elements, we have total operation count of $9(3+2b)n^4 + (31+18b+Q)n^3$ with $(b+3)n^3$ floating point values coming in and bn^3 out.

To understand these tradeoffs, we consider three block sizes at different approximation orders and plot the memory bandwidth and number of flops (calculated as twice the number of FMAs) required to compute each entry in the output vector. The performance model for assembled sparse matrices assumes `BAIJ` storage and perfect vector reuse within the cache system. The model for unassembled storage applied by tensor product assumes a general tensor-product quadrature of equal order, that coordinates are stored in a function space of the same order as the independent variables and are needed to compute the coordinate transformation but do not contribute to a result, that the physics can be represented using stored data equivalent in size to a stored gradient plus stored function values, and that the physics can be applied using an operation similar in structure to Newton-linearization of Navier-Stokes with power-law rheology plus a reaction term coupling all components. This is intended to be nearly a worst-case setting for the unassembled tensor product formulation; it is cheaper for simpler physics, simpler coefficients, collocated quadratures, etc.

The memory and floating point costs are shown in Figure 4.1. The unassembled representation is uniformly better in terms of memory use for all orders $p \geq 2$, but has significant floating point overhead for the smallest sizes, especially with small block size. Note that both bandwidth and computation requirements improve for the tensor product formulation when the block size increases (because the overhead of the coordinate transformation is reduced), while they degrade for the assembled representation. The arithmetic intensity for unassembled representations is typically around 8 flops/byte with weak dependence on polynomial order, block size, and even details of the physics (unless a great deal of recomputation is required for linearization, or a nonsymmetric high-rank tensor needs to be stored). This should be compared to the assembled sparse matrix representation which cannot surpass 1 flop/4 bytes even if column indices were not stored. As noted in Section 4.2.1, modern hardware balances computation with memory when the arithmetic intensity is between about 2 and 6 flops/byte and expected to increase. The 30-fold increase in arithmetic intensity while simultaneously reducing memory traffic below even the lowest order representation bodes well for the relevance of unassembled Jacobian representations on future architectures.

4.3 Generic finite elements

Given the performance of unassembled Jacobian representations in Section 4.2.2 and the availability of inexpensive low-order preconditioners presented in Chapter 2, it is natural to ask how much effort is

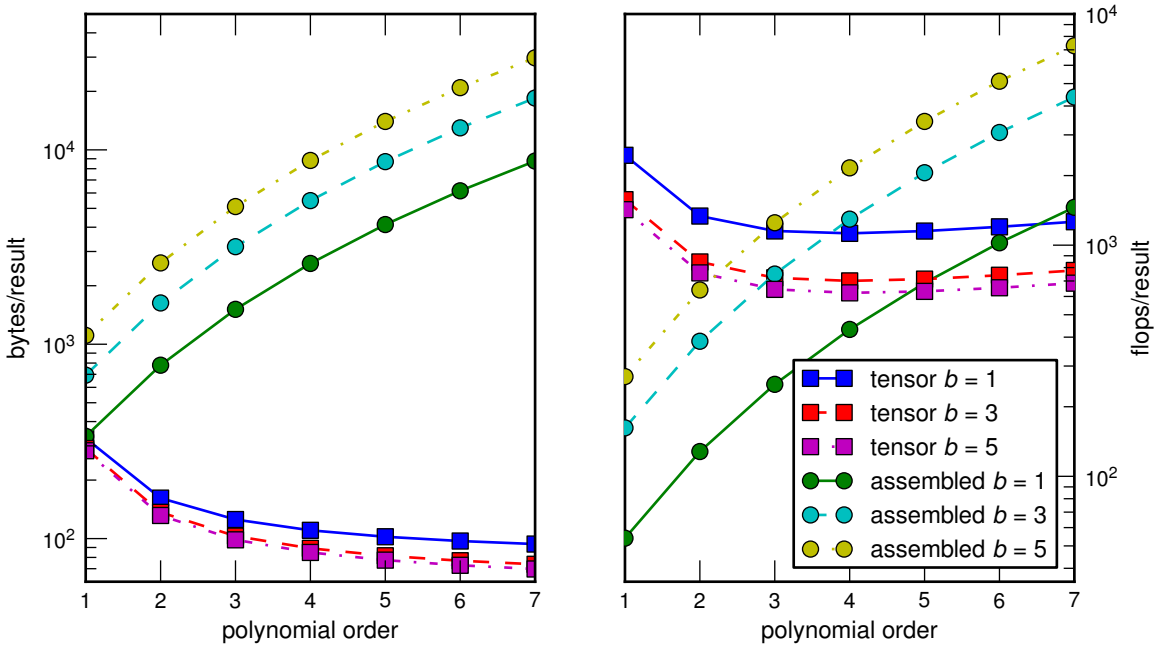

Figure 4.1: Memory and floating point requirements for matrix-free tensor-product application of an operator versus representation as an assembled matrix stored in `BAIJ`(b) format. The same operation $y \leftarrow Ax$ is applied in both cases, the storage is just different. A "result" is a single scalar entry in y, regardless of the block size b.

required to use these methods. If the equations or parameters change so that a specific discretization or solution method is no longer working well, how much effort does it take to change the method? In this section, we outline the approach taken by the Dohp library to facilitate run-time flexibility without sacrificing performance.

4.3.1 Dual order finite elements

Standard finite element methods define both the basis functions and the integration method in terms of a single unit known as an element. In order to use the dual order methods of Chapter 2, we need to define quadrature on smaller units because the low-order approximation space has more locality and less regularity within elements. We will continue to use "element" to refer to the maximal units of structure as with p-version finite element and spectral element methods, but introduce the new term "quadrature patch" for the smaller units. There is always at least one quadrature patch per element. For conventional methods and for evaluating high-order operators in Dohp, there is exactly one quadrature patch per element. Assembly of a dual-order preconditioning matrix requires evaluating a nonlinearity on quadrature points and then assembling an element matrix depending only on those basis functions with support on the quadrature patch. This suggests a specific quadrature rule that is essentially the concatenation of several low-order quadratures that define integration on each quadrature patch, which are sub-elements in this case.

To represent this in software while permitting run-time choice of quadrature method, basis order, and choice of whether to assemble the true operator or low-order approximations, we introduce two small immutable objects, a `Rule` which represents a quadrature rule which may contain multiple quadrature patches and an "element function space" `EFS`. The `EFS` performs interpolation and differentiation, mapping from the degrees of freedom associated with an element to all the quadrature points of a `Rule`, and the

transpose operation. Both `Rules` and `EFSs` are created lazily (as needed) and stored in a cache that is typically shared by all components of an application, but can be managed independently. Because these objects are immutable, they can be safely shared an arbitrary number of times and only a single pointer per element needs to be stored (this can be further compressed for meshes with nearly homogeneous topology). It is possible to have multiple function spaces evaluate on the same `Rule` or to have a single function space evaluate on multiple `Rule`. This flexibility is important for multi-physics problems in which only select pieces are assembled.

An arbitrary amount of specialization is performed when a `EFS` is created so the resulting kernels can be anywhere from completely unrolled to entirely dynamic, with the choice made as a run-time option. Common specialization choices are to unroll completely over the last dimension and to assume even or odd parity in earlier dimensions (for consistent handling of fringes). For lowest order elements, the tensor product formulation offers no savings so evaluation can be implemented with the usual dense matrix-vector product. Since the `EFS` evaluates all the basis functions at all the quadrature points, the granularity is large enough to effectively hide function call overhead for all but lowest order (linear) elements.

Evaluating residuals simply requires pointwise operations at quadrature points, using the current trial vector to define coefficients of test functions at quadrature points. For example, residual evaluation for an elliptic problem with weak form

$$\boldsymbol{v} \cdot \boldsymbol{f}(\boldsymbol{u}) \sim \int_\Omega \nabla v \cdot (1+u^2)\mathbf{1} \cdot \nabla u - v \exp u - v 1 = 0$$

(the bold symbols on the left represent discrete vectors representing global state, the corresponding continuum formulation is on the right) would be implemented as the following pointwise loop where `Q` is the number of quadrature points.[2]

```
for (i=0; i<Q; i++) {
    v[i] = -weight[i] * (exp(u[i]) - 1.);
    for (j=0; j<3; j++)
        dv[i][j] = -weight[i] * (1 + u[i]*u[i]) * du[i][j];
}
```

Matrix-free application of the Jacobian $\boldsymbol{v} \cdot J(\boldsymbol{u})\boldsymbol{w}$ can be implemented as a similar loop, but it requires knowledge of the state $\boldsymbol{u}$ evaluated at quadrature points. The Jacobian is only required after residual evaluation so this state was recently available; suppose that u and ∇u were stashed in the residual evaluation above. In Dohp's implementation, storage for the stash is typically managed by the iterator which associates a user-provided number of bytes with each quadrature point and/or each element. With $\boldsymbol{u}$ and $\nabla \boldsymbol{u}$ provided inside `stash[]`, Jacobian application would look like

```
for (i=0; i<Q; i++) {
    v[i] = -weight[i] * exp(stash[i].u) * w[i];
    for (j=0; j<3; j++)
        dv[i][j] = -weight[i] * (2 * stash[i].u * w[i] * stash[i].du[j]
                                 + (stash[i].u * stash[i].u) * dw[i][j]);
}
```

We can break the above process into three pieces, `PointwiseStash` which may be as simple as saving u and/or ∇u, `PointwiseResidual` which may use the just-computed stashed values, and `PointwiseJacobian` which necessarily uses stashed values to apply the action of the continuous Jacobian. The key to the ability to split the problem up like this is that discretization and linearization commute

[2] C99 variable length array pointers are used to simplify indexing here, `du[i][j]` is effectively translated by the compiler to `du[i*3+j]` so there is no pointer indirection.

for Galerkin methods. This is not generally true for non-Galerkin methods, and not easily rectifiable for methods involving nonlinear reconstruction such as high resolution[3] finite volume schemes.

For preconditioning purposes, we typically also need to assemble some approximation to the Jacobian. The dual order method defines this matrix by modifying the high order basis functions to have more local support (thus improving sparsity), using a quadrature that respects the reduced continuity these more local basis functions, and exploiting the improved sparsity by assembling a separate element stiffness matrix for each of these smaller quadrature patches. This traversal has a natural interpretation in Dohp's language. A `Rule` is defined that represents all the quadrature patches on an element. The actual storage retains the tensor product structure on the full element so we create an `EFS` that evaluates the trial function and its gradient on the quadrature points using the true basis functions, and it will be implemented by efficient tensor product kernel. These values are used to evaluate nonlinearities at quadrature points, coefficient filtering and other techniques can also be done. The true basis functions are also used to define geometry, thus the geometry can reside in an arbitrarily high order space without impacting the fidelity of gradients or volumes.[4]

We create another `EFS` that evaluates the low-order basis functions at the quadrature points in the patch. While this `EFS` is capable of all the same operations described above, we do not typically use that functionality. Instead, we get an explicit representation of the basis functions and derivatives for each quadrature patch, as well as the node indices needed to insert values into the matrix using the `EFSGetExplicitSparse` function. This is exactly what is needed for matrix assembly, but it is messy to make the user roll the loop over quadrature patches and manage associated memory, therefore we hide the hierarchy in our iterator. The "patch" assembly loop for any scalar second-order equation is

```
for (dInt q=0; q<Q; q++) {                        // Loop over quadrature points
  struct UserStash stash;
  PointwiseStash(param,x[q],u[q],du[q],&stash); // Full accuracy nonlinearity
  for (dInt j=0; j<P; j++) {                      // Loop over trial functions
    dScalar v,dv[3];                              // Test function coefficients
    PointwiseJacobian(param,&stash,weight[q],interp[q][j],deriv[q][j],&v,dv);
    for (dInt i=0; i<P; i++) {                    // Loop over test functions
      K[i][j] += (interp[q][i] * v
                  + deriv[q][i][0] * dv[0]
                  + deriv[q][i][1] * dv[1]
                  + deriv[q][i][2] * dv[2]);
    }
  }
}
FSMatSetValuesLocal(fs,Jp,P,lidx,P,lidx,&K[0][0],ADD_VALUES);
```

where `PointwiseJacobian` applies the action of the continuous Jacobian at the current "stashed" state to perturbations $w =$ `interp[q][j]` and $\nabla w =$ `deriv[q][j][:]`. This generalizes readily to vector problems and an arbitrary number of function spaces, see the examples in the Dohp source tree.

With this software design, the physics is isolated from the discretization, therefore a great deal of run-time flexibility can be provided. Some possible choices that are automatically supported by client applications of Dohp include

- Use the high-order method for everything including the assembled Jacobian. This is the typical method used by p-version finite element methods.

[3] In the hyperbolic community, "high resolution" refers to methods that have a high order of accuracy in regions where the solution is smooth. The actual solutions usually contain discontinuities in which case genuine high *order* accuracy is not possible.

[4] Higher order quadratures would be required to evaluate the isoparametrically mapped elements, but we typically do not raise the integration order. This has not yet posed a difficulty and is standard practice in isogeometric analysis and extended finite element methods, neither of which use exact integration.

- Define residuals and the assembled matrix using the high-order method. Use the assembled matrix to form a preconditioner, but apply the action of the Jacobian by tensor product because it is cheaper (see Figure 4.1).
- Define residuals with the high-order method, assemble preconditioning matrix with low-order (and either low- or high-order definition of nonlinearity), apply the true Jacobian matrix-free. This is the method promoted in Chapter 2.
- As above, but use the low-order assembled matrix as the "Jacobian" inside of a defect correction scheme. This is a sort of modified Newton method which has relatively poor convergence properties (certainly not quadratic).
- Define residuals and the matrix in terms of the low-order approximation. This does not have high-order convergence properties, but has local structure which reduces the overhead of a fully unstructured mesh and permits vectorization. It is practical for problems where coefficients are rough on a finer scale than geometric features in the domain.
- Standard low-order finite element method on an unstructured mesh. This is practical for problems where the geometry of the domain has finer structure than coefficients.

To demonstrate that this generic formulation does not severely degrade performance for low-order approximations, we compare Dohp's assembly cost to that of the well-known finite element libraries `libMesh` (Kirk et al., 2006) and `Deal.II` (Bangerth et al., 2007). Table 4.2 shows the assembly time for a Laplacian on a cube using all three libraries. Example 4 was used from the latest versions of both `libMesh` and `Deal.II`. For the `libMesh` example, the assembly loop was modified to only compute the matrix without contributions to the right hand side or adaptivity, and to use a low order quadrature rule when integrating Q_1 elements. For `Deal.II`, constants in the code needed to be changed to change the mesh resolution and element order, and timing calls (using `clock_gettime(2)`) needed to be added. The Dohp results use the `ellip.c` example which solves the "p-Bratu" equation

$$-\nabla\cdot\left(|\nabla u|^{p-2}\nabla u\right) - \lambda \exp u = f$$

where f is a manufactured forcing term. The Laplace equation is the special case $p = 2, \lambda = 0$, which is chosen at run-time in the example and produces the timing results shown in the p-Bratu column. Timing with a modified version of `ellip.c` specialized for the homogeneous Laplacian is also included so that the overhead of including nonlinearity can be seen. It is clear that Dohp is faster than `libMesh` and `Deal.II` for everything but the assembled Q_2 case, including more than twice as fast for unstructured Q_1 elements. This was a surprise considering that Dohp has no special optimization for unstructured Q_1 elements. Dohp uses a very naive method for assembly of Q_k elements when $k > 1$, so there remains significant opportunity for optimizing the assembly of Q_2 discretization. However, we consider assembly (and solves) for such elements to be excessively expensive unless it is absolutely necessary for an application. The overhead of using an unstructured mesh can be seen by comparing to the `DMDA` column which uses a structured grid and is written specifically for Q_1 elements. The unstructured Q_1 case requires 30% overhead, and drops below 7% when using a structured representation for each block of 4^3 Q_1 elements (Q_4 connectivity).

For multi-physics problems and mixed finite element formulations, there are multiple function spaces and perhaps several quadrature rules in use at once. Dohp always stores coordinates in a bona fide function space so the mesh geometry can naturally be made a part of the solution or not. The order of approximation can be chosen independently for each function space, including for coordinates. When an iterator is created, an arbitrary number of function spaces can be added. When the iterator is started, one input vector and one output vector per function space may be provided (either one can be `NULL`). For each vector, the user specifies whether it should reside in a function space with homogeneous Dirichlet conditions or an affine function space with inhomogeneous Dirichlet conditions. Dirichlet conditions are typically enforced by the library on a user-specified part of the boundary using (5.2.1). The API to start

Library	Mesh	Connectivity	Discretization	Laplace (s)	p-Bratu (s)
`libMesh`	64^3	Q_1	Q_1	4.8	—
`libMesh`	32^3	Q_2	Q_2	5.0	—
`Deal.II`	64^3	Q_1	Q_1	7.8	—
`Deal.II`	32^3	Q_2	Q_2	4.9	—
Dohp	64^3	Q_1	Q_1	2.32	2.58
Dohp	32^3	Q_2	Q_1	2.06	2.35
Dohp	32^3	Q_2	Q_2	5.44	5.78
Dohp	16^3	Q_4	Q_1	1.91	2.22
`DMDA`	64^3	structured	Q_1	1.79	—

Table 4.2: Assembly time for the Laplace and p-Bratu problems with different element types and libraries. All cases have the same number of degrees of freedom, $65^3 = 274625$. The structured problem is treated as unstructured by `libMesh`, `Deal.II`, and Dohp, with granularity shown in the Connectivity column. `DMDA` uses a structured grid and is specialized for Q_1 assembly.

an iterator and to evaluate test and trial functions uses variadic functions so it is easy to add more spaces without needing to use an excessively verbose or fine-grained API.

Finally, we mention that this software design naturally supports isogeometric analysis (IGA) (Cottrell et al., 2009) which replaces the Lagrange interpolants with splines, usually non-uniform rational b-splines (NURBS). Advantages of IGA relative to conventional finite element methods include higher order continuity (especially important for equations beyond second order such as shell elements for structural mechanics or the Cahn-Hilliard equation for separation), the ability to represent certain geometries exactly (especially those created by computer-aided design software, thus simplifying the meshing process), and non-negative basis functions which enable robust definitions of normals for conservative slip and better control over positivity. Disadvantages include more nonzero entries in the assembled matrices and more complex implementation. To support IGA under the framework above, we identify a "NURBS patch" with Dohp's "element" and IGA's "element" with Dohp's "quadrature patch". The implementation in Dohp will work after definition of the spline bases, though it would require additional code to support especially large NURBS patches, such as those spanning multiple processors.

4.3.2 Input/output and visualization for high-order mixed spaces

Although there has been some work visualizing high-order basis functions (Schroeder et al., 2005, 2006; Üffinger et al., 2010), the code has not been released and it is not supported by the popular visualization tools such as VTK (Schroeder et al., 1998), Paraview (Henderson et al., 2004), and VisIt (Childs and Miller, 2006). Due to this deficiency, visualization of a model that uses high-order elements requires losing the high-order information by either representing each element as a structured grid or by replacing it with many smaller unstructured elements. Experience from other high-order projects (Fischer et al., 2008) indicated that VisIt (and presumably other software) had poor scaling when thousands or millions of "grids" were defined, therefore Dohp chose to simply represent everything using unstructured low-order elements. It is simple for an application to write it's entire state into an unstructured file, but the connectivity for lowest order elements is expensive to store and it is no longer possible to recover the high-order state. Because there were no extensible formats for storing high-order mesh data, I made a new HDF5-based (HDF5 Group, 2011) format for use with Dohp, with read and write functionality for function spaces and vectors using PETSc's Viewer component. My intent was for the format to be simultaneously usable as checkpoints in forward simulation, as part of time-dependent adjoint solves, and for visualization, without the need for custom support for each application.

This format stores unstructured time-dependent multi-domain multi-field data. The mesh topology is contained in MOAB's format (Tautges et al., 2004), although any iMesh implementation would, in

principle, be sufficient. The tags defining the function space are stored only once, unless the connectivity or distribution changes. A function space has a reference to the geometry it is defined on. For problems with moving meshes, the geometry evolves, but not the topology (unless remeshing is needed, this is not yet implemented), so we avoid storing duplicate copies. Similarly, for arbitrary vectors defined on function spaces, only the vector values are stored at each step. The actual output for vectors is typically done using collective or independent MPI-IO (Corbett et al., 1995) accessed through HDF5.

For each scalar or vector field, function spaces keep track of the name, units, and an arbitrary scaling factor. The scaling factor is needed for bitwise exact recovery of serialized state when the model works with non-dimensionalized fields, otherwise rounding error contaminates the low bits. This is bitwise-reproducible file IO is a stated objective of the Community Earth System Model (Collins et al., 2006) and other projects, despite the fact that parallel implicit solvers cannot produce bitwise-reproducibile results on a different number of processors due to the non-associativity of floating point arithmetic (Goldberg, 1991) and algorithmic changes resulting from domain decomposition (Smith et al., 1996).

I implemented a reader plugin for VisIt to read this format. It works by reading in the function space and vector state (as requested by the VisIt client) and placing the topology and field data in VTK data structures. The implementation is a very thin C++ layer atop the standard C library. If at some point in the future, VisIt adds native support for high-order elements, the storage format would not need to change and the VisIt plugin would need only minor changes to support the new API. The field and mesh names are visible in the VisIt graphical user interface. Determining what fields are defined on a given part of the domain at a given time is not an especially simple task using HDF5's hierarchical structure. In particular, it involves walking the hierarchy in different ways to determine what is fundamentally *relational* data. Since these queries could be answered faster and more extensibly in one-line SQL queries instead of about a page of C code, it seems likely that a simpler, more user-friendly, and possibly more performant format could be defined using a small relational database for relational metadata and plain binary blobs for everything else.

4.4 Verification

How reliable are the results of a numerical simulation? Under what circumstances can decisions be made based on the results of a calculation? Answering these questions involves two distinct steps known as Verification and Validation (Babuška and Oden, 2004; Roache, 1998). Verification is the purely mathematical endeavor of determining if a computational model obtained by discretizing a mathematical model of a physical process can be used to represent the mathematical model with sufficient accuracy—*solving the equations right*. It is an essential prerequisite for Validation which is the process of assessing whether a mathematical model is a sufficiently accurate model of a physical process—*solving the right equations* (Roache, 1998). To quote Babuška and Oden (2004),"any validation exercise that is based on a computational model in which discretization error is not quantified is futile, because modeling and approximation error are then intertwined in an indecipherable way." The distinction, and uncertainty quantification in general, has been largely overlooked by numerical modeling efforts in glaciology. Many problems in geophysics, and especially in glaciology, lack accurate observations of material parameters, boundary conditions, or geometry, and thus, can only rigorously be modeled as stochastic processes. The field of stochastic PDEs (Deb et al., 2001; Ghanem and Spanos, 2003; Chow, 2007) is however, quite young, and despite promising work on challenging prototype application problems (Asokan and Zabaras, 2006; Ganapathysubramanian and Zabaras, 2007; Zabaras and Ganapathysubramanian, 2008; Mishra et al., 2011), is not yet mature enough for analysis of ice flow problems. Therefore, we are limited to deterministic models in which uncertainty in material parameters, boundary conditions, and geometry are handled in a more heuristic way. Inability to measure and/or control these sources of uncertainty, as well as the diversity of measurement types, each with its own (often sparse) spatial and temporal distribution, makes validation an ongoing process that can never truly be completed. Verification of a deterministic numerical model for a known class of input parameters, on the other hand, is a process that

can be completed.

It is useful to distinguish between verification of code and verification of a calculation (Roache, 2002). The first is concerned with demonstration that all terms, including boundary conditions and source terms, are implemented correctly and exhibit the designed order of accuracy for the method. It is a check for consistency between the analysis and the implementation. Verification of a calculation involves the estimation of uncertainty for a particular problem, usually by demonstrating mesh independence and the use of a posteriori error estimators (Ainsworth and Oden, 1997). The former can be done using exact solutions (provided they exercise all terms and options in the code), but exact solutions are generally not available for the latter.

This section is devoted to code verification and in particular, the design of software so that the model is readily verifiable. We focus on the method of manufactured solutions (MMS) (Roache, 2002) which has been shown to be an extremely robust methodology. The method relies on the simple observation that the correctness of a numerical method for discretizing and solving a nonlinear system

$$F(u) = b$$

must be independent of the source term $b(x,y,z,t)$, and in particular, that there is no requirement that b have physical significance. Therefore, for the purpose of code verification, we need not obtain analytical solutions with physically realizable source terms. This is fortuitous because such solutions are nearly impossible to find for complex problems. Instead, MMS chooses a *solution* $u(x,y,z,t)$, independent of the domain and even the physics F. There is no need for u to have any similarity to reality, but it should possess a sufficient number of derivatives for the strong form of the PDE or variational problem F to be valid. Ideal candidates involve transcendental functions such as tanh and log, shifted or scaled so as to break any possible symmetries. Once u has been chosen, a compatible forcing term b is found using analytical methods. This does not involve discretization and can be readily performed with symbolic algebra systems. With the chosen u and symbolically computed b in hand, we have an analytic solution to $F(u) = b$ that has no degeneracies and no symmetries, thus exercising all terms in the equations.

At this point, we choose a domain and boundary conditions, and a sequence of discrete approximation spaces. The discrete problem is solved using the manufactured forcing term b and the discrete solution is compared to the exact solution in whichever norms the user is interested in. If the numerical approximation converges at the rate predicted by the analysis, the code has been verified and the process moves on to verification of calculations and validation. The only requirements MMS imposes on the code is that it be possible to solve with arbitrary user-provided forcing terms and inhomogeneous boundary conditions, and to evaluate the error as compared to a user-provided solution (Roache, 2004).

It is argued by Roache (2002) that this methodology constitutes an "engineering proof" to the extent that the chance of a computer program delivering correct orders of convergence with even one manufactured solution, whilst possessing implementation errors affecting the quality of solution is vanishingly small. Indeed, Knupp and Salari (2002) conducted a blind study in which one author sabotaged a previously verified Navier-Stokes solver (compressible and incompressible, steady and unsteady) developed by the other. Every introduced error affecting quality of solution was detected by MMS. A pointwise error degrading the order of a boundary condition at a single corner node a domain is still visible in the global error and correspondingly degrades the global error. For the purpose of verifying a calculation (in which the answer is not known), MMS provides confidence in the assertion that if a the code is self-convergent under grid refinement, then it is converging to the correct solution. In other words, numerical evidence that a sequence of computational results on successively refined approximation spaces is *Cauchy* becomes a convincing argument that the sequence converges to the correct solution of the governing equations, and even at the rate predicted by the theory.

As an example, we consider large deformation elasticity and implement the weak form in Python using the SymPy (Certik et al., 2011) symbolic algebra package as

```
def weak_form(u, du, v, dv):
  I = eye(3)                    # Identity tensor
```

```
  F = I - du                      # Deformation gradient
  E = (F.T*F - I)/2               # Green-Lagrange tensor
  S = lmbda*E.trace()*I + 2*mu*E  # Second Piola-Kirchoff tensor
  Pi = F * S                      # First Piola-Kirchoff tensor
  return dv.dot(Pi)
```

where `lmbda` and `mu` are material parameters and a St. Venant-Kirchoff constitutive model has been used (neo-Hookean and other models are handled simply by changing the definition of `S`). Further discussion of finite element methods for large-deformation elasticity can be found in Wriggers (2008) or Bathe (1996). The homogeneous weak form above is symbolically differentiated to produce a strong form of the governing equations. A manufactured solution is defined in Python as

```
def solution(x,y,z, a,b,c):
  return Matrix([cos(a*x) * exp(b*y) * z + sin(c*z),
                 sin(a*x) * tanh(b*y) + x * cosh(c*z),
                 exp(a*x) * sinh(b*y) + y * log(1+(c*z)**2)])
```

with free parameters `a`, `b`, and `c`. The strong form of the governing equations is then applied to this exact solution, with spatial differentiation performed symbolically, and a forcing term is generated. The forcing term for this model involves more than a thousand transcendental functions, but C code for its evaluation is automatically generated and does not need to be examined. The same governing equations are discretized by the C code, the manufactured forcing term is incorporated in the residual, and errors in several norms are computed once the nonlinear solve has converged. This methodology is implemented for all PDE examples in Dohp.

We continue with the large-deformation elasticity problem using manufactured solution described above and solve the problem on a sequence of refined meshes with different orders. Continuous norms of the discrete errors $\boldsymbol{u}_h - \boldsymbol{u}$ are shown in Table 4.3. These norms are estimated using a standard quadrature, as was used to integrate the manufactured forcing term. The expected convergence rate for a Q_k discretization of this problem is of order $k+1$ in L^p and of order k for the gradients. The noisiness in the sup norms are due to the inexact quadrature since neither the exact solution nor the manufactured forcing term can reasonably be integrated exactly. The highest order cases are also sensitive to iterative solver tolerance and floating point rounding error, but it is clear in each case that the expected order of accuracy is being attained. The computed solution on the 4^3 mesh with Q_5 elements is shown in Figure 4.2. We note that the attained accuracy of this method for $\|\nabla \boldsymbol{u}_h - \nabla \boldsymbol{u}\|_2$, which is equivalent to the strain, of $2.59 \cdot 10^{-5}$ would require a 100000^3 mesh with Q_1 elements (10^{15} nodes) and a 400^3 mesh with Q_2 elements (half a billion nodes), but the computation runs in a few seconds on a single core with Q_5 elements.

There are some implementation errors that MMS will not detect. Such mistakes do not affect the quality of the solution, but they may affect the speed at which convergence is achieved. This is especially clear when one considers the recommended way to write a new PDE code in which implicit methods are used. The first step is to implement the nonlinear residual. This nonlinear system is solved by matrix-free Newton-Krylov methods and the solution can be verified using MMS. At this point, the residual will never be modified again, therefore programming and/or mathematical in pursuit of *getting a solution fast* will not affect the quality of the solution that is eventually obtained. If the nonlinear solve converges, the solution will have whatever accuracy the underlying discretization provides. Of course unpreconditioned methods are not fast so practical simulation requires further effort, which is typically the most complicated and error-prone part of an application. This is not tested by MMS, but consistency can be checked in other ways, such as by comparing to an explicit Jacobian computed using finite differences, by confirming that application of an "exact preconditioner" results in convergence in one or two or three iterations as predicted by theory (e.g. Murphy et al., 2000; Ipsen, 2001), or by confirming that a Newton method is converging quadratically.

Finally, a note on automation. The Python function `weak_form` contains the governing equations, therefore it need not be necessary to manually write the same equations in C. Indeed, it would be very little effort to

Table 4.3: Convergence rates for the large-deformation elasticity problem with manufactured solution.

			$\|\boldsymbol{u}_h - \boldsymbol{u}\|_2$		$\|\boldsymbol{u}_h - \boldsymbol{u}\|_\infty$		$\|\nabla\boldsymbol{u}_h - \nabla\boldsymbol{u}\|_2$		$\|\nabla\boldsymbol{u}_h - \nabla\boldsymbol{u}\|_\infty$	
Mesh		# Nodes	Error	O	Error	O	Error	O	Error	O
Q_1	1^3	8	1.79e+00	—	6.50e-01	—	3.70e+00	—	1.08e+00	—
Q_1	2^3	27	5.49e-01	1.71	3.40e-01	0.93	1.61e+00	1.20	6.92e-01	0.64
Q_1	4^3	125	1.53e-01	1.84	1.26e-01	1.43	8.01e-01	1.01	4.51e-01	0.62
Q_1	8^3	729	3.94e-02	1.96	3.73e-02	1.76	3.98e-01	1.01	2.81e-01	0.68
Q_1	16^3	4913	9.95e-03	1.99	1.01e-02	1.88	1.98e-01	1.01	1.57e-01	0.84
Q_1	32^3	35937	2.49e-03	2.00	2.61e-03	1.95	9.92e-02	1.00	8.32e-02	0.92
Q_2	1^3	27	2.44e-01	—	1.82e-01	—	9.48e-01	—	4.60e-01	—
Q_2	2^3	125	3.71e-02	2.72	4.47e-02	2.03	2.86e-01	1.73	1.54e-01	1.58
Q_2	4^3	729	4.48e-03	3.05	6.23e-03	2.84	6.94e-02	2.04	4.34e-02	1.83
Q_2	8^3	4913	5.60e-04	3.00	9.31e-04	2.74	1.74e-02	2.00	1.29e-02	1.75
Q_2	16^3	35937	7.01e-05	3.00	1.23e-04	2.92	4.34e-03	2.00	3.52e-03	1.87
Q_3	1^3	64	4.14e-02	—	2.71e-02	—	2.90e-01	—	1.63e-01	—
Q_3	2^3	343	2.06e-03	4.33	2.06e-03	3.72	2.39e-02	3.60	1.14e-02	3.84
Q_3	4^3	2197	1.81e-04	3.51	2.06e-04	3.32	4.23e-03	2.50	2.88e-03	1.98
Q_3	8^3	15625	1.22e-05	3.89	1.87e-05	3.46	5.79e-04	2.87	5.84e-04	2.30
Q_5	1^3	216	3.76e-03	—	2.90e-03	—	4.69e-02	—	3.16e-02	—
Q_5	2^3	1331	7.58e-05	5.63	5.92e-05	5.61	1.62e-03	4.86	1.05e-03	4.91
Q_5	4^3	9261	7.33e-07	6.69	6.61e-07	6.48	2.59e-05	5.97	1.76e-05	5.90
Q_7	1^3	512	4.46e-04	—	3.59e-04	—	8.15e-03	—	5.83e-03	—
Q_7	2^3	3375	2.95e-06	7.24	2.95e-06	6.93	8.21e-05	6.63	6.05e-05	6.59
Q_7	4^3	24389	7.65e-09	8.59	1.07e-08	8.11	4.09e-07	7.65	3.95e-07	7.26
Q_9	1^3	1000	5.81e-05	—	5.04e-05	—	1.42e-03	—	1.05e-03	—
Q_9	2^3	6859	6.27e-08	9.86	7.59e-08	9.38	1.63e-06	9.77	1.60e-06	9.36

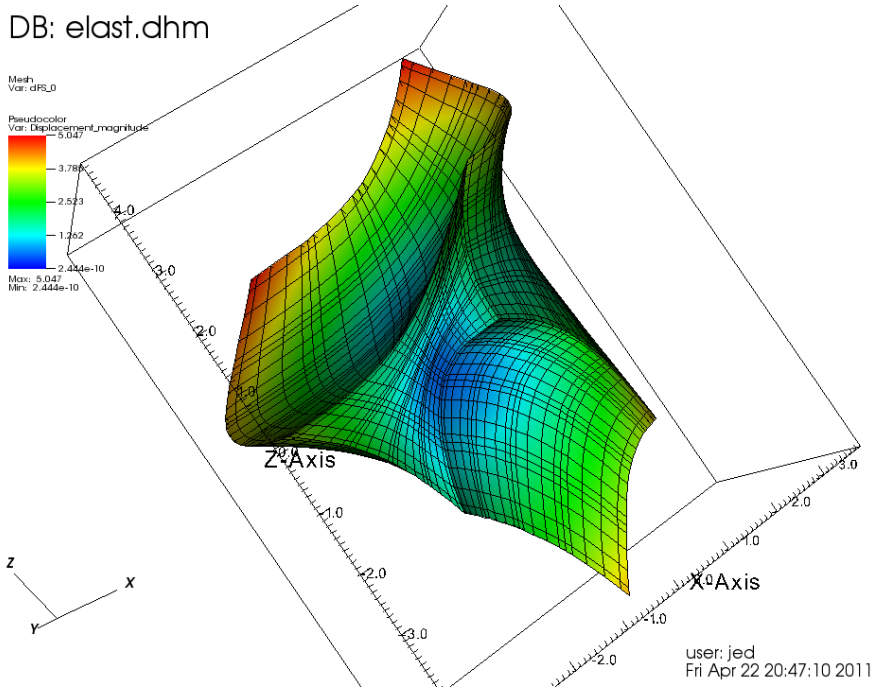

Figure 4.2: Solution of the large-deformation nonlinear elasticity problem with manufactured solution on a Q_5 mesh.

automatically generate C code from a weak form written in Python, in much the same spirit as has been done by the FEniCS project (Logg et al., 2011). For example, it would be possible to use physical models implemented in the "unified form language" (UFL) (Alnaes et al., 2009), but generate pointwise physics kernels for Dohp instead of the assembly code used by DOLFIN (Logg and Wells, 2010). UFL is based on Python and looks very similar to the symbolic `weak_form` above, but includes specification of the function space. The FEniCS project considers that to be an advantage because it permits optimizations that involve mixing the discretization with the physics, such as tabulation of element matrices on the reference element and mapping them to physical elements by dense matrix-matrix product (Kirby et al., 2005). While such methods can provide substantial speedup for simple physics on affine elements (DOLFIN only handles simplicial meshes), the method becomes untenable for more general nonlinear problems on non-affine meshes, and a quadrature representation must be adopted. As seen by comparing the matrix assembly performance of Dohp to the structured grid with inlined Q_1 discretization in Table 4.2, there is very little benefit to mixing physics with discretization when using a quadrature representation. Separating physics from discretization, as Dohp's interface encourages, has the benefit that element types can be changed at run-time (skipping possibly time-consuming compilation steps) and that meshes of mixed topology and/or element order can be handled transparently. Automation is frequently convenient, but it is the author's opinion that in order to manage complexity and maintain extensibility, that the automation process must be as transparent as possible. Furthermore, the C API for which code is to be automatically generated must be sufficiently clean that a human can easily use it. The ability to "get dirty" and incrementally explore the entire software stack is important.

The specification of elements can also be automated. The FEniCS project makes it very easy to specify new elements on simplicial topologies using the formalism of Ciarlet (1978), and implemented in the Finite Element Automatic Tabulator (FIAT) (Kirby, 2004, 2006). The FIAT implementation could be extended to handle more general element topologies, but would need non-trivial modification to support preservation of structure like tensor products. A unification of isogeometric analysis (e.g. NURBS) and extended finite element basis functions is presented by Benson et al. (2010), with a simple way to tabulate basis functions for a variety of element types including shells, with complex continuity conditions such as T-splines (Sederberg et al., 2003; Bazilevs et al., 2010). Once again, this tabulation does not

retain structure such as tensor products and thus gives up the algorithmic and performance advantages of preserving the structure. However, the philosophy of these recent papers appears useful in a specification capable of preserving structure like tensor products without impacting genericity.

4.5 Time integration

Most problems in glaciology have elliptic constraints that apply at all times. Therefore, semi-discretization in space results in a system of differential algebraic equations (Hairer and Wanner, 2010) instead of ordinary differential equations. I converted the implicit methods in PETSc's TS package to work with differential algebraic equations, and implemented some recently developed general linear methods from Butcher (2006); Butcher et al. (2007).

Differential algebraic equations (DAE) are written in implicit form

$$F(t, \boldsymbol{x}, \dot{\boldsymbol{x}}) = 0, \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{4.5.1}$$

where the nonlinear function F represents the spatial discretization for PDE problems. If the matrix $F_{\dot{\boldsymbol{x}}}(t) = \partial F / \partial \dot{\boldsymbol{x}}$ is nonsingular then it is an ODE and can be transformed to the standard explicit form, although this transformation may not lead to efficient algorithms. For ODE with nontrivial mass matrices such as arise in FEM, the implicit/DAE interface significantly reduces overhead to prepare the system for algebraic solvers by having the user assemble the correctly shifted matrix, therefore it is also useful to solve ODE systems by writing them in the implicit form (4.5.1). Preconditioning is usually necessary for stiff systems and requires an approximation to the Jacobian

$$J = F_{\boldsymbol{x}} + a F_{\dot{\boldsymbol{x}}}$$

which is the Jacobian of $G(\boldsymbol{x}) = F(t, \boldsymbol{x}, \boldsymbol{z} + a\boldsymbol{x})$ where $G(\boldsymbol{x}) = 0$ is the nonlinear system solved internally by the time integrator. Internally, the DAE integrator approximates $\dot{\boldsymbol{x}}$ as $\boldsymbol{z} + a\boldsymbol{x}$ for some vector $\boldsymbol{z}$ and scalar "shift" a. For example, the implicit Euler method uses $\dot{\boldsymbol{x}} = (\boldsymbol{x} - \boldsymbol{x}_-)/h$ which is $\boldsymbol{z} = -\boldsymbol{x}_-/h$ and $a = 1/h$ where $\boldsymbol{x}_-$ is the value of the solution at the beginning of the time step and $h = \Delta t$ is the time step size. Other methods have more complicated expressions for $\dot{\boldsymbol{x}}$ in these implicit systems (e.g. using several previous steps) and may combine the results of these implicit solves in different ways.

Two important properties for DAE and singularly perturbed ODE with mixed diffusive and transport phenomena are A-stability which ensures stability for time steps with length independent of parameters and L-stability which ensures that high-frequency oscillations are rapidly damped (to prevent "ringing" as seen when, e.g. the trapezoid rule is used for stiff diffusive processes). Additionally, all stages should be evaluated to some minimum order of accuracy, known as the *stage order*, to prevent order degredation. Most conventional methods for stiff systems are either linear multi-step or Runge-Kutta methods. These classes have certain undesirable properties such as Dahlquist's second barrier which precludes A-stable linear multi-step methods of order greater than 2 and that diagonally implicit Runge-Kutta (DIRK) methods have stage order at most 1. A traditional alternative is to use singly implicit Runge-Kutta (SIRK) methods, but these place absicassa outside the time step interval which tend to produce poor results in the presence of bifurcations. Fully implicit Runge-Kutta methods such as the Radau schemes (Hairer and Wanner, 1999) are highly successful when used with direct solvers, but must solve with all stages coupled together which increases the dimension of the Krylov space. The fully implicit Lobatto schemes have attractive stability properties and are self-adjoint which is desirable for adjoint sensitivity analysis (Sandu, 2006; Sandu et al., 2003). There appears to be significant room for algorithmic development using fully implicit Runge-Kutta methods with parallel iterative solvers.

General linear methods provide access to methods of arbitrary order with A and L-stability while retaining diagonally implicit structure. One such example is the family of general linear methods with "inherent Runge-Kutta stability" (Wright, 2002; Butcher, 2006). The methods from this family that are designed for stiff systems have A- and L-stability as well as stage order equal to classical order. In addition, they

come with asymptotically correct error estimates for the current method and methods of order one higher. General linear methods can be written in a tablea similar to Runge-Kutta methods

$$\begin{bmatrix} Y \\ X^{n+1} \end{bmatrix} = \begin{bmatrix} A & U \\ B & V \end{bmatrix} \begin{bmatrix} h\dot{Y} \\ X^n \end{bmatrix}.$$

where the Nordsieck vector

$$X = \{x_1, \dots, x_r\} = \{x, h\dot{x}, h^2\ddot{x}, \dots\}$$

is passed between steps. Because A is lower triangular, the stage values $Y = \{y_1, \dots, y_s\}$ can be computed sequentially be solving $F(t_i^n, \boldsymbol{y}_i, \dot{\boldsymbol{y}}_i) = 0$ with $\dot{\boldsymbol{y}}_i$ written in terms of $\boldsymbol{y}_i$ using the top block. I implemented these methods for DAE in the TS component of PETSc. Methods of orders 1 to 5 are available and new methods can be added by providing the entries of the tableau $\left(\begin{smallmatrix} A & U \\ B & V \end{smallmatrix}\right)$. The error estimators and rescale-and-modify scheme for changing step size are computed automatically using the methods in Butcher et al. (2007). An adaptive-order adaptive-step controller is available with a plugin architecture for extending or using user-provided controllers.

Unfortunately, despite great properties on paper, the tableaus published in Butcher and Podhaisky (2006); Podhaisky (2006) seem to have rather noisy error estimates and poor monotonicity properties. The great freedom in optimizing coefficients for these metheds is both a blessing and a curse since the additional free parameters make the tableaus difficult to optimize. It is likely that more robust tableaus will be published in the future which will make this implementation more practical. Since all linear multistep and Runge-Kutta methods are general linear methods, they can be used simply by providing the tableau for the desired scheme.

For hyperbolic problems such as transport phenomena discretized using a method of lines approach, it is important for the ordinary differential equation solvers to be Strong Stability Preserving (SSP) (Gottlieb et al., 2009). The SSP property, when combined with a spatial discretization that is TVD (LeVeque, 2002) or TVB (e.g. (weighted) essential non-oscillatory, Shu (2003)), ensures that the full discrete method is also TVD or TVB (Gottlieb et al., 2001). For linear multistep and Runge-Kutta methods, it is known that implicit methods of order greater than 1 have a time step restriction in order for the SSP property to be satisfied (Spijker, 1983) and it is conjectured that this holds for general linear methods and that the maximum coefficient is only twice as large as for explicit methods (Gottlieb et al., 2009), therefore explicit methods appear to be more practical when SSP methods are required. I implemented a new family of optimal explicit SSP methods of second, third, and fourth order with various numbers of stages (Ketcheson, 2008). These methods are low-memory in the sense that memory usage is independent of the number of stages (usually two vectors), but methods with more stages have better effective CFL coefficients allowing longer stable time steps. These have been extremely practical methods.

Chapter 5

Discretization issues

The flow of ice sheets and especially outlet glaciers like Jakobshavn Isbræ presents several challenges to mathematical analysis and numerical methods. The "shallow" continuum forumlations traditionally used in glaciology, as well as the stronger results in mathematical analysis and convergence of numerical methods, require assumptions that do not hold for more demanding ice flow problems. Weaker assumptions that remain physically valid for the problems of interest require substantially more technical analysis. This analysis often takes place on the "purer" side of applied mathematics, and is thus quite far removed from glaciology. This section surveys the known results on several technical matters and explains how they relate to problems of interest in glaciology.

5.1 Regularity and approximation

Realistic bathymetry posesses little regularity, thus, at every resolution that could be used for a numerical model, the slip boundary will be "rough". If the roughness is smoothed significantly, then bathymetric features such as the deep channel at Jakobshavn Isbræ will be under-resolved unless an excessively fine mesh is used. But if a slip boundary is rough on the same scale as the mesh, it becomes critical that the discretization preserve local conservation across the interface exactly instead of merely up to some mesh-dependent truncation error.

Recall the non-Newtonian Stokes problem which is the preferred kinematic model for the flow of ice streams and outlet glaciers,

$$-\nabla\cdot(\eta D\boldsymbol{u}) + \nabla p - \boldsymbol{f} = 0 \tag{5.1.1a}$$

$$\nabla\cdot\boldsymbol{u} = 0 \tag{5.1.1b}$$

with nonlinear viscosity

$$\eta(\gamma) = B(\theta,\dots)\left(\varepsilon+\gamma\right)^{\frac{\mathfrak{p}-2}{2}} \tag{5.1.2}$$

where $D\boldsymbol{u} = \frac{1}{2}\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T\right)$ is the strain rate, $\gamma(D\boldsymbol{u}) = \frac{1}{2}D\boldsymbol{u}:D\boldsymbol{u}$ is the second invariant, $\mathfrak{p} = 1+\frac{1}{\mathfrak{n}} \approx \frac{4}{3}$ in terms of Glen's (1955) exponent $\mathfrak{n}$, B is a hardness parameter depending on enthalpy θ and perhaps other variables (e.g. grain size, dust content, damage), and ε is the second invariant of a reference strain rate that provides regularization to prevent viscosity from becoming infinite.

Common boundary conditions for (5.1.1) include $\boldsymbol{u} = \boldsymbol{0}$ at a frozen bed, $\eta D\boldsymbol{u} - p\mathbf{1} = 0$ at the free surface, and $\eta D\boldsymbol{u} - p\mathbf{1} = -\rho_w z\boldsymbol{n}$ at the ice-ocean interface underneath an ice shelf where $\boldsymbol{n}$ is the unit outward-facing normal. Open boundary conditions are also needed for regional models, but should be applied in places where the shallow ice approximation is accurate, thus causing the flow to be defined by local geometry. A final, and much more difficult boundary condition, is slip at the bed. Slip is a Dirichlet

condition on the normal component and a nonlinear Robin condition on the tangent components,

$$\boldsymbol{u}\cdot\boldsymbol{n} = \boldsymbol{g}_{\text{melt}}(T\boldsymbol{u},\dots) \tag{5.1.3a}$$

$$T(\eta D\boldsymbol{u} - p\mathbf{1})\cdot\boldsymbol{n} = \boldsymbol{g}_{\text{slip}}(T\boldsymbol{u},\dots) \tag{5.1.3b}$$

where $T = \mathbf{1} - \boldsymbol{n}\otimes\boldsymbol{n}$ is a projector into the tangent space. Melt rate and basal traction generally depend on a basal hydrology model, involve spatially-variant parameters, and are coupled because sliding produces heat at a rate $T\boldsymbol{u}\cdot(\eta D\boldsymbol{u} - p\mathbf{1})\cdot\boldsymbol{n}$. The precise form of the sliding relation is a subject of extensive debate, but is often taken to have the form

$$\boldsymbol{g}_{\text{slip}}(T\boldsymbol{u},\theta,\cdots) = \beta_m(\theta,\cdots)\,|T\boldsymbol{u}|^{m-1}\,T\boldsymbol{u}$$

where $m = 1$ is Navier slip, $m = 1/3$ is the popular "Weertman sliding" (Weertman, 1957) and $m \to 0$ is the Coulomb limit. See Iverson et al. (1998) for empirical support of the Coulomb limit and Schoof (2006a,b, 2007) for analysis of the associated variational inequalities. Some continuum models for basal hydrology are discussed in Flowers and Clarke (2002a,b); Johnson and Fastook (2002), but basal processes are poorly understood and outside the scope of the present work, see Clarke (2004) for a review.

Classical solutions to the linear constant-coefficient Stokes problem with frictional slip boundary conditions do not exist in general. For example, in addition to C^2 continuity on Dirichlet boundaries, Saito (2004) required much higher C^4 continuity for slip surfaces; see also the analysis by Fujita (2002). Since the boundaries of interest for our problems are merely Lipschitz and change type abruptly, we cannot use the strong form (5.1.1). To discuss regularity issues at boundaries and our discretization, we need the weak form which is formally obtained by introducing test functions $\boldsymbol{v}, q$ and integrating by parts. Given a Lipschitz domain $\Omega \subset \mathbb{R}^3$ and open subsets Γ_s and Γ_b of the boundary $\partial\Omega$ which we identify as "surface" and "non-frozen bed", the problem is to find $(\boldsymbol{u}, p) \in \boldsymbol{W}_D^{1,\mathfrak{p}}(\Omega)\times L^2(\Omega)$ such that

$$\int_\Omega D\boldsymbol{v} : \eta\mathbf{1} : D\boldsymbol{u} - q\nabla\cdot\boldsymbol{u} - p\nabla\cdot\boldsymbol{v} - \boldsymbol{v}\cdot\boldsymbol{f} - \int_{\Gamma_b}\boldsymbol{v}\cdot(\eta D\boldsymbol{u} - p\mathbf{1})\cdot\boldsymbol{n} = 0 \tag{5.1.4}$$

for all $(\boldsymbol{v}, q) \in \boldsymbol{W}_0^{1,\mathfrak{p}}(\Omega)\times L^2(\Omega)$. Readers unfamiliar with Sobolev spaces may think of $W^{1,\mathfrak{p}}$ simply as the space with sufficiently well-behaved first derivatives. The subscripts in $W_D^{1,\mathfrak{q}}, W_0^{1,\mathfrak{q}}$ indicate that inhomogeneous and homogenous Dirichlet boundary conditions respectively are built into the space. That is, all components of velocity are specified in regions where the bed is frozen and normal components are specified at places where sliding may take place. The implementation of Dirichlet conditions is somewhat different from its definition here and will be discussed later. For well-posedness, it remains to specify $(\eta D\boldsymbol{u} - p\mathbf{1})\cdot\boldsymbol{n}$ on Γ as an algebraic function $\boldsymbol{g}(\boldsymbol{u})$ to enforce stress conditions on tangent components at slip surfaces and on all components at free surfaces in which case $\boldsymbol{g}$ is independent of $\boldsymbol{u}$. Provided the surface Γ_s is non-empty, the pressure is uniquely determined so there is no constant null space to remove, c.f.Section 2.4.2. Although it is not used directly in our work, (5.1.4) corresponds to the minimization of the viscous energy $\int_\Omega \frac{1}{2} D\boldsymbol{u} : \eta\mathbf{1} : D\boldsymbol{u}$ over the subspace where $\nabla\cdot\boldsymbol{u} = 0$. In particular, it is the first variation of the Lagrangian obtained when pressure p is introduced as a Lagrange multiplier to enforce the constraint. Boundedness of the Lagrangian requires coercivity of the viscous energy term which follows from Korn's inequality[1] which controls $\|\boldsymbol{u}\|_{W^{1,\mathfrak{p}}}$ using the symmetric gradient $\|D\boldsymbol{u}\|_{L^{\mathfrak{p}}}$ and an inf-sup condition to control pressure using divergence of velocity (Evans, 1998; Brenner and Scott, 2008). These conditions are easily satisfied by the continuum spaces, but they place important restrictions on the discrete spaces, an issue which we revisit in Section 5.1.2.

5.1.1 Singularities in the continuum formulation

The transition from no-slip to slip boundary conditions is exactly analogous to mode II and III fracture in nonlinear elasticity theory, transition from no-slip to an unconstrained stress condition (e.g. floating) is

[1] See Kondrat'ev and Oleinik (1988) for Korn's inequality in $L^{\mathfrak{p}}$. I am not aware of a priori error estimates for finite element methods that apply for solutions in $W^{1,\mathfrak{p}}$, but expect that similar results to the $\mathfrak{p} = 2$ case hold.

the mode I case. In the case of linear rheolgy, this is the classical inverse square root stress singularity $\sigma \sim r^{-1/2}$ in fracture mechanics (Anderson, 2005), where r is the distance from the transition, see Erdogan and Biricikoglu (1973) for two bonded materials. For nonlinear rheology, the same energy estimates (Rice, 1968) apply and the singularity becomes $|\sigma| \sim r^{(1-\mathfrak{p})/\mathfrak{p}}$ and $|D\boldsymbol{u}| \sim r^{-1/\mathfrak{p}}$ as shown by Rice and Rosengren (1968); Hutchinson (1968). These functions are all integrable and the singularity does not pose a fundamental regularity problem for the continuum equations, but the velocity in this latter case behaves as $|\boldsymbol{u}| \sim r^{(\mathfrak{p}-1)/\mathfrak{p}}$ which is a fourth root for the typical $\mathfrak{p} = 4/3$, thus difficult to approximate with a polynomial basis. Note that in reality, there is not a true singularity because of friction and plastic failure in the immediate vicinity of the transition, but micro-scale physical processes are fundamentally different, therefore the meso-scale asymptotics are most relevant when designing an approximation space.

The singularity is stronger for the heat production term $\sigma : D\boldsymbol{u}$, of order $1/r$ regardless of rheology. Enthalpy cannot have infinite slopes because there is always physical diffusion, but the approximation problem is more difficult because of the need to represent a "spike" instead of a "kink". The most important case for glaciology is mixed mode II and III which occurs at the margins of ice streams, the thermal structure of which was investigated in Jacobson and Raymond (1998); Raymond (2000).

Reentrant corners in the ice domain caused by incompletely resolved bathymetry are the source of the other important singularity. For second order elliptic equations, solutions around reentrant corners of angle $\omega > \pi$ have singularities of order $r^{\pi/\omega} \sin \frac{\pi\phi}{\omega}$ where $(r, \phi), r < 0, 0 < \phi < \omega$ are polar coordinates centered at the corner, see Grisvard (1985); Nazarov and Plamenevskiĭ (1994), also Bacuta et al. (2003) which has new sharp finite element convergence estimates. In the strongest case $\omega \to 2\pi$, this singularity becomes $r^{1/2}$ which is the same as a crack in linear media. Nonlinear rheology is analogous and indeed, the singularity for flow past a reentrant corner is never worse than for a transition from no-slip to free slip.

Although not considered here, viscoelastic flows have the further difficulty that the Weissenberg number blows up at viscous stress singularities, see Lipscomb et al. (1987); Davies (1988); Hinch (1993); Owens and Phillips (2002).

5.1.2 Approximation spaces

The Banach space $W^{1,\mathfrak{p}}$ in (5.1.4) has the correct regularity for investigating the singularities discussed in the last section, but many approximation results are not known in this setting, therefore we will fall back to the smaller space H^1 when necessary. Approximation spaces in the present work will always be piecewise polynomial so this is not as egregious as it would be with non-polynomial bases. Ainsworth and Kay (1999) have proven an interesting result for a class of problems including the $\mathfrak{p}$-Laplacian (a useful model problem for power-law fluids): in the presence of reentrant corner singularities, p-version finite element methods attain twice the order of accuracy of h-version methods that use any fixed order p. I believe that the rational bases used in isogeometric analysis (Hughes et al., 2005; Cottrell et al., 2009) and physics-adapted bases in extended finite element methods (Belytschko et al., 2009; Mohammadi, 2008; Elguedj et al., 2006; Jiang et al., 2011) are very promising and should be explored for treating several boundary singularities in the ice flow problem.

Finite element methods choose a discrete subspace $\mathcal{V}_D, Q$ of the continuous trial space $\boldsymbol{W}_D^{1,\mathfrak{p}} \times L^2$ from (5.1.4), discrete spaces for the test functions, and a way of approximately[2] evaluating integrals. The present work considers only Galerkin methods for which case the test and trial spaces are equivalent except for inhomogenous boundary values. Galerkin methods for elliptic problems come with a property called Galerkin orthogonality which asserts that the error in a discrete approximation is within a constant of the minimum error within the discrete space. In other words, the PDE is solved to within a constant of the pure approximation problem for the exact solution. This is a powerful property and not generally available for nonsymmetric problems or non-Galerkin methods. Error estimators, adaptivity, and uncertainty

[2]Inexact quadrature is a "variational crime" (Brenner and Scott, 2008), but nonlinear rheology produces terms that cannot reasonably be integrated exactly.

quantification for both smooth and non-smooth problems are also most mature in the Galerkin context, see Ainsworth and Oden (1997); Matthies and Keese (2005); Babuška et al. (2005); Barth et al. (2011).

We consider two classes of finite element space defined on hexahedral meshes. The first is spanned by a tensor product of 1D polynomials of degree $k \geq 1$ that have been pushed forward from the reference cube $[0,1]^3$ to the physical element. This has a variant Q_k that is continuous between elements and a discontinuous variant Q_k^{disc}. The other is a P_k^{disc} which is spanned by polynomials of maximum degree k and is discontinuous between elements.

Stability of the Galerkin approximation depends on discrete versions of Korn's inequality and the inf-sup condition. Korn's inequality bounds the $W^{1,\mathfrak{p}}$ norm in terms of the $L^{\mathfrak{p}}$ norm of symmetric gradient and is easily satisfied by C^0 vector-valued spaces such as Q_k. The inf-sup condition is more troublesome. Given a velocity space $\mathcal{V}$ and pressure space Q, the inf-sup constant

$$\beta = \inf_{p \in Q} \sup_{\boldsymbol{u} \in \mathcal{V}} \frac{\int p \nabla \cdot \boldsymbol{u}}{\|p\| \, \|\boldsymbol{u}\|} \tag{5.1.5}$$

is a measure of how well the velocity space spans the pressure space. If β is bounded below by a positive constant as the mesh is refined, then the finite element method will converge at an optimal rate. More precisely, given discrete solutions $(\boldsymbol{u}_h, p_h) \in \mathcal{V} \times Q$ with inf-sup constant β and exact solution $(\boldsymbol{u}, p)$, the velocity and pressure errors satisfy the a priori estimate (Brezzi and Fortin, 1991)

$$\eta \, \|\boldsymbol{u} - \boldsymbol{u}_h\|_{H^1} \leq C \left[\frac{\eta}{\beta} \inf_{\boldsymbol{v} \in \mathcal{V}} \|\boldsymbol{u} - \boldsymbol{v}\|_{H^1} + \inf_{q \in Q} \|p - q\|_{L^2} \right] \tag{5.1.6a}$$

$$\|p - p_h\|_{L^2} \leq \frac{C}{\beta} \left[\frac{\eta}{\beta} \inf_{\boldsymbol{v} \in \mathcal{V}} \|\boldsymbol{u} - \boldsymbol{v}\|_{H^1} + \inf_{q \in Q} \|p - q\|_{L^2} \right]. \tag{5.1.6b}$$

The H^1 norm is used here because I am not aware of an analogous result in $W^{1,\mathfrak{p}}$. This shows that the error in the discrete solution is bounded in terms of how well the true solution can be represented in the discrete space. For problems posed in anisotropic domains or containing thin boundary layers, the true solution is most efficiently represented by highly anisotropic meshes, therefore we desire uniform inf-sup stability with respect to aspect ratio.

For many choices of $\mathcal{V} \times Q$, such as using the same basis functions, β is zero meaning there is a discrete pressure mode that is untested by the velocity space, thus the Stokes system is singular. Other choices, such as $Q_1 - P_0^{\text{disc}}$ have positive values of β on most grids, but β decays under mesh refinement so spurious pressure modes appear and convergence rates suffer (Brenner and Scott, 2008; Chapelle and Bathe, 1993; Babuška and Narasimhan, 1997). It is important to recognize that if the pressure space contains the piecewise constant functions, the constraint equation will force the discrete velocity field $\boldsymbol{u}$ to be exactly divergence-free when integrated over an element e,

$$\int_e \nabla \cdot \boldsymbol{u} = \int_{\partial e} \boldsymbol{u} \cdot \boldsymbol{n}.$$

This local conservation property is important for free surface flows, density-driven flows, and for long time integration. Note that although it is possible to solve incompressible flow problems by introducing stabilization (e.g. residual-based (Hughes et al., 1986) or polynomial projection (Dohrmann and Bochev, 2004)), these formulations sacrifice local conservation and perform poorly on problems with sharp structure.

In the following, let $d \in \{2,3\}$ be the number of spatial dimensions. A particularly useful and robust element pair is $Q_k - P_{k-1}^{\text{disc}}$ where $k \geq 2$, for which local conservation holds and the inf-sup constant is uniformly bounded with respect to mesh resolution and independent of the polynomial degree (Bernardi and Maday, 1999). This result also holds for non-affine hp-adaptive meshes under modest adaptivity assumptions (Schieweck, 2008), see also Matthies and Tobiska (2002). For smooth solutions, the $Q_k - P_{k-1}^{\text{disc}}$ element produces velocity and pressure errors of order k in H^1 and L^2 respectively, which is optimal.

Unfortunately, the inf-sup constant β degrades as $O(\varepsilon^{1/2})$ where $\varepsilon > 0$ is the aspect ratio of the mesh. While this is acceptable for some fluid dynamics problems, it is unusable for those geophysical flows in which extreme aspect ratio is inherent in the problem, as well as for wall-resolved large eddy simulation where boundary layer elements have aspect ratio on the order of 10^{-6}. An alternative is $Q_k - Q_{k-2}^{\text{disc}}$ which is locally conservative and has uniform inf-sup stability independent of aspect ratio (Stenberg and Suri, 1996; Schötzau et al., 1998), albeit with $O(k^{\frac{1-d}{2}})$ dependence on polynomial degree (Maday et al., 1992). On affine meshes with corners and/or hanging nodes, this k-dependence is sharp in 2D (Schötzau et al., 1999), but degrades to $k^{-3/2}$ in 3D (Toselli and Schwab, 2003). We are not aware of an analogous result for non-affine meshes. This element is more "squishy" inside elements than with P_{k-1}^{disc} pressure, and has suboptimal order of accuracy $k-1$ in H^1 for velocity and in L^2 for pressure due to poor representation of the pressure space exhibited in the rightmost terms in (5.1.6).

For high-aspect ratio meshes around corners, the uniform inf-sup stability of $Q_k - Q_{k-2}^{\text{disc}}$ for edge-refined meshes is lost and $\varepsilon^{1/2}$ dependence appears (Schötzau et al., 1998). In two dimensions, Ainsworth and Coggins (2000) propose ways to improve inf-sup stability for edge and corner refinement. In the first case, if $Q_k - P_{k-1}^{\text{disc}}$ is augmented by increasing the polynomial degree of the velocity in the direction tangent to the stretched edge element to $k+1$, uniform inf-sup stability is recovered. For corners, the $\varepsilon^{1/2}$ dependence is due to a single pressure mode which is also critical for representing the solution near the corner, therefore the velocity space must once again be augmented. A single velocity mode that stabilizes the problematic pressure mode is identified by Ainsworth and Coggins (2000), thus achieving uniform inf-sup stability independent of aspect ratio and polynomial degree. Unfortunately, this mode is only defined for affine corner macroelements and is thus difficult to use in practice. The brute-force method of simply raising the polynomial degree with the aspect ratio, $k \sim \varepsilon^{-1/2}$, improves the stability to $\beta \sim \varepsilon^{1/4}$ (or even to $\beta \sim (1+\log^{1/2} k)^{-1} \min(1, k\sqrt{\varepsilon})$ if the velocity space is allowed to grow slightly faster than the pressure space, though this result is less practical). To our knowledge, this analysis has not been extended to three dimensions, but remains the best known result for highly anisotropic meshes.

"Rotated" non-conforming elements were proposed in Rannacher and Turek (1992) that have uniform stability on affine meshes (Becker and Rannacher, 1994). These methods are amenable to very efficient coupled multigrid solvers (Rannacher, 2000). Unfortunately, these elements do not satisfy a discrete Korn's inequality and thus need complex edge stabilization techniques (Brenner, 2004; Turek and Ouazzi, 2007). This stabilization impacts sparsity and makes the multigrid solution process more complicated (Turek et al., 2002; Ouazzi and Turek, 2006). Note that stabilization is not required if one uses the "vector Laplacian" formulation for viscous stresses instead of the proper formulation in terms of symmetric gradient. While it is inexpensive and does not require stabilization, the vector Laplacian formulation cannot be used for variable viscosity and violates objectivity when used with certain boundary conditions (Limache et al., 2008).

5.2 Implementation of boundary conditions

5.2.1 Dirichlet boundary conditions

In the continuum context, Dirichlet boundary conditions are built into the approximation space and thus do not explicitly appear in the weak form. There are many ways to implement Dirichlet boundary conditions in the discrete context including removal from the ansatz space, penalties, "lifting" the known part to the right hand side for the linear problem, zeroing rows of the Jacobian, and zeroing both rows and columns by suitable evaluation of the residual. Most of these methods perform similarly for simple problems and simple preconditioners, but have serious deficiencies for more difficult problems and sophisticated solvers. When possible (e.g. the Dirichlet part of the domain is not itself part of the solution such as occurs with a frozen bed or contact problem), removal of Dirichlet unknowns is a robust solution, but it introduces some complexity in managing vectors residing in the ansatz space versus vectors residing in the closure (used for output and visualization) and prevents direct addressing of neighbors in structured grid computation.

Removal of Dirichlet degrees of freedom is the standard way to enforce Dirichlet conditions in Dohp, the function space provides access to the state as part of a homogeneous or inhomogeneous space. When removal is not practical, other approaches requiring user involvement must be used.

Penalties are easy to implement, but a penalty parameter must be chosen which contributes to ill-conditioning which reduces the accuracy of the solution (ability to converge to very high tolerance), can contaminate Schur complements, requires the Krylov method to work in the preconditioned norm (usually means left preconditioning instead of right), and must always be paired with a preconditioner that corrects the contribution from the penalty. "Lifting" can be performed directly on the linear system and is performed transparently using PETSc's `PCRedistribute`, but the method does not compose well with field-split and hierarchical preconditioners, and the sparse matrix manipulations require extra memory and time. Simply replacing rows of the Jacobian matrix with rows of the identity is easy to perform independent of the element assembly loop. Unfortunately, it destroys symmetry and pollutes the spectrum of the operator which has very problem-dependent effects on the iteration count.

Zeroing both the rows and columns corresponding to Dirichlet degrees of freedom is the best alternative when the degrees of freedom are not eliminated, but there are several implementation details to consider. Assembling into the matrix without observing boundary conditions and zeroing afterward is not efficient with sparse matrix representations since columns are difficult to address in a compressed row format (and rows are difficult to address for compressed column formats). A better approach is to discard contributions to those rows and columns during insertion of the element stiffness matrix and then simply place the diagonal entry afterward. This requires a compatible residual evaluation which we now consider.

Let $\mathcal{V}_D$ be a discrete ansatz space with inhomogeneous Dirichlet boundary conditions implicitly built in, $\mathcal{V}_0$ be the corresponding space with homogeneous conditions, $\mathcal{V}_\Gamma$ be the trace space on the Dirichlet boundary, and $\overline{\mathcal{V}} = \mathcal{V}_0 \times \mathcal{V}_\Gamma$ the space of all functions in the finite element space defined on the closure of the domain. We define three projectors on $\overline{\mathcal{V}}$, R_0 projects to the $\mathcal{V}_0$ subspace of $\overline{\mathcal{V}}$ with zero values on the boundary $\mathcal{V}_\Gamma$, R_D projects to the affine subspace $\mathcal{V}_D$, and R_Γ projects to the trace space $\mathcal{V}_\Gamma$ with zero in the interior. We can now write the discrete residual F in terms of the "interior" residual f that does not recognize boundary conditions as

$$F(u) = R_0 f(R_D u) + \alpha R_\Gamma (u - R_D 0). \tag{5.2.1}$$

The value of the possibly spatially varying scaling factor $\alpha \neq 0$ does not affect the correctness of the formulation, but weighting it to be of similar magnitude to nearby diagonal entries in the matrix (e.g. by using local viscosity and the mesh size) is preferred to improve the conditioning of the linear system including boundary conditions and more importantly when geometric multigrid is used with rediscretized (non-Galerkin) coarse level operators. The Jacobian of (5.2.1) isolates the Dirichlet degrees of freedom from the rest of the system. The implementation of (5.2.1) and its derivative is straightforward, local element residuals and Jacobians are evaluated with correct Dirichlet values imposed on the state u, the result is inserted with contributions to Dirichlet nodes discarded. Then a loop over the boundary places the α-scaled difference from the correct boundary values into the residual vector and inserts α on the diagonal of the Jacobian.

5.2.2 Slip

Slip boundary conditions are a combination of Dirichlet on the normal component and (usually nonlinear) Robin on the tangent component. When the slip surface is curved, there are multiple ways to define the normal direction. Geometric averages such as those advocated by Walkley et al. (2004) can be very accurate and appear to be preferable for problems with surface tension in which conservation is not essential. When exact conservation is critical, there is no choice but to use "conservative normals" (Lynch and Gray, 1980). Conservative normals are defined at an arbitrary node (or mode) i in which the basis function ϕ_i has support on the boundary Γ by

$$\boldsymbol{n}_i = \frac{\int_\Gamma \phi_i \boldsymbol{n}}{\left|\int_\Gamma \phi_i \boldsymbol{n}\right|}. \tag{5.2.2}$$

If the velocity field is constrained so that $\boldsymbol{u}_i \cdot \boldsymbol{n}_i = 0$ for a node i with support on the boundary, then node i will contribute zero flux through the boundary. We enforce this condition for all boundary nodes so the total flux is also zero across the boundary: $\int_\Gamma \boldsymbol{u} \cdot \boldsymbol{n} = 0$. Note that we do not in general have that $\int_f \boldsymbol{u} \cdot \boldsymbol{n} = 0$ for all mesh faces f for the same reason that continuous Galerkin methods are not locally conservative. However, a weaker local conservation statement similar to that in Hughes et al. (2000) still holds.

There is a technical difficulty observed by Walkley et al. (2004) when using conservative normals with inf-sup stable spaces. In particular, the velocity space must be at least quadratic for stability reasons and the corner basis functions of P_2 triangles (appearing on the surface of a tetrahedral mesh) have the property that

$$\int_\Gamma \phi_i = 0.$$

This means that the normal need not be constrained on any flat element face (but having no constraint admits non-physical recirculation within elements) and that for isoparametrically mapped elements with small face curvature, the definition of the normal becomes unstable due to near cancellation and can flip direction as the surface evolves. This problem does not occur for non-deformed quadrilaterals (on the surface of a hexahedral mesh), but may arise for sufficiently deformed elements. The extra constraint on element quality is inconvenient for moving mesh simulations and for meshing complex structures, but it is a limitation that we accept in the present work. An interesting alternative is to eschew Lagrange interpolants in favor of non-negative bases such as Bernstein polynomials or more generally, the spline bases used in isogeometric analysis (Cottrell et al., 2009); see Akkerman et al. (2011) for recent results with free surface flows.

When implementing the normal constraint in slip boundary conditions, it is not practical to remove the normal component from the ansatz space so we prefer to leave them in as described in Section 5.2.1. To enforce Dirichlet conditions on the normal component, we rotate coordinates in the global vector so that the normal component is isolated. The rotation is undone at the element level so that local operations need not be aware that the solution vector contains rotated blocks. This rotation affects the construction of coarse level spaces in multigrid and domain decomposition methods. The ML (Gee et al., 2006) algebraic multigrid package allows the user to specify low-energy modes to be represented in the coarse space. These modes should respect the change of basis so that their energies remain low. The impact on domain decomposition methods such as FETI-DP is more delicate, see Klawonn and Rheinbach (2007b); Klawonn and Widlund (2006); Dohrmann and Widlund (2010) for details on the construction of coarse spaces.

While use of conservative normals provides exact conservation across the interface, there is no guarantee that it will preserve realistic steady states, particularly for free surface flows where the hydrostatic contribution to pressure cannot be removed. Indeed, with a smooth curved boundary and a level surface, the momentum residual will not point in the same direction as the conservative normal. The tangent component of the momentum residual is a spurious tangent force that causes non-physical recirculation within the fluid domain that can not be prevented simply by adding artificial friction to the slip surface. This issue is addressed in Behr (2004) which we follow below. Recall the boundary integral appearing in the weak form of the Stokes problem (5.1.4)

$$-\int_\Gamma \boldsymbol{v} \cdot (\eta D\boldsymbol{u} - p\mathbf{1}) \cdot \boldsymbol{n} \tag{5.2.3}$$

where well-posedness of the continuous weak form requires specifying the stress $(\eta D\boldsymbol{u} - p\mathbf{1}) \cdot \boldsymbol{n}$ as an algebraic function of $\boldsymbol{u}$. We are only concerned with the tangential part of the stress since the normal components have Dirichlet conditions imposed. The approach of Behr (2004) integrates (5.2.3) "as is", without applying any boundary condition at all. This has the effect of extending the PDE to include the boundary. It is completely invalid for the continuum problem, resulting in non-uniqueness since the solution to the same PDE on any extended domain with any applied boundary conditions is also a solution on the initial domain, but turns out to be valid in the discrete context. The idea was introduced in the context of open boundary conditions for natural convection in Papanastasiou et al. (1992) and

later refined in the restricted context of outflow boundary conditions for advection-diffusion by Griffiths (1997) and Renardy (1997). In particular, Griffiths (1997) showed that the boundary condition produces $O((h+1/\mathrm{Pe})^{p+1})$ errors in L^∞ for mesh size h, Peclet number Pe, and finite elements of polynomial degree p, a result distinctly better than the $O(h^{p+1}+1/\mathrm{Pe})$ obtained for Neumann outflow conditions.

Chapter 6

Steady-state viscous heat transport at Jakobshavn Isbræ

Jakobshavn Isbræ is representative of many similar outlet glaciers which control the response of the Greenland and Antarctic ice sheets to changing climate. Understanding the effect of changing boundary conditions, especially ocean conditions near the grounding line and subglacial processes influenced by surface melt, is critical for predictive modeling of ice sheets. Models that do not support polythermal ice cannot describe the refreezing process and are thus thermodynamically inconsistent. Additionally, the viscosity model cannot be dependent on moisture content, and will thus appear too viscous in regions where moisture is present. At Jakobshavn Isbræ, the stress and strain rate is sufficiently high to melt a thick layer near the bed (Funk et al., 1994), then in the last few kilometers, the ice rises several hundred meters to the grounding line. The pressure change associated with that rise causes an increase in the pressure melting temperature of about 0.4 °C, leaving the ice supercooled. This refreezing of internal moisture may cause healing of ice that was damaged by the very high strain rates encountered in the region. Along with the supercooled ice, basal water is also becoming supercooled as it flows up the incline, which may lead to refreezing at the bed, sealing basal cracks.

This process involving polythermal ice and basal hydrology suggests a possible explanation for the seasonal calving cycle at Jakobshavn. From observations over the past few years (Joughin et al., 2008; Amundson et al., 2008), there is no calving until late February or March when rapid calving begins. This is long before surface temperatures are above freezing, and the calving stops abruptly in August, before surface melt has ceased. Sea ice breaking up offers a possible mechanism for the rapid breakup observed in early spring (Joughin et al., 2008; Amundson et al., 2008, 2010), but does not explain why calving ceases in late summer, before sea ice arrives and while air temperature is still quite warm. I propose that increased surface melt leads to more supercooled flow up the bed near the incline, thus more effectively sealing basal cracking and healing damaged ice. The presence of frozen rocks on the top of inverted icebergs during our 2008 field campaign supports the hypothesis that significant refreezing occurs at the bed. Although this hypothesis is not discussed further in the present work, a model capable of testing this hypothesis would require a polythermal ice flow model and a basal hydrology model. The former is considered in more detail.

Funk et al. (1994) provided an early model of polythermal ice at Jakobshavn Isbræ. This was a 2D flowline model using the shallow ice approximation for momentum balance and assuming steady state flow conditions, but with high vertical resolution and explicitly tracking the cold-temperate interface (CTS) which is assumed to be a single-valued function of horizontal position, as in Hutter (1982); Greve (1997). More recent work with polythermal ice has used an enthalpy method to avoid the need to explicitly track the interface (e.g. Aschwanden and Blatter, 2009; Aschwanden et al., 2011) and includes more complete momentum balance than the shallow ice approximation. Enthalpy-based methods are well-established for heat transfer problems with phase change (e.g. Shamsundar and Sparrow, 1975; White, 1982) and have been used for geophysical problems such as magma dynamics (Katz, 2008) which is has very similar form to polythermal ice. Such methods are simpler than explicitly tracking the CTS because only a single field

is needed and no jump conditions need to be evaluated inside the domain. Our formulation, explained in Section 6.1 is distinct in that it tracks total energy density instead of enthalpy, thus allowing a system that is conservative due to its geometric structure, thus making conservation more feasible to enforce up to rounding error (instead of truncation error) in numerical methods.

For predictive modeling, we need to determine a thermal field that is in general, non-equilibrium (informed by reconstructed climate and borehole measurements) and compatible with current flow conditions, such that forward modeling does not involve an initial non-physical transient. This problem of determining a compatible thermal field is coupled to that of determining basal boundary conditions. These problems have not yet been solved together, but more principled inverse methods have been applied to the latter problem. Bayesian inferrence is an especially elegant approach to inverse problems in glaciology, but unfortunately, the usual formulation (e.g. Tarantola, 2005) involves dense matrices which prevent the use of scalable algorithms. Consequently, use of these techniques (Gudmundsson and Raymond, 2008; Raymond and Gudmundsson, 2009), while offering theoretical insight, could not be applied to full-scale problems. An alternative approach is the use of deterministic inversion methods initially developed for optimal control problems (Büskens and Maurer, 2000) for which there are efficient, scalable solution algorithms (Akçelik et al., 2006). MacAyeal (1992, 1993) introduced adjoint-based methods for parameter inversion to glaciology, and more recent work (Johnson et al., 2004; Morlighem et al., 2010) have scaled the methods up to more complete continuum models and larger problem sizes. Related methods have been used by Heimbach and Bugnion (2009) to estimate the sensitivity of total ice volume to uncertain model inputs (basal, surface, and initial conditions).

6.1 Formulation and methods

We consider the problem of computing a steady-state energy (in the form of temperature, moisture, and kinetic energy) field coupled to a Stokes problem with rheology depending on strain rate, pressure, temperature, and melt fraction. This problem is most important as part of inversion for a temperature field that is compatible with the observed geometry, flow field, borehole temperature measurements, and any other field observations. Time-stepping a transient model to steady state is extremely expensive, therefore it will never be feasible as part of the functional to be minimized by an optimization algorithm. The implicit method used for the steady-state prbolem can also be used to solve the transient system with time steps of arbitrary length, with no CFL stability limitation. The transient system produces algebraic systems that are somewhat easier to solve numerically.

6.1.1 Problem description

We choose a formulation for polythermal ice that is unconventional in glaciology, but is similar to that used for compressible gas dynamics and other hyperbolic and viscous conservation laws (Liu, 2000; Toro, 2009; LeVeque, 2002). Instead of writing evolution equations for primitive (directly observable) quantities such as velocity or temperature, we write evolution equations only for conserved quantities such as total momentum and energy. When integrated over a reference cell, this formulation leads to an exact conservation statement in terms of integrals through cell faces. If a discretization containing the piecewise constants is used, the amount of each conserved quantity will be written using numerical evaluation of these integrals, with an equal amount flowing out of one cell and into another. This leads to a discretization that is locally conservative for *topological* reasons, independent of accuracy or constitutive relations. The fluxes are identified with physical processes such as thermal diffusion or viscosity, for which we provide constitutive relations. Constitutive relations are usually written in terms of primitive variables, so we must solve an equation of state to determine the primitive variables from the conserved quantities. Any equation of state can be used, provided that the primitive variables are a single-valued function of the conserved variables. This is typically the case, but would not be if, for example, the ice/melt mixture was not assumed to be in local thermodynamic equilibrium. Extending the present model

to avoid the assumption of local thermodynamic equilibrium would require additional state variables (e.g. independent ice momentum and melt momentum), such that the primitive variables once again became a well-defined function of the state variables. As usual in the study of conservation laws, our exposition begins by stating the conservative structure of the equations to be solved, then identifying the physical meaning of the fluxes, and finally "closing" the system by stating constitutive relations and an equation of state.

Given a Lipschitz domain Ω with surface Γ_s and time interval $(0,\tau)$, the strong form is to find total momentum, pressure, and total energy density $(\rho\boldsymbol{u}, p, E) \in W^{1,p} \times L^2 \times H^1$ such that

$$(\rho\boldsymbol{u})_t + \nabla\cdot(\rho\boldsymbol{u}\otimes\boldsymbol{u} - \eta D\boldsymbol{u}_i + p\mathbf{1}) - \rho\boldsymbol{g} = 0 \tag{6.1.1a}$$

$$\rho_t + \nabla\cdot\rho\boldsymbol{u} = 0 \tag{6.1.1b}$$

$$E_t + \nabla\cdot\left((E+p)\boldsymbol{u} - k_T\nabla T - L(1-\omega)\frac{\rho_i}{\rho}\kappa_\omega\nabla\omega\right) - \eta D\boldsymbol{u}_i : D\boldsymbol{u}_i - \rho\boldsymbol{u}\cdot\boldsymbol{g} = 0 \tag{6.1.1c}$$

on $\Omega\otimes(0,\tau)$, with Dirichlet flow boundary conditions except at the free surface, and all Dirichlet boundary conditions for energy E. These equations represent conservation of momentum, mass, and energy respectively. Note that ρ_t appears in (6.1.1b), but ρ is not an explicit variable and only depends on pressure when the melt fraction is positive, therefore this system is still differential algebraic. The energy equation consists of a transport term, thermal diffusion, moisture diffusion, and heat production due to strain heating. We switch immediately to the steady-state form of (6.1.1) in which $(\rho\boldsymbol{u})_t$, ρ_t, and E_t are all zero. Constitutive relations are needed for total density ρ ($\mathrm{kg\,m^{-1}}$), ice velocity $\boldsymbol{u}_i$ ($\mathrm{m\,s^{-1}}$), temperature T (K), volumetric moisture fraction (porosity) ω (nondimensional), and viscosity η ($\mathrm{Pa\,s} = \mathrm{kg\,m^{-1}\,s^{-1}}$). In general, each constitutive relation is a function of all field variables. The thermal conductivity k_T ($\mathrm{J\,m^{-1}\,K^{-1}\,s^{-1}}$) and hydraulic diffusivity κ_ω ($\mathrm{kg\,m\,s^{-1}}$) are taken to be constant because experimental data are sparse, but this assumption is in no way critical.

The constitutive relations for temperature and moisture fraction are usually defined piecewise. It is preferable for the convergence of Newton methods (c.f. Gropp et al., 2000a; Kelley, 1995) to have a discretization with C^1 continuity and it is simpler for manufactured solutions if the constitutive relation has a global (i.e. not piecewise) definition in terms of analytic functions. To achieve this, we decompose the function $\mathcal{S}(x) = x$ into two globally smooth parts

$$\begin{aligned}\mathcal{S}_\delta^-(x) &= \frac{x}{2} - \frac{x}{2}\operatorname{erf}\frac{x}{\sqrt{2}\delta} - \frac{\delta}{\sqrt{2\pi}}\exp\frac{-x^2}{2\delta^2}\\ \mathcal{S}_\delta^+(x) &= \frac{x}{2} + \frac{x}{2}\operatorname{erf}\frac{x}{\sqrt{2}\delta} + \frac{\delta}{\sqrt{2\pi}}\exp\frac{-x^2}{2\delta^2}\end{aligned} \tag{6.1.2}$$

which satisfy $\mathcal{S}_\delta^-(x) < 0$, $\mathcal{S}_\delta^+(x) > 0$ and $\mathcal{S}_\delta^-(x) + S_\delta^+(x) = x$. These functions arise from integrating the error function with standard deviation δ. Taking $\delta \to 0$ recovers the piecewise linear decomposition $\mathcal{S}_0^-(x) = \min(x,0)$, $\mathcal{S}_0^+(x) = \max(x,0)$. In applications where a single global function is not important, the decomposition $\mathcal{S}^\pm$ could be defined using a spline which would reduce the high computational cost of evaluating erf. This decomposition will be used to separate internal energy into thermal and melt contributions.

For convenience in defining constitutive relations, we introduce specific internal energy e ($\mathrm{J\,kg^{-1}}$) which is related to total and kinetic energy through

$$E = \rho e + \frac{1}{2}(1-\omega)\rho_i|\boldsymbol{u}_i|^2 + \frac{1}{2}\omega\rho_w|\boldsymbol{u}_w|^2$$

which we approximate as

$$E = \rho e + \frac{1}{2\rho}|\rho\boldsymbol{u}|^2. \tag{6.1.3}$$

This approximation may be violated, for example, at moderate porosity when the velocity of the melt fraction is very high compared to the bulk velocity, such as in an actively draining moulin. In such

circumstances, it is likely unavoidable to add additional variables for water momentum and energy. Consistent with exact incompressibility, the internal energy is independent of pressure even though observable quantities like temperature and moisture fraction are dependent on pressure. Removing this assumption would produce acoustic waves and a conservative formulation would require that density be an explicit degree of freedom. When combined with the closure for $\rho(p,e)$, shown below, (6.1.3) requires solving an implicit equation involving the decomposition $\mathcal{S}^{\pm}$. This implicit equation can always be reduced to one dimension and can be solved explicitly for some definitions of $\mathcal{S}^{\pm}$. For some purposes, it is acceptable to simply take $\rho \approx \rho_i$. The convective contributions to momentum balance $\rho \boldsymbol{u} \otimes \boldsymbol{u}$ and kinetic energy $\frac{1}{2\rho}|\rho \boldsymbol{u}|^2$ have vanishing influence in glaciology, but are easy to accommodate so we keep them for completeness. An alternative would be to discretize using ρ instead of p as the independent variable, but near incompressibility and the variation due to moisture content causes the resulting system to be extremely ill-conditioned.

The closures for (6.1.1) are

$$\rho(p,e) = \big(1-\omega(p,e)\big)\rho_i + \omega(p,e)\rho_w \tag{6.1.4a}$$

$$\boldsymbol{u}_i(\boldsymbol{u},p,e) = \boldsymbol{u} + \rho(p,e)^{-1}\kappa_\omega \nabla\omega(p,e) \tag{6.1.4b}$$

$$T(p,e) = T_0 + \frac{e_m(p) + \mathcal{S}_\delta^-\big(e - e_m(p)\big)}{c_i} \tag{6.1.4c}$$

$$\omega(p,e) = \frac{\rho_i \mathcal{S}_\delta^+\big(e-e_m(p)\big)}{\rho_w L - (\rho_w-\rho_i)\mathcal{S}_\delta^+\big(e-e_m(p)\big)} \tag{6.1.4d}$$

$$\eta(\gamma,p,e) = B(p,e)\left(\varepsilon^2 + \frac{\gamma}{\gamma_0}\right)^{\frac{\mathfrak{p}-2}{2}} \tag{6.1.4e}$$

with second invariant $\gamma = \frac{1}{2}D\boldsymbol{u} : D\boldsymbol{u}$ and the additional constitutive relations

$$\begin{aligned}
T_m(p) &= T_3 - \beta_{CC} p \\
e_m(p) &= c_i\big(T_m(p) - T_0\big) \\
T^*(p,e) &= T(p,e) - T_m(p) + T_3 \\
B(p,E) &= B_0 \exp\left(\frac{Q-pV}{\mathfrak{n}RT^*(p,e)} - \frac{Q}{\mathfrak{n}RT_0}\right)\big(1 + B_\omega \omega(p,E)\big)^{-1/\mathfrak{n}}
\end{aligned}$$

with physical constants given in Table 6.1. Since dimensional units are used here unlike in earlier sections, the strain rate regularization ε is now a fraction of γ_0 which is the second invariant of a reference strain rate. The Arrhenius relation is normalized to zero pressure and a reference temperature T_0. This formulation is not conventional in glaciology, but permits more intuitive understanding of parameters because they are no longer sensitive to the power law exponent $\mathfrak{p}$ and large exponential terms.

The definitions of the moisture flux in (6.1.1c) and ice velocity $\boldsymbol{u}_i$ require further explanation. The total momentum can be defined in terms of constituent momenta as

$$\begin{aligned}
\rho\boldsymbol{u} &= (1-\omega)\rho_i \boldsymbol{u}_i + \omega\rho_w \boldsymbol{u}_w \\
&= (1-\omega)\rho_i \boldsymbol{u}_i + \omega\rho_w \boldsymbol{u}_i + \omega\rho_w(\boldsymbol{u}_w - \boldsymbol{u}_i) \\
&= \rho\boldsymbol{u}_i + \omega\rho_w(\boldsymbol{u}_w - \boldsymbol{u}_i).
\end{aligned} \tag{6.1.5}$$

The second term is the momentum of the moisture content in the reference frame of the ice. The mass flux of the moisture is $-\kappa_\omega\nabla\omega$ which is also the momentum density. That is, the integral of $-\kappa_\omega\nabla\omega$ over a surface element is the mass flux ($\mathrm{kg\,s^{-1}}$) through that surface, while the integral over a volume element is the momentum ($\mathrm{kg\,m\,s^{-1}}$) of that volume. Substituting $\omega\rho_w(\boldsymbol{u}_w - \boldsymbol{u}_i) = -\kappa_\omega\nabla\omega$ into (6.1.5) and solving for $\boldsymbol{u}_i$ yields

$$\boldsymbol{u}_i = \boldsymbol{u} + \rho^{-1}\kappa_\omega\nabla\omega$$

Symbol	Value	Description
c_i	$2009\,\mathrm{J\,kg^{-1}\,K^{-1}}$	Specific heat capacity of ice
k_T	$2.1\,\mathrm{W\,m^{-1}\,K^{-1}}$	Thermal conductivity of ice
ρ_i	$910\,\mathrm{kg\,m^{-3}}$	Density of ice
ρ_w	$1000\,\mathrm{kg\,m^{-3}}$	Density of liquid water
L	$3.34\times10^5\,\mathrm{J\,kg^{-1}}$	Latent heat of fusion
g	$9.81\,\mathrm{m\,s^{-2}}$	Gravitational acceleration
κ_ω	$1.045\times10^{-4}\,\mathrm{kg\,m^{-1}\,s^{-1}}$	Hydraulic diffusivity of ice
β_{CC}	$7.9\times10^{-8}\,\mathrm{K\,Pa^{-1}}$	Clausius-Capeyron gradient
T_3	$273.15\,\mathrm{K}$	Triple point of water
T_0	$260\,\mathrm{K}$	Reference temperature
γ_0	$\frac{1}{2}(10^{-10}\,\mathrm{s^{-1}})^2$	Second invariant of reference strain rate
B_0	$8.56\times10^{14}\,\mathrm{Pa\,s}$	Viscosity at reference strain rate and temperature
Q	$6.0\times10^4\,\mathrm{J\,mol^{-1}}$	Activation energy for creep
V	$-13.0\times10^{-6}\,\mathrm{m^3\,mol^{-1}}$	Activation volume for creep
R	$8.31441\,\mathrm{J\,mol^{-1}\,K^{-1}}$	Ideal gas constant
B_ω	181.25	Influence of water content on viscosity (Greve and Blatter, 2009)

Table 6.1: Physical constants used for the viscous heat transport problem. The same constants are used in Aschwanden et al. (2011).

as in (6.1.4b). In (6.1.1c), we need the energy flux in the reference frame of the total velocity. Starting from the mass flux in the reference frame of total velocity,

$$\begin{aligned}\omega\rho_w(\boldsymbol{u}_w-\boldsymbol{u}) &= \omega\rho_w(\boldsymbol{u}_w-\boldsymbol{u}_i)+\omega\rho_w(\boldsymbol{u}_i-\boldsymbol{u})\\ &= -\kappa_\omega\nabla\omega+\frac{\omega\rho_\omega}{\rho}\kappa_\omega\nabla\omega\\ &= -\left(1-\frac{\omega\rho_\omega}{\rho}\right)\kappa_\omega\nabla\omega\\ &= -(1-\omega)\frac{\rho_i}{\rho}\kappa_\omega\nabla\omega\end{aligned} \tag{6.1.6}$$

where the moisture flux and (6.1.4b) was used on the second line. The mass flux in (6.1.6) is converted to energy flux by multiplying by the latent heat L to produce the moisture flux appearing in (6.1.1c). The use of constant hydraulic conductivity κ_ω is a poor approximation for large amplitude ω since intraglacial conduits form for higher melt fractions, thus conductivity becomes nearly infinite. For such cases, it would likely be better to use a Darcy-type constitutive relation accommodating the gravitational contribution and with conductivity dependent on moisture, perhaps of the form $\kappa(\omega)=\kappa_0\exp\frac{\omega}{\omega_0}$ where κ_0 is the conductivity for vanishing moisture fraction and ω_0 is a characteristic melt fraction on the order of 1 %.

The volumetric flux $p\boldsymbol{u}$ appearing in (6.1.1c) could have been written as part of a single heating term. For single-phase compressible flows, the heat production can be written $(\eta D\boldsymbol{u}-p\mathbf{1}):\nabla\boldsymbol{u}$ from classical definitions of work (e.g. Hutter and Jöhnk, 2004), plus a kinetic energy contribution $-\boldsymbol{u}\cdot\nabla p$. These can be manipulated as

$$\begin{aligned}(\eta D\boldsymbol{u}-p\mathbf{1}):\nabla\boldsymbol{u}-\boldsymbol{u}\cdot\nabla p &= \eta D\boldsymbol{u}:\nabla\boldsymbol{u}-p\nabla\cdot\boldsymbol{u}-\boldsymbol{u}\cdot\nabla p\\ &= \eta D\boldsymbol{u}:D\boldsymbol{u}-\nabla\cdot(p\boldsymbol{u})\end{aligned} \tag{6.1.7}$$

which is the form appearing in (6.1.1c). Although both expressions are equivalent for the continuous equations, the discrete equations are different because numerical solutions only satisfy partial differential equations weakly. Only the latter formulation is conservative. This formulation is also more amenable to discretization and is mandatory for stability when density is dependent on pressure (as it is here when melt

content is available, though the underlying physical process still does not support acoustics) and diffusion is poorly resolved on the mesh. The first term in (6.1.7) is non-reversible dissipation while the second can be recovered through volume change and thus does not affect the global entropy. The ice velocity $\boldsymbol{u}_i$ is used in (6.1.1c) because the melt fraction is assumed to be able to move through the ice matrix without viscous dissipation. Perhaps this is a reasonable assumption because the viscosity of ice is 10^{15} times larger than water, but for relatively stagnant ice with high moisture velocity, it is likely to be significant and the equations can be augmented with an additional diffusive term.

The present formulation is similar to Aschwanden et al. (2011) and approaches their model in the limit $\omega \to 0$, but is more conservative. The formulation here uses a 3D momentum balance and conserves mass, momentum, and energy, regardless of large amplitude moisture content and presence of numerical diffusion. We have intentionally neglected to write constitutive relations of the form $e(T, \omega, p)$ because such relationships are superfluous for the purpose of solving the equations. They can be obtained by inverting the constitutive relations presented here and, depending on the experimental setup, may be useful for model validation.

For the purpose of determining cell Peclet numbers (a diagnostic tool and possible input to numerical stabilization methods), it is useful to write the thermal and moisture fluxes using

$$\begin{aligned} \frac{\partial T}{\partial E} &= \frac{\partial T}{\partial e}\frac{\partial e}{\partial E} \approx \frac{1}{c_i}\frac{1}{\rho} && \text{Cold ice} \\ \frac{\partial \omega}{\partial E} &= \frac{\partial \omega}{\partial e}\frac{\partial e}{\partial E} \approx \frac{1}{L}\frac{1}{\rho} && \text{Temperate ice} \end{aligned}$$

so that the diffusive energy flux driven by energy gradient can be written as $-K_T \nabla E - K_\omega \nabla E$ with

$$K_T \approx \frac{k_T}{\rho c_i} = 1.15 \times 10^{-6}\,\mathrm{m^2\,s^{-1}} \tag{6.1.8}$$

$$K_\omega \approx \frac{L\kappa_\omega}{\rho L} = 1.15 \times 10^{-7}\,\mathrm{m^2\,s^{-1}}. \tag{6.1.9}$$

Note that we have neglected kinetic energy and density dependence in this approximation. For an element with streamline length $h = 1\,\mathrm{km}$ and velocity $v = 1\,\mathrm{km\,a^{-1}}$, the cell Peclet number is $\mathrm{Pe}_h = hv/K = 2.8 \times 10^4$ for cold ice and ten times larger for temperate ice. Numerical methods for such systems require upwinding to prevent non-physical oscillations. Godunov's Theorem (1954, see e.g. LeVeque, 2002) states that non-oscillatory linear methods for hyperbolic equations are at most first-order accurate. Higher order accuracy requires a nonlinear method, even if the equation being solved is linear. Robust methods for such systems are based on finite volume methods of total variation diminishing (TVD) and total variation bounded (TVB) type, as well as discontinuous Galerkin methods with limiters when necessary (LeVeque, 2002; Harten, 1983; Boris and Book, 1973; Zalesak, 1979; Harten et al., 1987; Liu et al., 1994; Jiang and Shu, 1996; Shu, 2003; Hesthaven and Warburton, 2008). In comparison, continuous finite element methods are much less robust, and most efforts to stabilize finite element methods for transport-dominated processes have used linear stabilization (Brooks and Hughes, 1982; Hughes et al., 1989, 1998; Matthies et al., 2008). Attempts to use nonlinear stabilization with continuous finite element methods have not been very successful. The best methods, according to the recent comparisons (John and Knobloch, 2007, 2008; John and Schmeyer, 2008), involve extreme restrictions on element types and suffer from difficulty in converging the nonlinear systems (Mizukami and Hughes, 1985) or an algebraic construction (Kuzmin et al., 2004) that is difficult to apply with mesh anisotropy and material nonlinearity. For simplicity, we adopt the streamline upwind Petrov-Galerkin method of (Brooks and Hughes, 1982), but a discontinuous Galerkin method would be a better choice for discretization of the energy equation on unstructured grids.

An additional detail appears in the finite element discretization of (6.1.1). The symmetric gradient of ice velocity $D\boldsymbol{u}_i$ is needed to define the stress, but only the gradient of momentum

$$\begin{aligned} \nabla(\rho \boldsymbol{u}) &= \boldsymbol{u} \otimes \nabla\rho + \rho \nabla \boldsymbol{u} \\ &= \boldsymbol{u} \otimes \nabla\rho + \rho \nabla(\boldsymbol{u}_i - \rho^{-1}\kappa_\omega \nabla\omega) \end{aligned} \tag{6.1.10}$$

is avialable. Density is not an explict variable in this formulation and the definition of density involves the gradient of the explicit variables, therefore solving for $\nabla \boldsymbol{u}_i$ in (6.1.10) produces a second order term. This term can be evaluated using the second derivative of the basis functions or using a local projection, but neither of these methods are conveniently available in Dohp at present, therefore we approximate $\nabla \boldsymbol{u}_i$ using

$$\nabla(\rho \boldsymbol{u}) \approx \boldsymbol{u} \otimes \nabla \rho + \rho \nabla \boldsymbol{u}_i$$

of (6.1.10).

Although this simulation is run using realistic geometry on a section of the Jakobshavn Isbræ channel, not all boundary conditions necessary for a realistic simulation have been implemented in Dohp. More sophisticated lateral boundary conditions, energy conditions, and free-surface evolution are not considered in the present model. See Aschwanden et al. (2011) for further discussion of boundary conditions for energy transport. The output of this simulation is *not* intended as a predictive model, instead it is a demonstration of the capability of the methods to handle systems that were previously not possible or too computationally expensive due to the need for short time steps.

6.1.2 Numerical solution

As discussed above and in Section 5.1, the equation system (6.1.1) must be interpreted weakly because the fields have insufficient regularity. Dropping the transient terms, the weak form is: find $(\rho \boldsymbol{u}, p, E) \in W_D^{1,\mathfrak{p}} \times L^2 \times H_D^1$ such that

$$\int_\Omega \Big[\nabla \hat{\boldsymbol{m}} : (-\rho \boldsymbol{u} \otimes \boldsymbol{u} + \eta D\boldsymbol{u}_i) - p \nabla \cdot \hat{\boldsymbol{m}} - \rho \hat{\boldsymbol{m}} \cdot \boldsymbol{g} - \hat{p} \nabla \cdot \rho \boldsymbol{u} \\ + \nabla \hat{E} \cdot \Big(-(E+p)\boldsymbol{u} + k_T \nabla T + L(1-\omega)\frac{\rho_i}{\rho} \kappa_\omega \nabla \omega \Big) - \hat{E} \left(\eta D\boldsymbol{u}_i : D\boldsymbol{u}_i - \rho \boldsymbol{u} \cdot \boldsymbol{g} \right) \Big] = 0 \quad (6.1.11)$$

for all momentum, mass, and energy test functions $(\hat{\boldsymbol{m}}, \hat{p}, \hat{E}) \in W_0^{1,\mathfrak{p}} \times L^2 \times H_0^1$. The trial spaces $W_D^{1,\mathfrak{p}}$ and $\times H_D^1$ have inhomogeneous Dirichlet boundary conditions built in on $\partial\Omega \setminus \Gamma_s$ and $\partial\Omega$ respectively, where Γ_s is the free surface. The corresponding test spaces have homogeneous boundary conditions built in. We assume that the free surface $\Gamma_s \subset \partial\Omega$ is non-empty, therefore the pressure trial space and corresponding mass test space do not need a constraint.

The finite element method solves (6.1.11) by introducing discrete test and trial spaces as well as a method of numerical integration. The numerical examples in this section use $Q_3 - Q_2 - Q_3$, but other choices are possible. In the present implementation, the order of each approximation space can be chosen independently through run-time options. Many of the more complicated spaces discussed in Section 5.1.2 are not yet implemented in Dohp, but would automatically become available through run-time options once the library support is added. Discontinuous P_k^{disc} spaces are not used here because the visualization support is not complete.

Evaluation of the discrete equations and components needed by the solver are implemented using the methods discussed in Section 4.3.1. The discrete residual is evaluated monolithically, by evaluating all fields on a single set of quadrature points and computing the coefficients of the test functions at those points. For manufactured solutions, the artificial source terms are also evaluated at the quadrature points. During residual evaluation, certain intermediate quantities including the velocity $\boldsymbol{u}$, temperature T, and melt fraction ω, as well as the derivatives of these quantities with respect to the independent variables $\rho \boldsymbol{u}$, p, E and their gradients, are stashed away in quadrature-local storage managed by Dohp. These stashed values are used to apply the Jacobian and blocks of the Jacobian matrix-free. The memory and asmyptotic flops benefits of this approach were discussed in Section 4.3. Computation of the gradients was done by hand, but is an error-prone process that should be automated using reverse-mode automatic differentiation in a highly local manner (a single quadrature point at a time). An alternative would be to simply store the independent variables and their gradients, then evaluate the action of the Jacobian using forward-mode

(automatic or by-hand) differentiation. The disadvantage of this is that constitutive relations would have to be re-evaluated during Jacobian application. Since constitutive relations be quite expensive, it is preferable to store some intermediate values. Another alternative is to store the full Jacobian at each quadrature point. This would be a nearly dense matrix of size 17×20 (20 comes from the state and gradients of all five fields, three rows are dropped because the gradient of the pressure/mass test function does not appear), or slightly smaller if momentum convection and other symmetry-breaking terms were dropped. This matrix would be more effort to compute, would require more storage, and would involve more floating point operations to apply than the present method. On today's hardware, the hybrid method of storing coefficients of specific intermediate quantities is clearly preferable to these alternatives. However, on some vector hardware, or for other problems with less compact representations, this explicit form could still be desirable.

The Jacobian resulting from Newton linearization of (6.1.11) has the block structure

$$J = \begin{pmatrix} J_{uu} & J_{up} & J_{uE} \\ J_{pu} & 0 & 0 \\ J_{Eu} & J_{Ep} & J_{EE} \end{pmatrix}. \tag{6.1.12}$$

The contents of these blocks is summarized below.

J_{uu} The viscous and (much smaller for ice) momentum convection terms. This block is nearly symmetric positive definite and has variable coefficients and anisotropy created by differentiating the power-law constitutive relation.

J_{up} The weak pressure gradient, viscosity dependence on pressure (directly and through density, temperature, and melt fraction), and the gravitational contribution from pressure-induced density variation (from changing the pressure melting temperature which may change the melt fraction). This block is large in magnitude, mostly due to the weak pressure gradient which is nearly balanced by the gravitational forcing source term.

J_{uE} The viscous dependence on energy (via temperature, melt fraction, and density) as well as the gravitational contribution due to energy-induced density variation. This term comes from a highly nonlinear term (the Arrhenius and moisture-dependent constitutive relations) but is typically not especially large in glaciology. For bouyancy-driven flows such as mantle convection, the density variation is the crucial driving force for circulation. However, despite the crucial coupling provided by this block and it's significant nonlinear influence, it contributes quite little to linear stiffness.

J_{pu} The divergence of momentum density which enforces mass conservation. It is nearly equal to J_{up}^T and is thus also large in magnitude. Being the only non-zero block in its row, this is of critical importance.

J_{Eu} The sensitivity of energy on momentum, which is mostly the advective transport (the gradient of energy divided by density). This term is very large in boundary layers containing large thermal and moisture gradients. In particular, due to the high Peclet numbers involved, the boundary layers will often never be fully resolved at a practical resolution, thus mesh refinement is capable of resolving ever-larger gradients. This results in the size of this term being essentially mesh-dependent. This block also contains a similar term involving the pressure gradient, but pressure does not contain the same boundary layers as energy (indeed, the pressure gradient is approximately equal to $\rho \boldsymbol{g}$), so its contribution is much smaller and more benign (provided the pressure discretization is stable so that there are no oscillations).

J_{Ep} The contribution to thermal and moisture diffusion as well as the advective contribution $\boldsymbol{u} \cdot \nabla$. If the transition width δ in (6.1.2) is made small, small changes in pressure can move the pressure melting which locally switches from thermal to moisture diffusion. According to (6.1.8), this is an order of magnitude change in coefficients, but could be different depending on the constitutive relation for moisture diffusion.

J_{EE} An advection-diffusion system for energy. It is generally advection-dominated except in boundary layers and regions of stagnant ice where diffusion driven by temperature and moisture gradient become significant. It is typical that the velocity is almost parallel to the bed in which case vertical diffusion remains significant compared to vertical advection while horizontal (nearly streamline) diffusion is negligible. In such cases, the thermal gradients in the horizontal direction are very small. There are also advection-like terms arising from differentiating through $S_\delta^\pm$ which appear as transport in the directions ∇T and $\nabla\omega$ (usually nearly vertical).

All blocks in this system are available in unassembled form, using the reduced storage at quadrature points discussed above.

The discussion above suggests that the blocks J_{uu}, J_{up}, J_{pu}, J_{Eu}, and J_{EE} account for most of the linear stiffness, therefore the multiplicative preconditioner

$$P = \begin{bmatrix} \begin{pmatrix} J_{uu} & J_{up} \\ J_{pu} & 0 \end{pmatrix} & \\ \begin{pmatrix} J_{Eu} & J_{Ep} \end{pmatrix} & J_{EE} \end{bmatrix} \tag{6.1.13}$$

should be an effective preconditioner for (6.1.12). Indeed, if applied exactly,

$$P^{-1}J = \begin{bmatrix} 1 & \begin{pmatrix} J_{uu} & J_{up} \\ J_{pu} & \end{pmatrix}^{-1} \begin{pmatrix} J_{uE} \\ 0 \end{pmatrix} \\ 0 & 1 \end{bmatrix}$$

which satisfies $(P^{-1}J-1)^2 = 0$ ensuring that left-preconditioned GMRES converges in two iterations. Computing

$$\begin{aligned} (JP^{-1}-1)^2 &= JP^{-1}JP^{-1} - 2JP^{-1} + 1 \\ &= P(P^{-1}JP^{-1}J - 2P^{-1}J + 1)P^{-1} \\ &= P(P^{-1}J-1)^2P^{-1} \\ &= (P^{-1}J-1)^2 = 0 \end{aligned}$$

ensures that right-preconditioned GMRES also converges in two iterations. Since GMRES convergence is more reliable for matrices that are not too far from normal (Nachtigal et al., 1992; Embree, 1999; Trefethen and Embree, 2005) and because we intend to apply P only approximately, it is beneficial that the off-diagonal part of $P^{-1}J$ is relatively small. This preconditioner can be applied with one (anisotropic variable-coefficient) Stokes solve and one advection-diffusion (variable-coefficient, with anisotropy due to stabilization) solve.

Due to the mechanics of GMRES, three preconditioner applications are required for two iterations. By storing more information, the flexible variant FGMRES (Saad, 1993) can extract the solution after two right-preconditioned iterations without a third preconditioner application. The generalized conjugate residual method GCR (Eisenstat et al., 1983) has the same property and has equivalent convergence properties to GMRES when the preconditioner is linear Saad and Schultz (1986). Both of these methods store two vectors per Krylov iteration which enables them to be tolerant of variable preconditioners. They both work with the right-preconditioned form, have convergence tests in terms of unpreconditioned residuals, and are commonly restarted to bound the total storage requirement. FGMRES defines the residual and its norm through a recurrence relation similar to GMRES which has the advantage of requiring less arithmetic per iteration, but means that the residuals are expensive to compute during the iteration, therefore the norm estimated by the algorithm may be unstable. Classical Gram-Schmidt orthogonalization is not stable, but it is much faster than modified Gram-Schmidt, especially in parallel where reductions are very expensive. Since GCR computes the norm of the true residual explicitly instead of through a recurrence relation, the CGR convergence test is independent of inaccuracy in the classical Gram-Schmidt process. GCR with an inexact iterative preconditioner was introduced in Van der Vorst and Vuik (1994)

under the name GMRESR and a variant using the same amount of arithmetic per iteration as FGMRES was presented in Vuik (1995). See also Brakkee et al. (1998) for practical comparisons with domain decomposition methods for incompressible flow.

For the Stokes solve in the application of (6.1.13), we start with the factorization

$$\begin{pmatrix} J_{uu} & J_{up} \\ J_{pu} & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ J_{pu}J_{uu}^{-1} & 1 \end{pmatrix} \begin{pmatrix} J_{uu} & J_{up} \\ 0 & S \end{pmatrix}$$

where $S = -J_{pu}J_{uu}^{-1}J_{uu}$. Since S is dense and will be solved with approximately, it is reasonable to drop the lower-triangular block, leaving

$$P_s = \begin{pmatrix} J_{uu} & J_{up} \\ 0 & S \end{pmatrix} \tag{6.1.14}$$

since the preconditioned operator also has a minimal polynomial of degree 2.

Three assembled matrices are used in the solver defined using approximate solves with (6.1.13) and (6.1.14). The first is B_{uu} which is an approximation of the momentum block J_{uu} assembled using the same physics with truncated basis functions and sub-element 2^3-point Gauss quadrature. This matrix corresponds to a Q_1 discretization on the sub-elements. We approximate the inverse of J_{uu} by 10 conjugate gradient iterations preconditioned by block incomplete Cholesky applied to B_{uu}.

An approximation of the energy coupling B_{EE} is assembled using the same methodology and the inverse of J_{EE} is computed using GMRES preconditioned by incomplete LU. This system is converged to a reasonable tolerance of 10^{-5} because it is relatively inexpensive compared to the Stokes solve.

Approximating the inverse of S and the greater Stokes system is more involved. Following the results of (Olshanskii and Reusken, 2006) extrapolated to viscosity variation that is not isotropic or piecewise constant, we approximate S by assembling B_{pp}, the mass matrix weighted by the inverse of effective viscosity η. The quadrature for this operator in the pressure space is defined using local 2^3-point Gauss quadratures on the sub-elements associated with the Q_3 basis functions. While this quadrature has a lower order of accuracy than a Gauss quadrature with similar number of points on the whole element, it is more local and more robust to sharp viscosity variation. It also has the advantage of reusing the same constitutive relation evaluations as the assembled momentum and energy operators. The inverse of S is approximated using incomplete Cholesky applied to the the scaled mass matrix B_{pp} which has been significantly more robust than the lumped variant in common use (e.g. Burstedde et al., 2008; May and Moresi, 2008).

We converge the Stokes solve to a relative tolerance of 10^{-3} using GCR with the upper-triangular preconditioner (6.1.14) where S is replaced by B_{pp} and J_{uu} solved inexactly. The Eisenstat-Walker Eisenstat and Walker (1996) method is used to adjust the linear solve tolerance of the outer Krylov iteration as the Newton iteration converges. Since the inner solves are relatively accurate and the J_{uE} block has little linear stiffness, this usually converges in one iteration.

A natural alternative preconditioner which was proposed by Elman et al. (2011) for Picard linearization of 2D isoviscous bouyancy-driven flows with the Boussinesq approximation using $Q_2 - Q_1$ elements avoids nested iteration by using

$$P_1 = \begin{pmatrix} J_{uu} & J_{up} & J_{uE} \\ 0 & B_{pp} & 0 \\ 0 & 0 & J_{EE} \end{pmatrix}.$$

Despite several attempts, this preconditioner and many variants were not found to deliver the robustness desired for the present model. Among other deficiencies, it tended to have very poor behavior under GMRES restarts and often triggered instability in classical Gram-Schmidt. Since full orthogonalization using modified Gram-Schmidt is not practical, this approach is not used in the present numerical study. However, through the compositional algebra implemented in PETSc, especially the `PCFieldSplit` component, all such variants remain available as run-time options.

If a scalable preconditioner such as multigrid is available for B_{uu} and B_{pp}, and if the results of Olshanskii and Reusken (2006) carry over to the present setting, then the methods used here will also be scalable. Unfortunately, B_{pp} does not capture the anisotropy appearing due to Newton linearization, therefore we cannot expect it to be spectrally equivalent to S. An alternative is the least-squares commutator of Elman et al. (2006); Elman (1999) discussed in Section 4.1.1. Experiments with the LSC preconditioner (of which there are many variants) have not shown a clear advantage for this problem. Since the present tests have a sticky bed, are not especially high resolution, and algebraic multigrid tends not to be robust for vector problems with anisotropy and variable coefficients or for advection-dominated problems, we are using incomplete factorization preconditioners for B_{uu}, B_{pp}, and B_{EE}. Multigrid is likely necessary for other problems, especially those having large regions with a slippery bed, and I believe that pervasive support for various forms of geometric multigrid is important.

6.1.3 Verification

To verify the correctness of the implementation, we consider a manufactured solution with rich structure and choose a parameter range to activate all the terms in (6.1.1) and constitutive relations. Instead of the physical parameters in Table 6.1, we take all parameters to be of order one, with solid and melt densities of 1 and 2 respectively. For this problem, the melt fraction rises as high as 28 % and temperature ranges 11 % of its absolute value. Since the activation volume V and Clausius-Capeyron gradient β_{CC} are relatively large, the pressure plays a significant, and occasionally dominant role in defining the temperature and the material rheology. The large moisture content and large density contrast also increase the strength of the nonlinearity. The chosen solution is given by

$$\begin{aligned}
\rho u &= \frac{1}{3}\sin\frac{\pi x}{2}\cdot\sin\frac{\pi y}{2}\cdot\sin\frac{\pi z}{2} \\
\rho v &= -\frac{1}{3}\cos\frac{\pi x}{2}\cdot\cos\frac{\pi y}{2}\cdot\sin\frac{\pi z}{2} \\
\rho w &= -\frac{2}{3}\cos\frac{\pi x}{2}\cdot\sin\frac{\pi y}{2}\cdot\cos\frac{\pi z}{2} \\
p &= 1+\cos\frac{\pi x}{2}\cdot\sin\frac{\pi y}{2}\cdot\sin\frac{\pi z}{2} \\
E &= \sin\pi\frac{x+y+z^2}{2}\cdot\cos\pi\frac{x^2+y+z}{2}.
\end{aligned} \tag{6.1.15}$$

Due to the non-uniform flow and various constitutive nonlinearities, nondimensional numbers are spatially variable. The Reynolds number ranges up to about 2.4, the Peclet number ranges up to 5.3, and the Prandtl number ranges from 0.6 to 1. A computed solution, accurate to 0.5 % is shown in Figure 6.1. As usual, this manufactured solution is not physically realizable, but it excercises all the terms.

We consider norms for a $Q_3 - Q_2 - Q_3$ approximation under h-refinement. Due to the direct appearance of pressure in the equations, we cannot expect to realize fourth order convergence, at least not for the energy equation. Figure 6.2 shows the observed convergence behavior in which energy clearly converges with only third order accuracy. I do not have an explanation for why the convergence for the energy equation eventually stagnates. It could be an artifact of the manufactured solution process (perhaps rectifiable using more accurate quadrature for the forcing term), other quadrature errors due to nonlinearity, or stability of the continuum equations or discretization.

The nonlinear solver converges quadratically as seen in Table 6.2. All subsequent examples have exhibited similar convergence behavior.

Numerical experiments with nonlinear solver convergence and pseudo-transient continuation (Coffey et al., 2003; Kelley and Keyes, 1998) indicates that this system does not always have steady-state solutions, and when steady-state solutions exist, they may be non-unique. It would be interesting to explore these uniqueness properties using bifurcation techniques such as those in Allgower and Georg (2003). However, these phenomena have not been observed in parameter ranges that are realistic for ice flow, so we do not

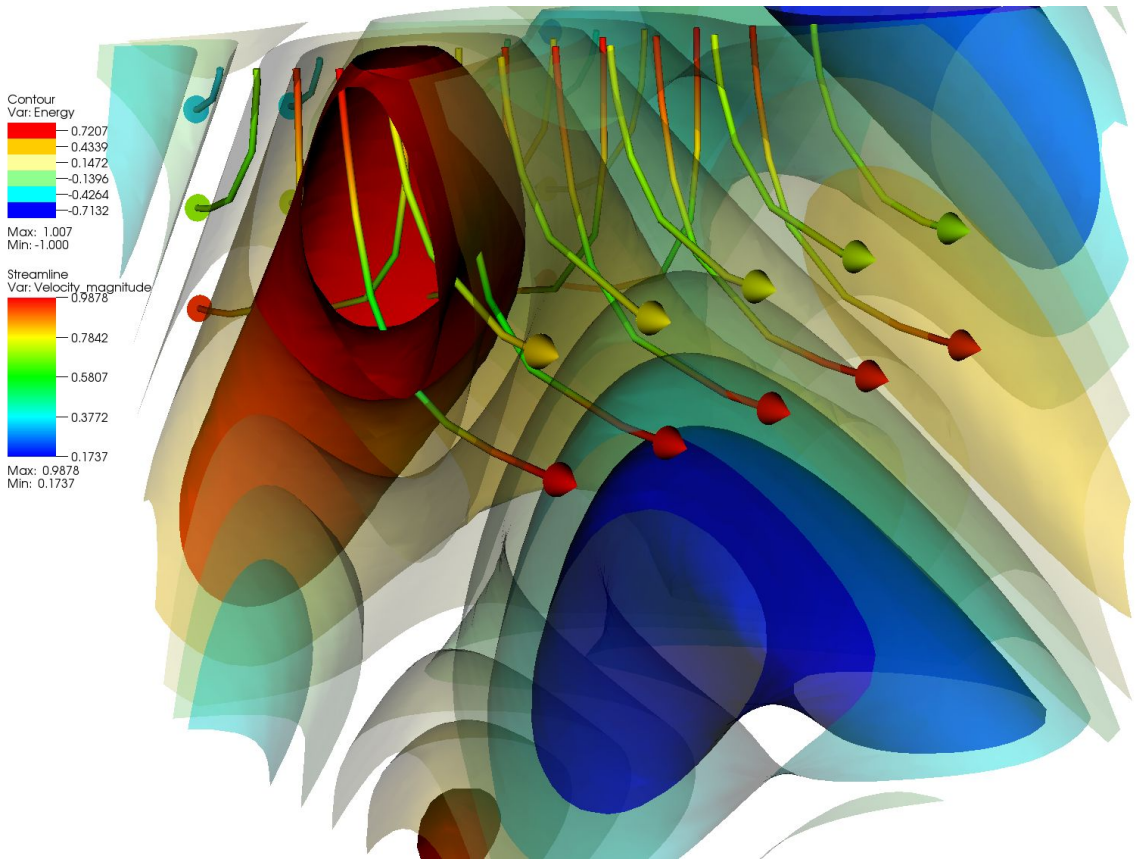

Figure 6.1: The numerical approximation to the manufactured solution (6.1.15) as computed using a $Q_3 - Q_2 - Q_3$ finite element discretization on a $12 \times 12 \times 12$ mesh. The converging streamlines are symmetric from below. Energy isosurfaces are shown, with the phase transition occuring at approximately $E = 0$. The numerical solution is accurate to 4 digits for momentum and energy and 2 digits for pressure, evaluated using the maximum norm and a higher order quadrature rule. The gradients are accurate to 3 digits for momentum and energy, with 3 % error in the pressure gradient.

Iteration	Mass	Momentum	Energy	Total
0	2.142762e-01	1.431024e+01	2.742861e+01	3.093796e+01
1	1.086178e-05	8.386431e+00	8.412471e+00	1.187863e+01
2	2.928430e-06	4.103421e+00	3.579857e+00	5.445497e+00
3	1.744093e-06	3.059956e+00	6.853340e-01	3.135764e+00
4	8.688964e-07	1.891518e+00	2.980380e-01	1.914854e+00
5	4.952597e-07	6.852763e-01	7.214378e-02	6.890634e-01
6	1.430063e-07	4.827890e-02	6.381075e-03	4.869877e-02
7	1.057706e-08	3.257086e-05	3.007905e-05	4.433521e-05
8	1.759267e-11	4.249319e-10	1.387815e-10	4.473666e-10

Table 6.2: Nonlinear convergence rates.

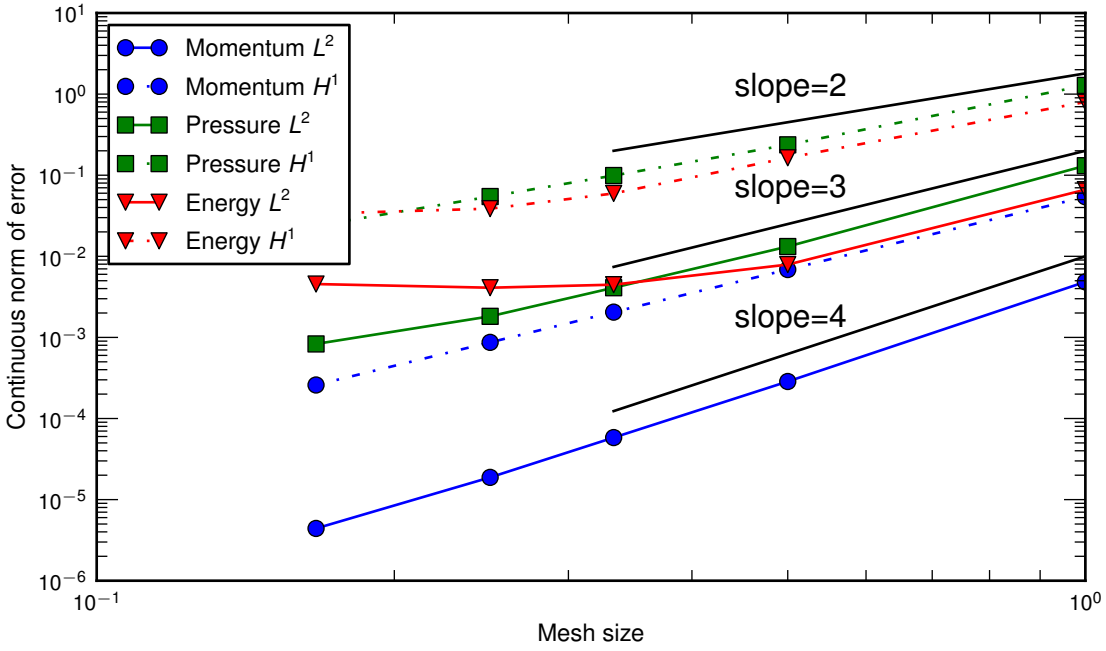

Figure 6.2: Convergence rates for $Q_3 - Q_2 - Q_3$ under h-refinement.

explore them further. Note that such nonlinear effects do occur in glaciology, but generally involve sliding and/or geometry change. With thermally-induced density variation, gravity, and a boundary heat source, there are not steady-states above a critical Rayleigh number. This is seen in mantle convection and other fields, for which (6.1.1) is also valid, therefore it is no surprise that steady-states are not present.

For Dirichlet problems, pressure is only determined up to a constant. As far as the solver is concerned, this is easy to handle by applying the Krylov iteration with null space removed and using a preconditioner that is tolerant of the one dimensional null space. Most preconditioners other than direct solvers are suitable, and the methods of Section 4.1.1 perform fine as long as inner solvers (if used) are also informed of the null space. However, unlike for linear and power-law Stokes problems, the absolute value of pressure affects the other variables. Thus there is a one-parameter family of solutions in which all variables change rather than only the pressure. Additionally, it is possible to compute a negative pressure during the nonlinear solve. This is most common at higher viscosity because the forcing terms in the manufactured solutions become huge, thus producing extreme values of pressure. Negative pressures produce feedback through the activation volume V to generate exponentially large viscosity.

6.1.4 A simple problem

To help understand the equations, we consider a block of size $[0,1]^3$ resting on a plate inclined at 30°. The block flows under its own weight with density variation due to moisture content. No-slip boundary conditions are imposed at the bottom, the sides and surface are free. This problem is nondimensional with densities of 1 and 2 for "ice" and "water" respectively. The conductivities are $\kappa_\omega = 2 \times 10^{-2}$ and $k_T = 4 \times 10^{-2}$ which produces a Peclet number of 120. Streamline stabilization similar to SUPG (Brooks and Hughes, 1982) was used (SUPG does not apply directly to this sort of problem). The power law exponent is $\mathfrak{p} = 1.5$ with reference viscosity $B_0 = 5$, leading to a Reynolds number of 0.24. Dirichlet energy boundary conditions $E(x,y,z) = -x(1-y^2)$ are used, where energy is measured relative to $T_0 = T_3 = 10$ at zero pressure. The latent heat of fusion is $L = 10$ which produces a maximum moisture fraction of 15 %.

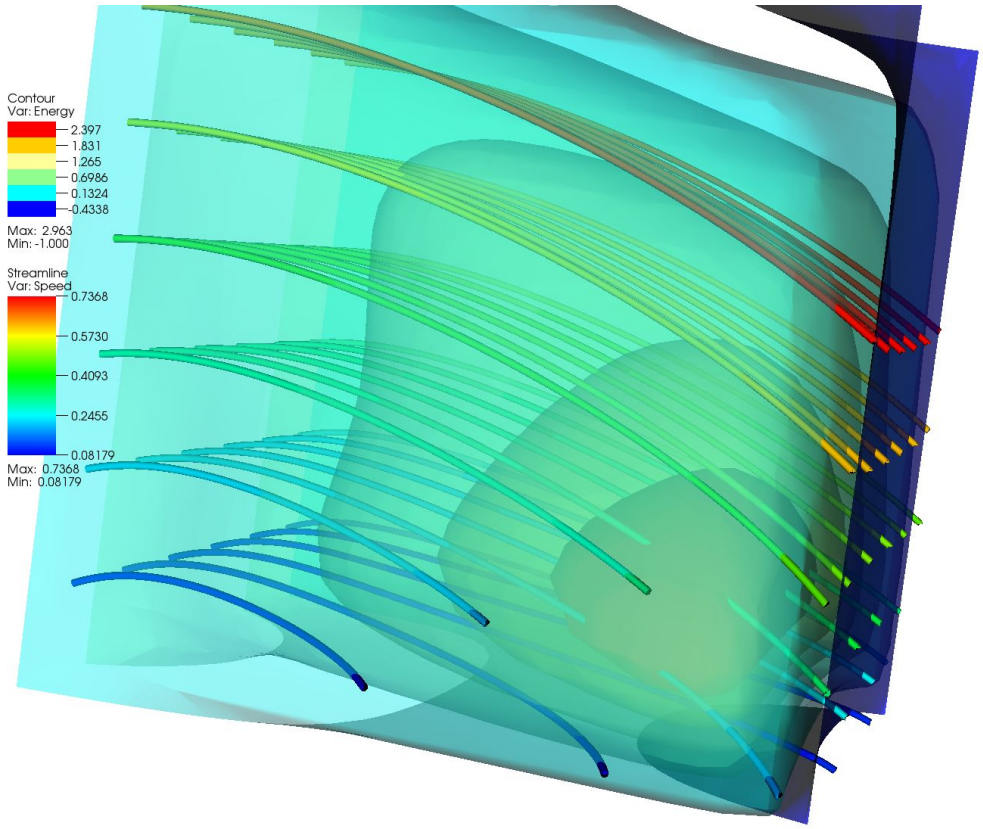

Figure 6.3: Energy isosurfaces and velocity streamlines for the block on an inclined plate. The energy scale has been shifted so that the phase transition occurs at approximately $E = 0$. The warmest region (inside the red isosurface) occurs where the singular viscous heat production (see Figure 6.4) balances advection and combined thermal and moisture diffusion.

The computed energy and flow fields are shown in Figure 6.3 with the corresponding viscous dissipation $\eta D\boldsymbol{u} : D\boldsymbol{u}$ in Figure 6.4. Note that these steady states are not realizable because this configuration does not have steady solutions.

Streamline diffusion such as SUPG is not a robust stabilization for this problem and it breaks down for this problem when the Peclet number exceeds about 200. This seems to be due to the localized shear region near the downstream corner with high viscous dissipation. The velocity in this region is very small, so the streamline diffusion proportional to $|\boldsymbol{u}|\,h$ [1] is small, but there are still sharp gradients. There is a $1/r$ singularity in the viscous heat production term $\eta D\boldsymbol{u}_i : D\boldsymbol{u}_i$, as discussed in Section 5.1.1. The precise mechanism for numerical instability with low diffusivity at the corner that SUPG appears unable to stabilize has not been identified, but it is less pronounced when the angle of incline is reduced. It is conceivable that a steady state solution to the continuum problem no longer exists.

[1] More precisely, the streamline diffusion is proportional to velocity times the length of the cell in the streamline direction, which can be computed using the transform from the reference coordinates as $|\nabla_X \boldsymbol{x} \cdot \boldsymbol{u}|$.

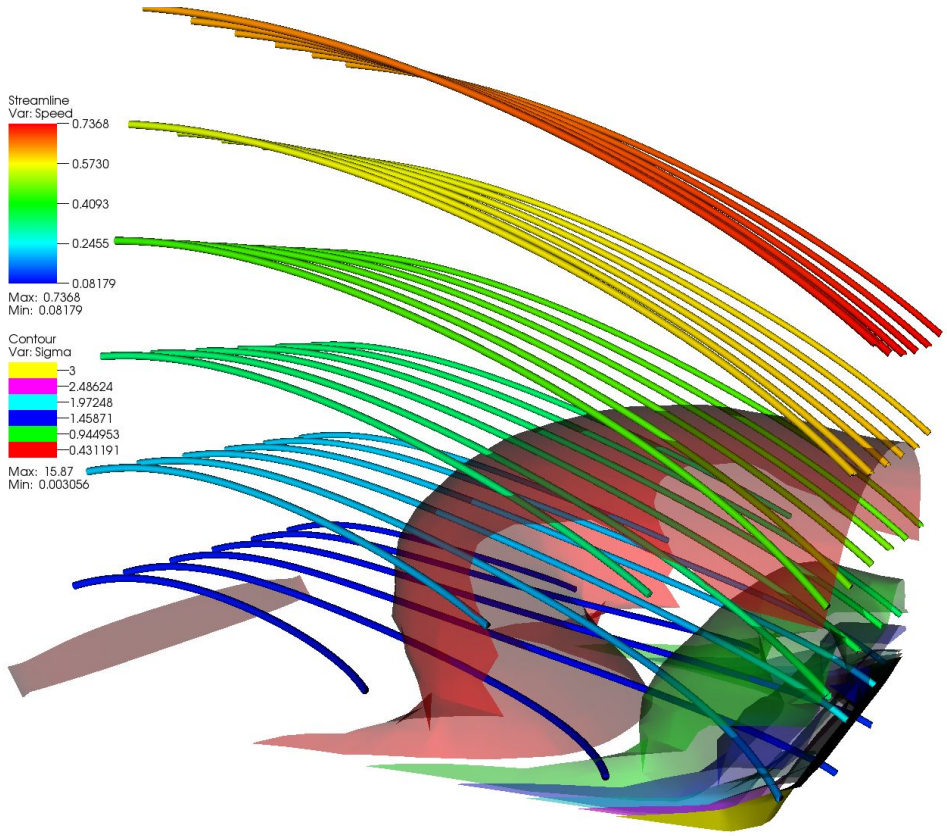

Figure 6.4: Isosurfaces of the viscous heat production rate $\eta D\boldsymbol{u}_i : D\boldsymbol{u}_i$ and velocity streamlines for the block on an inclined plate. Viscous heat production has a $1/r$ singularity at both the upstream and downstream corners, clearly stronger at the downstream corner.

6.2 Jakobshavn Isbræ

6.2.1 Scaling

For the purpose of improving the conditioning for numerical stability and iterative solvers, it is desirable that all field variables assume values of similar size and that the residuals also have similar size. Since we can choose the overall norm of the system arbitrarily, we choose to make field variables and residuals both of order 1. For the field variables, we choose characteristic sizes for the independent variables $\rho\boldsymbol{u}$, p, and E. These are just meant to identify the order of magnitude so we choose powers of ten for simplicity. We would like the following characteristic quantities to have nondimensional size 1.

$$\begin{aligned}[\rho\boldsymbol{u}] &= 10^3\,\mathrm{kg\,m^{-3}}\,10^{-4}\,\mathrm{m\,s^{-1}} = 10^{-1}\,\mathrm{kg\,m^{-2}\,s^{-1}}\\ [p] &= \rho\,|\boldsymbol{g}|\;10^3\,\mathrm{m} = 10^7\,\mathrm{Pa} = 10^7\,\mathrm{kg\,m^{-1}\,s^{-2}}\\ [E] &= \rho c_i\,5\,\mathrm{K} = 10^7\,\mathrm{J\,m^{-3}} = 10^7\,\mathrm{kg\,m^{-1}\,s^{-2}}\end{aligned} \tag{6.2.1}$$

This is degenerate because energy density and pressure have the same units, therefore we can ask for one further constraint. We choose to make a viscous stress of 100 kPa integrated over an area of 1 km^2 have nondimensional size 1 which ensures that the viscous part of the momentum residual will be of order 1. Asking for momentum, pressure, and viscous flux over a characteristic area to be of order unity amounts to solving

$$\begin{aligned}0.1\,\mathrm{kg\,m^{-2}\,s^{-1}} &= 1\\ 10^7\,\mathrm{kg\,m^{-1}\,s^{-2}} &= 1\\ 10^{11}\,\mathrm{kg\,m\,s^{-2}} &= 1\end{aligned}$$

which has the unique solution

$$\mathrm{kg} = 10^3 \qquad \mathrm{m} = 10^{-2} \qquad \mathrm{s} = 10^6. \tag{6.2.2}$$

In other words, the characteristic scales are 1 g, 100 m, and 1 µs. Due to the Lagrange-multiplier coupling of the pressure, this also ensures that the pressure is well-behaved relative to the velocity. Independently changing the scaling for the mass continuity equation would cause the Stokes problem to be non-symmetric. The energy equation, however, is very poorly scaled by this choice. For example, the advective energy flux $[E][u]$ integrated over an are of size 1 km^2 has size

$$(10^6\,\mathrm{m^2})(10^7\,\mathrm{kg\,m^{-1}\,s^{-2}})(10^{-4}\,\mathrm{m\,s^{-1}}) = 10^9\,\mathrm{kg\,m^2\,s^{-3}} = 10^{-10}.$$

We scale the energy equation by 10^{10} to correct this. With nondimensionalization defined by (6.2.2), all solution variables have order unity and the unconverged residuals (after balancing the hydrostatic mode which is about 100 times larger) also have the same order. As a consequence, the Jacobian has norm of order unity in the subspace where the hydrostatic mode is balanced.

6.2.2 Meshing

For the surface, we use the digital elevation model compiled by Motyka et al. (2010) using aerial photography taken 24 July 1985 and ground control points resurveyed during our field campaigns in summer 2007 and 2008. The bedrock location was provided by CReSIS Plummer et al. (2011). A smooth geometric model based on an unstructured triangular mesh was constructed from these rasters using MOAB (Tautges et al., 2004) and other tools being developed as part of the MeshKit project (Tautges, 2011). This unstructured triangular mesh was decimated to reduce size while preserving features. The boundary of the region of interest was described with a polygon in the map plane. Figure 6.5 shows the decimated and meshed regions.

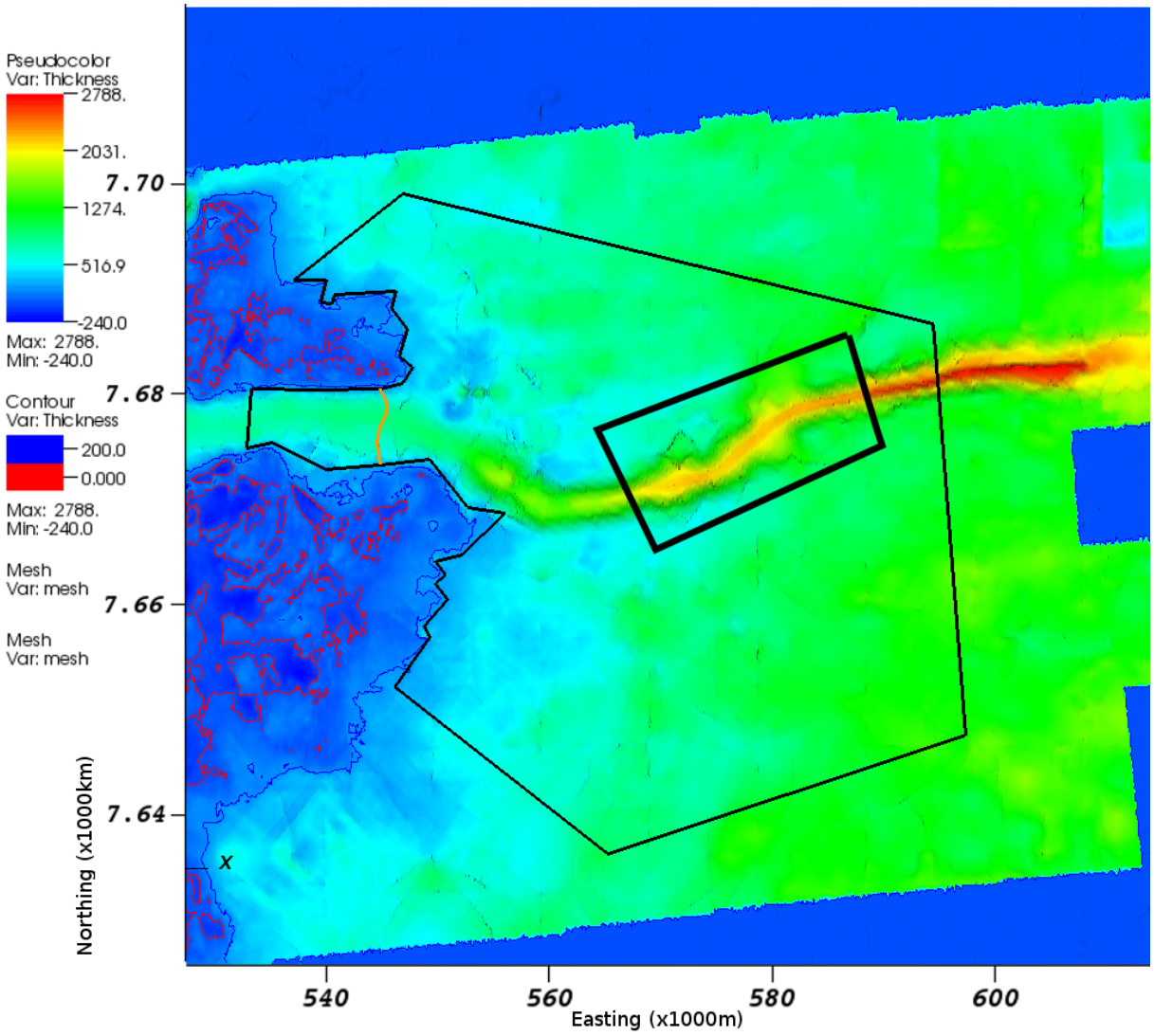

Figure 6.5: Regions of interest for Jakobshavn Isbræ. The outer black line marks the area on which bed and surface measurements were accurate enough to perform decimation. The inner (thick) black line marks the meshed region.

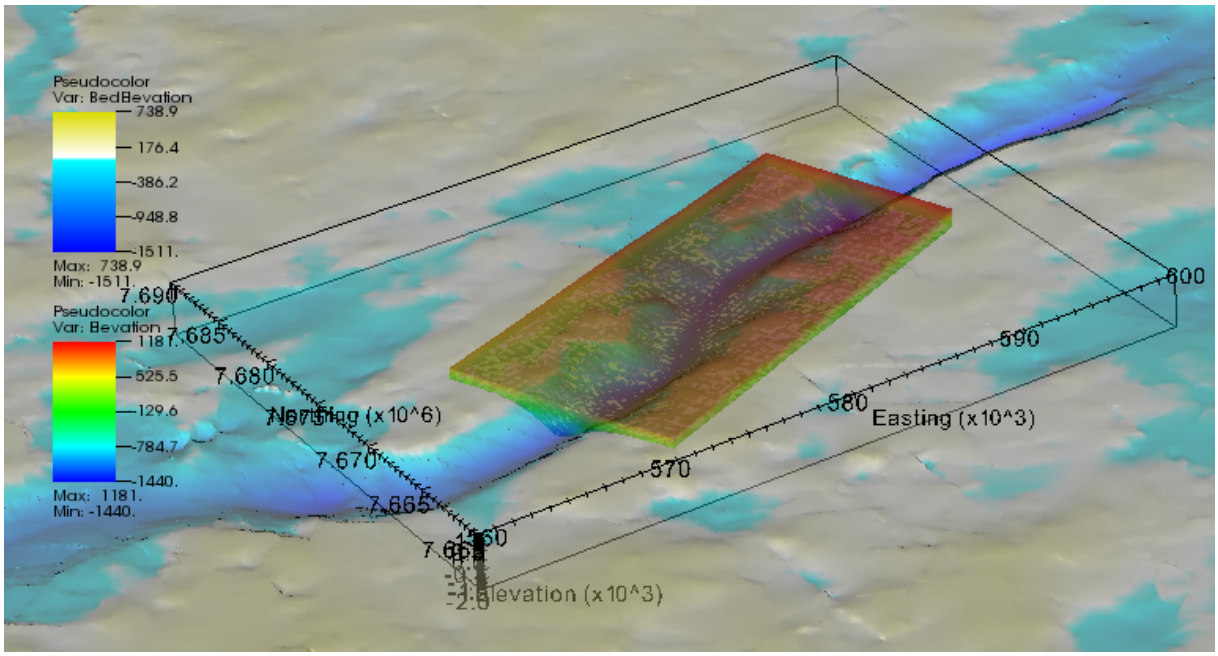

Figure 6.6: The volume to be meshed resting on the geometric model for the bed.

The bed, surface, and lateral cut defines three geometric surfaces and a volume which is shown in Figure 6.6 resting on the decimated geometric model for the bed. Mesh generation involves two phases, unstructured quadrilateral meshing in the horizontal followed by graded sweeping in the vertical. Bed slope was used as a refinement indicator for the horizontal, the quadrilateral surface mesh was generated using the CUBIT Adaptive Meshing Algorithm Library (Blacker et al., 1994). This mesh was swept to create a hexahedral mesh, but the quadrilateral elements on the surfaces were still stored, along with association to the geometric model. In the future, this geometric model will be used to improve geometric fidelity and define high-order slip conditions that allow the mesh to move along the geometry. Additionally, we intend to further improve mesh quality, especially in the vicinity of steep bed and surface slopes, by solving elliptic smoothing equations (Liseikin, 2009) (e.g. Laplace-Beltrami using the methods of Chacón and Lapenta (2006); Hansen and Zardecki (2007); Berndt et al. (2008), though we are investigating a modification of the recently proposed mesh motion formulations based on Monge-Kantorovich optimization Delzanno et al. (2008); Chacón et al. (2011) that would be favorable for evolution problems). See Budd et al. (2009) for a recent survey of moving mesh methods.

6.2.3 Solutions

The mesh is read by the analysis code and high-order function spaces for velocity and pressure were defined on it according to runtime parameters, usually Q_3 momentum, Q_2 pressure, and Q_3 energy with SUPG-style stabilization. We solve the steady-state problem (6.1.11) using boundary conditions defined using the three tagged boundary sets. For boundary conditions for momentum are free surface, no-slip at the bed, and velocity at the lateral boundary given by the shallow ice approximation. Due to surface and bed roughness, these lateral boundary conditions are noisier than desired, but the noise can be seen to dissipate in one to two ice thicknesses and thus has little influence on the flow in the vicinity of the ice stream. The boundary conditions for energy are all Dirichlet, with a temperate bed 273.15 K, intermediate temperature 260 K surface, and a cold upstream region at 246.85 A computed flow field is shown in Figure 6.7. A side view of momentum density and energy field is shown in Figure 6.8. Figure 6.9 is a contour plot of temperature/moisture looking upstream. The isosurfaces show the effect of tributaries on thermal structure inside the ice stream.

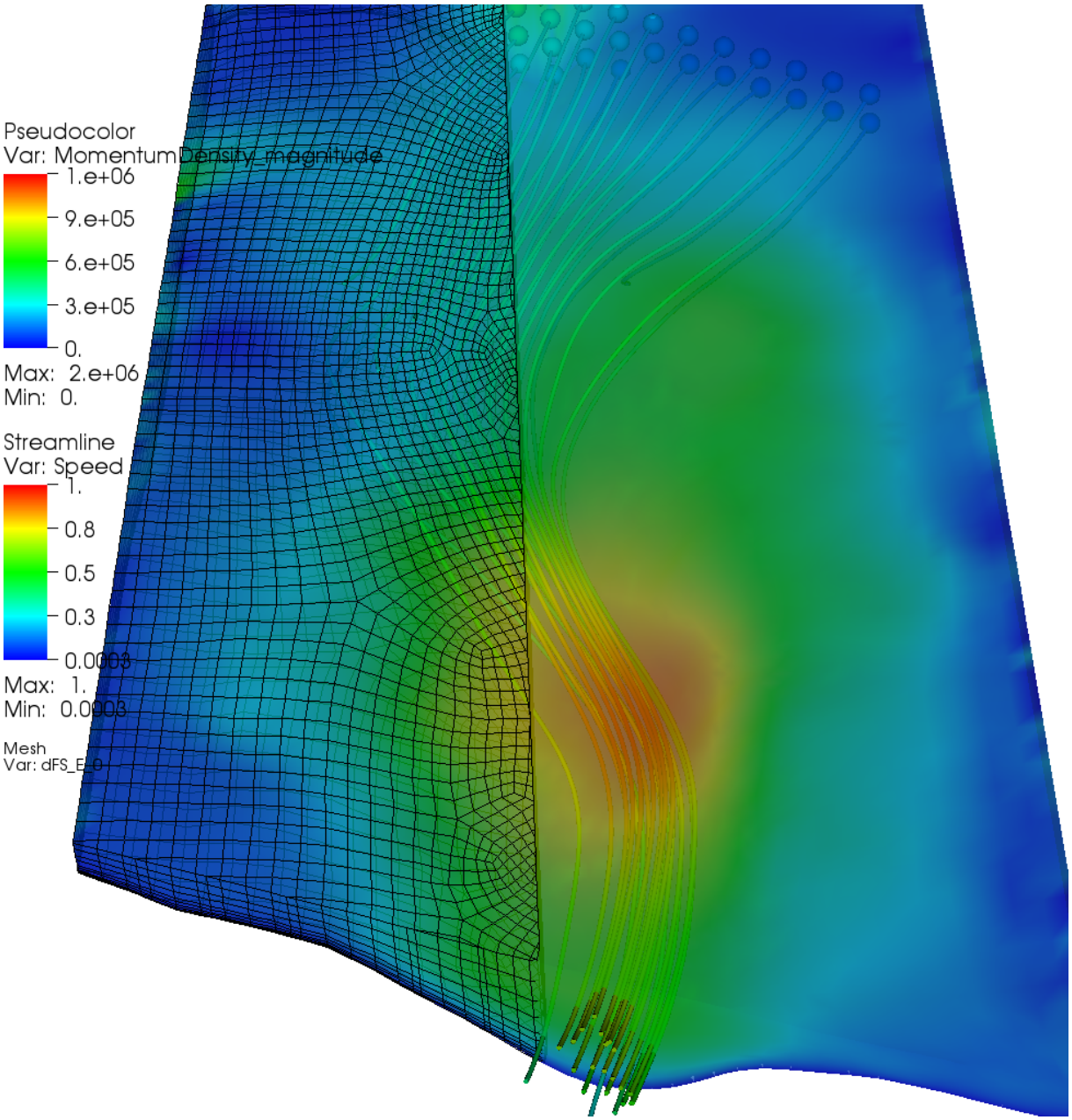

Figure 6.7: Computed momentum density streamlines for the ice stream region at Jakobshavn Isbræ. The highest velocity occurs in a deep, narrow region of the ice stream.

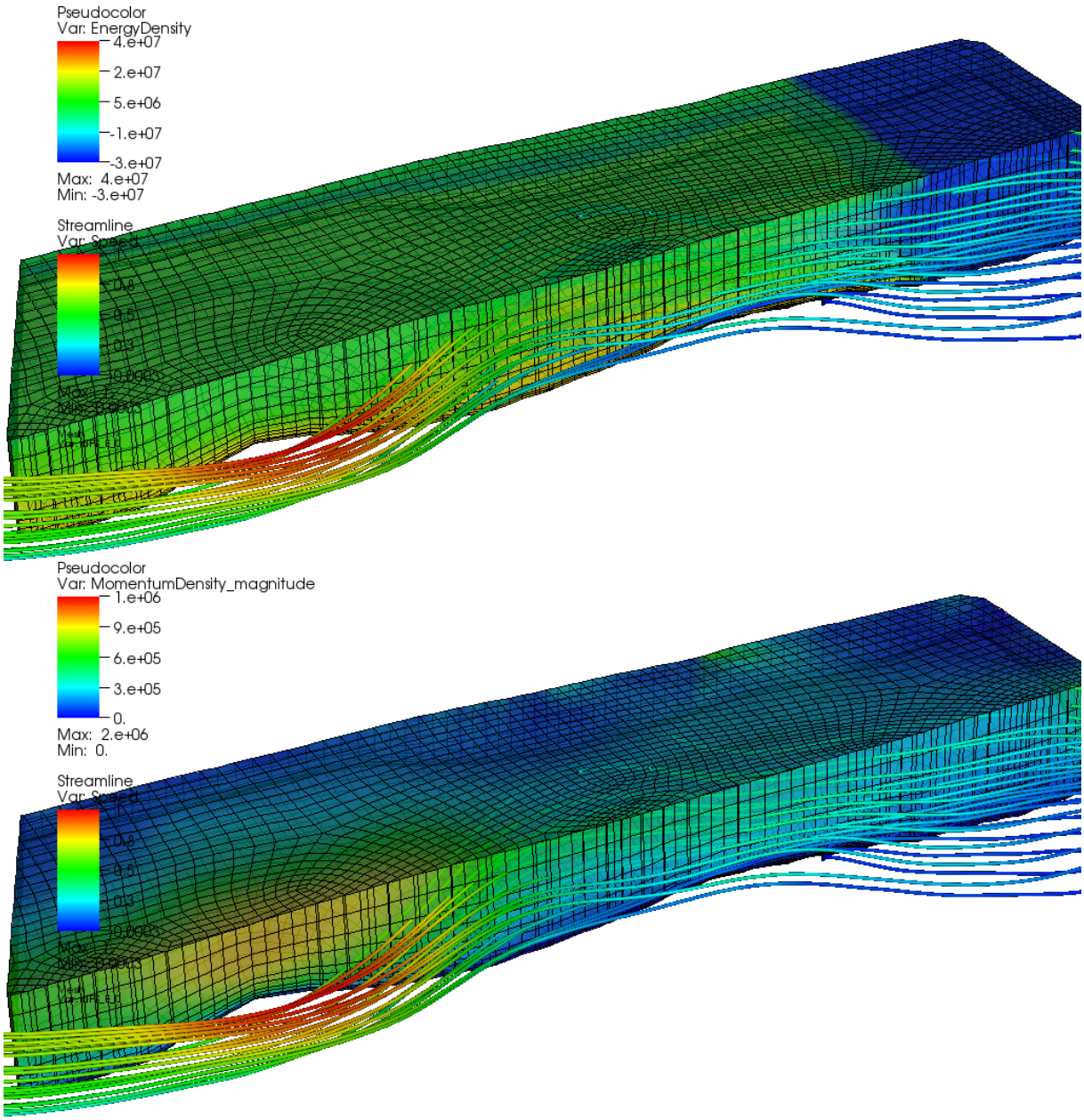

Figure 6.8: Computed energy density (top) and momentum density (bottom) with flow streamlines for the ice stream region at Jakobshavn Isbræ.

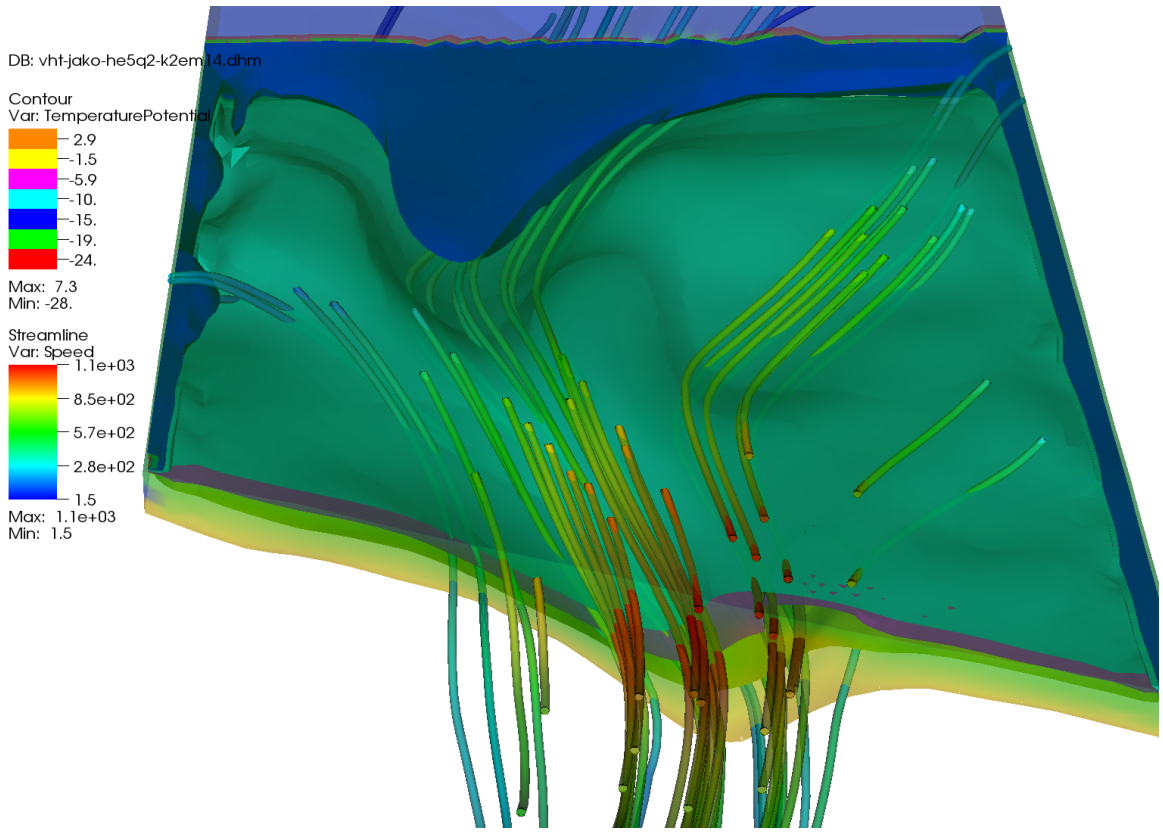

Figure 6.9: Contours of potential temperature (°C) and velocity streamlines (colored by speed measured in $\mathrm{m\,a^{-1}}$). Variability in layer thickness is clearly caused by tributaries and geometry. The maximum potential temperature is 7.3 °C which corresponds to a maximum melt fraction of 4.8 %. The sharp transition from cold upstream boundary condition is just visible at the top of this plot.

The main weakness with the energy transport scheme is that it is overly diffusive and that the boundary layer near the surface is poorly resolved. The cold region in Figure 6.8 does not extend far into the downstream region with warm boundary conditions. The maximum cell Peclet number, after accounting for crosswind numerical stabilization terms, was computed to be 81. As discussed earlier, a physical value for the Peclet number is on the order of 10^4. Unfortunately, SUPG is not capable of delivering stable low-diffusion solutions for such advection-dominated systems. One partial solution is to refine the mesh near the surface, but this is computationally expensive and wasteful in the vast majority of an ice sheet where there is no thermal boundary layer near the surface. In contrast, the velocity field has a boundary layer near the bed almost everywhere, so refinement near the bed is certainly desirable. An additional source of thermal diffusion in Figure 6.8 is the velocity field which is not in equilibrium with the surface. Therefore streamline diffusion, which necessarily produces a stabilized cell Peclet number no larger than 1, causes extra diffusion normal to the surface.

Chapter 7

Conclusion and Outlook

We have presented a variety of numerical methods and software tools for the simulation of ice dynamics. These methods provide increased accuracy, improved performance (especially on emerging hardware), and make the use of more advanced analysis techniques viable. There are still many open problems in the design of computational methods for glacier flow, and many methods known to other fields and widely used for problems with similar characteristics, that have not yet been applied to glaciology. Robust software libraries are often not yet available for many of these techniques, or are not in a form that is directly usable for glaciological problems. The software developed in the present work is often applicable to other problems, and wherever possible, has been generalized and placed into libraries, especially PETSc. This ensures that the maintained library implementations are directly usable for glaciological problems.

7.1 Future directions

Any worthwhile research creates more questions than it answers.

A recurring theme of this thesis has been that computational infrastructure for glaciological problems is immature, thus impeding the use of better numerical methods and limiting the scope of scientific inquiry. The US Department of Energy's Office of Advanced Scientific Computing Research took notice of this issue in 2009 by creating an initiative to improve the software library support and availability of modern computational methods for addressing the scientific problems posed by the IPCC AR4 (Lemke et al., 2007), namely the dynamic response of ice sheets to climate change. The Scalable Ice-sheet Solvers and Infrastructure for Petascale High-resolution Unstructured Simulations (SISIPHUS) (Brown et al., 2009) project was funded under this initiative for 2M USD over three years. I wrote a large fraction of the technical content for the SISIPHUS proposal and there is some overlap with this thesis. Objectives that I will be pursuing in collaboration with other SISIPHUS personnel include an implicit ALE free surface for the Stokes model, automated remeshing after topological changes and large deformations, semi-smooth Newton methods for sliding and grounding line contact problems, bifurcation analysis, and adjoint sensitivity analysis. The rest of this section elaborates on some of these and other techniques useful for analysis of ice dynamics.

There are many periodic processes in glaciology, including ice stream margin migration and shut-down (Raymond, 2000; Bueler and Brown, 2009), Heinrich events (Heinrich, 1988; Calov et al., 10), and grounding line migration (Schoof, 2007; Schoof et al., 2009). All numerical studies to date have explored these phenomena by direct time integration. This involves large spin-up times and many time steps (sometimes millions) to complete a period. An alternative is to pose the time-periodic problem as a nonlinear rootfinding problem and apply Newton-Krylov techniques. This was done by (Merlis and Khatiwala, 2008) for the spin-up of an ocean general circulation model and provided a factor of 10 to 100 speedup compared to direct time integration. Furthermore, it provided access to stable limit cycles and equilibrium solutions that could not be accessed by direct time integration. With such a method, it

would also be possible to efficiently explore the effect of parameters on these cycles. For example, using multi-parameter continuation methods (Allgower and Georg, 2003) to find combinations of parameters that cause a qualitative change in the structure of the periodic solution.

Glacier dynamics are greatly influenced by the basal hydrology which can account for rapid changes in boundary conditions. Basal hydrology is undoubtedly a stiff process, with recent modeling efforts (e.g. Pimentel et al., 2010; Pimentel and Flowers, 2011) requiring very short time steps. The current continuum formulations are not known to be entropy-stable and numerical simulations in two horizontal directions exhibit strong grid dependence (Schoof, 2010). The equations of basal hydrology are variational inequalities in which part of the domain may by strictly hyperbolic (e.g. producing shocks) while other regions may be highly diffusive. The lack of entropy and mixed characteristic makes it difficult to formulate robust discretizations and solvers. I believe that the semi-smooth Newton methods for variational inequalities that were recently added to PETSc will be effective at solving these discrete systems implicitly, and coupled to a flow model.

The symbolic manipulations done to compute manufactured solutions are near their practical limit for the coupled transport problem in Section 6.1 because the symbolic expressions may grow exponentially. The first implementation required half an hour to generate a manufactured solution; the resulting C source file was 11 MB and crashed the compiler when optimization was turned on. The symbolic structure was manipulated somewhat by hand to identify certain intemediate expressions and defer their evaluation to bring code generation time under 1 minute and produce more manageable code size, but this approach will not scale indefinitely. Such limits of symbolic differentiation for finite element computations have been identified by Wang (1986); Fritzson and Fritzson (1992); Korelc (1997, 2002); Wriggers (2008). Algorithmic differentiation Griewank (2003); Griewank and Walther (2008) offers an alternative way to generate high-performance code for these operations. Algorithmic differentiation tools are based either on operator overloading or source transformation, and can run in forward or reverse (adjoint) mode. For high-performance unassembled methods, both forward and reverse mode are needed. Those based on operator overloading, such as Griewank et al. (1996, 1999); Walter (2011), are relatively unintrusive to use, but less scalable and less performant, especially when adjoints are needed. The Python package AlgoPy (Walter, 2011; Walter and Lehmann, 2010) could be used to differentiate the Python code discussed in Section 4.4 and used again in Section 6.1 almost without modification, but the derivatives would still be evaluated in the Python interpreter rather than by compiled C or Fortran code.

Source transformation tools (Bischof et al., 1997; Utke et al., 2008) produce higher performance code and are much more scalable. They can even be applied at a higher level, for example to differentiate the MIT global circulation model (Heimbach et al., 2005). Unfortunately, source transformation AD currently imposes a complicated build process and would be more awkward to use than the current symbolic representations. I am investigating maintainable and less intrusive, but still performant, ways to automate the simultaneous generation of manufactured forcing terms and physics components with Jean Utke, a primary author of OpenAD (Utke et al., 2011; Utke, 2004) and collaborator on the SISIPHUS project. We also intend to use AD for adjoint sensitivity analysis, but with AD applied only in localized parts of the code rather than at a higher level. For example, automatically differentiating iterative solvers is not desirable for performance and algorithmic reasons. Additionally, AD places unacceptable constraints on the software design that can be used when developing discretization and solvers libraries.

An implicit arbitrary Lagrangian-Eulerian (ALE) formulation for the free surface is desirable for accuracy reasons and because it enables the use of more powerful methods for stability, sensitivity, and parameter inversion. ALE formulations for free surface flows have been investigated by several authors, Braess and Wriggers (2000); Walkley et al. (2004); Behr (2004); Cairncross et al. (2000); Baer et al. (2000, e.g.). From a solvers perspective, the most difficult problem is accounting for the "bobbing" mode for a long ice shelf. The transient behavior associated with this mode is much faster than any other dynamics in the system, but it is very poorly captured by the Stokes problem alone. Indeed, the Stokes problem for a detached floating ice berg at equilibrium has six rigid body modes, three in-plane (two translations and one rotation) that have zero energy when coupled to displacement and three modes (two rocking, one

bobbing) that are also zero energy for the Stokes problem, but high energy when the domain is allowed to move under the buoyancy force. When a long thin ice shelf is connected to land, the three in-plane modes are well-constrained due to the strength of in-plane viscous deformation, but rocking and bobbing remains low energy for the Stokes problem while being high energy when the domain moves due to buoyancy This heuristic argument explains why a split preconditioner that separates the mesh motion from the Stokes problem will struggle for long ice shelves and long time steps. This is also the explanation for the transient instability observed by Durand et al. (2009) and stabilized by introducing large non-physical damping. If the buoyancy force is approximately balanced locally by using a coupled solver for both mesh motion and Stokes instead of segregated multigrid on each problem separately, and if a suitable coarse space is chosen, then these bobbing and rocking should be well-controlled by the preconditioner and no artificial damping or time-step restriction should be present.

Scalable methods for large-scale parameter inversion generally come from posing the (ill-posed) inversion as a (well-posed, regularized) deterministic PDE-constrained optimization problem for which Newton and multigrid methods can be applied (Biros and Ghattas, 2005a,b; Akçelik et al., 2006). A significant recent development on this front is Akçelik et al. (2011) which offers the ability to use the Hessian structure from deterministic inversion to rapidly converge on Bayesian (Tarantola, 2005) statistics which were otherwise unattainable in a scalable way. The problem of determining the thermal state or basal conditions from primarily surface observations can be written as such a problem. The reduced Hessian for these problems is spectrally equivalent to a Fredholm integral operator of the second kind. Such operators are well-approximated by low-rank updates to the identity, which means that iterative methods can, in principle, converge very quickly. In practice, despite having rapid decay in the spectrum, a large number of iterations are still required unless the reduced Hessian is preconditioned. Fortunately, the multigrid method, with some modifications relative to differential operators, is effective for Fredholm integral operators of the second kind (Hackbusch, 1985). The key distinction is that for differential operators, the short wavelength modes of the error are also the high energy modes so there is no problem of a smoother polluting the long wavelength components. In contrast, the long wavelength modes are high energy for integral operators, so the smoother needs to be restricted to operating on the short wavelength modes only in order to prevent pollution. In return, multigrid of the second kind converges even faster than multigrid for differential operators. This form of multigrid preconditioning has been used successfully for a variety of inverse and optimization problems of similar flavor to parameter inversion in glaciology, (Akçelik et al., 2005; Biros and Dogan, 2008; Rees et al., 2010). Regardless of the details of the optimization method, the adjoint of the PDE is a required ingredient. Therefore the adjoint of the PDE serves at least three purposes: error estimation, sensitivity analysis, and optimization/inversion.

References

Adams, M. (2004). Algebraic multigrid methods for constrained linear systems with applications to contact problems in solid mechanics. *Numerical Linear Algebra with Applications*, 11(23):141–153.

Adams, M., Samtaney, R., and Brandt, A. (2010). Toward textbook multigrid efficiency for fully implicit resistive magnetohydrodynamics. *Journal of Computational Physics*, 229(18):6208–6219.

Ainsworth, M. and Coggins, P. (2000). The stability of mixed *hp*-finite element methods for Stokes flow on high aspect ratio elements. *SIAM Journal on Numerical Analysis*, 38(5):1721–1761.

Ainsworth, M. and Kay, D. (1999). The approximation theory for the *p*-version finite element method and application to non-linear elliptic PDEs. *Numerische Mathematik*, 82(3):351–388.

Ainsworth, M. and Oden, J. (1997). A posteriori error estimation in finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, 142(1-2):1–88.

Ainsworth, M. and Senior, B. (1998). An adaptive refinement strategy for *hp*-finite element computations. *Applied Numerical Mathematics*, 26(1):165–178.

Akçelik, V., Biros, G., Draganescu, A., Hill, J., Ghattas, O., and van Bloemen Waanders, B. (2005). Dynamic data-driven inversion for terascale simulations: real-time identification of airborne contaminants. *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*.

Akçelik, V., Biros, G., Ghattas, O., Hill, J., Keyes, D., and van Bloemen Waanders, B. (2006). Parallel algorithms for PDE-constrained optimization. *Parallel Processing for Scientific Computing*, pages 291–322.

Akçelik, V., Flath, H., Ghattas, O., Hill, J., van Bloemen Waanders, B., and Wllcox, L. (2011). Fast algorithms for Bayesian uncertainty quantification in large-scale linear inverse problems based on low-rank partial Hessian approximations. *SIAM Journal on Scientific Computing*, 33(1).

Akkerman, I., Bazilevs, Y., Kees, C., and Farthing, M. (2011). Isogeometric analysis of free-surface flow. *Journal of Computational Physics*, 230(11):4137–4152. Special issue High Order Methods for CFD Problems.

Allgower, E. and Georg, K. (2003). *Introduction to Numerical Continuation Methods*. Society for Industrial and Applied Mathematics Philadelphia, PA, USA.

Alnaes, M., Logg, A., Mardal, K., and Skavhaug, O. (2009). Unified framework for finite element assembly. *International Journal of Computational Science and Engineering*, 4(4):231–244.

AMD (2011). *Software Optimization Guide for AMD Family 10h and 12h Processors*. Number 40546-3.13 in AMD Technical Report. Advanced Micro Devices.

American Society of Mechanical Engineers (2004). ASME Journal of Fluids Engineering editorial policy statement on the control of numerical accuracy.

Amundson, J., Truffer, M., Lüthi, M., Fahnestock, M., West, M., and Motyka, R. (2008). Glacier, fjord, and seismic response to recent large calving events, jakobshavn isbræ, greenland. *Geophys. Res. Lett*, 35(22):L22501.

Amundson, J. M., Fahnestock, M., Truffer, M., Brown, J., Lüthi, M. P., and Motyka, R. J. (2010). Ice mélange dynamics and implications for terminus stability, Jakobshavn Isbræ, Greenland. *J. Geophys. Res*, 115:F01005.

Anderson, T. (2005). *Fracture mechanics: fundamentals and applications*. CRC.

Aschwanden, A. and Blatter, H. (2009). Mathematical modeling and numerical simulation of polythermal glaciers. *Journal of Geophysical Research-Earth Surface*, 114(F1):F01027.

Aschwanden, A., Bueler, E., Khroulev, C., and Blatter, H. (*submitted* 2011). An enthalpy formulation for glaciers and ice sheets. *Journal of Glaciology*.

Asokan, B. and Zabaras, N. (2006). A stochastic variational multiscale method for diffusion in heterogeneous random media. *Journal of Computational Physics*, 218(2):654–676.

Awanou, G. and Lai, M. (2005). On convergence rate of the augmented Lagrangian algorithm for nonsymmetric saddle point problems. *Applied Numerical Mathematics*, 54(2):122–134.

Babuška, I. and Narasimhan, R. (1997). The Babuška-Brezzi condition and the patch test: an example. *Comput. Methods Appl. Mech. Engrg*, 140(1-2):183–199.

Babuška, I., Nobile, F., and Tempone, R. (2005). A stochastic collocation method for elliptic partial differential equations with random input data. *ICES Report*, pages 05–47.

Babuška, I. and Oden, J. (2004). Verification and validation in computational engineering and science: basic concepts. *Computer Methods in Applied Mechanics and Engineering*, 193(36-38):4057–4066.

Bacuta, C., Bramble, J., and Xu, J. (2003). Regularity estimates for elliptic boundary value problems in Besov spaces. *Mathematics of Computation*, 72(244):1577–1595.

Baer, T., Cairncross, R., Schunk, P., Rao, R., and Sackinger, P. (2000). A finite element method for free surface flows of incompressible fluids in three dimensions. Part II. Dynamic wetting lines. *International Journal for Numerical Methods in Fluids*, 33(3):405–427.

Balay, S., Brown, J., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H. (2012a). PETSc users manual. Technical Report ANL-95/11 - Revision 3.3, Argonne National Laboratory.

Balay, S., Brown, J., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H. (2012b). PETSc Web page. `http://www.mcs.anl.gov/petsc`.

Bangerth, W., Hartmann, R., and Kanschat, G. (2007). deal.II—A general-purpose object-oriented finite element library. *ACM Transactions on Mathematical Software (TOMS)*, 33(4):24–es.

Barth, A., Schwab, C., and Zollinger, N. (2011). Multi-level Monte Carlo finite element method for elliptic PDEs with stochastic coefficients. *Numerische Mathematik*, 119:123–161. 10.1007/s00211-011-0377-0.

Bathe, K. (1996). Finite element procedures. *Englewood Cliffs, New Jersey*.

Bazilevs, Y., Calo, V., Cottrell, J., Evans, J., Hughes, T., Lipton, S., Scott, M., and Sederberg, T. (2010). Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):229–263.

Becker, R. and Rannacher, R. (1994). *Finite element solution of the incompressible Navier-Stokes equations on anisotropically refined meshes*. Interdisziplinäres Zentrum für Wissenschaftl. Rechnen, Univ.

Behr, M. (2004). On the application of slip boundary condition on curved boundaries. *International Journal for Numerical Methods in Fluids*, 45(1):43–51.

Belytschko, T., Gracie, R., and Ventura, G. (2009). A review of extended/generalized finite element methods for material modeling. *Modelling and Simulation in Materials Science and Engineering*, 17:043001.

Benson, D. J., Bazilevs, Y., Luycker, E. D., Hsu, M.-C., Scott, M., Hughes, T. J. R., and Belytschko, T. (2010). A generalized finite element formulation for arbitrary basis functions: From isogeometric analysis to XFEM. *International Journal for Numerical Methods in Engineering*.

Benzi, M., Golub, G., and Liesen, J. (2005). Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137.

Bernardi, C. and Maday, Y. (1999). Uniform inf-sup conditions for the spectral discretization of the Stokes problem. *Mathematical Models and Methods in Applied Sciences*, 9(3):395–414.

Berndt, M., Moulton, D., and Hansen, G. (2008). Efficient nonlinear solvers for Laplace-Beltrami smoothing of three-dimensional unstructured grids. *Computers & Mathematics with Applications*, 55(12):2791–2806.

Bernstein, L. et al. (2008). Climate Change 2007: Synthesis report, Intergovernmental Panel on Climate Change. `www.ipcc.ch/ipccreports/ar4-syr.htm`.

Biros, G. and Dogan, G. (2008). A multilevel algorithm for inverse problems with elliptic PDE constraints. *Inverse Problems*, 24:034010.

Biros, G. and Ghattas, O. (2005a). Parallel Lagrange-Newton-Krylov-Schur methods for PDE constrained optimization. Part I: The Krylov-Schur solver. *SIAM Journal on Scientific Computing*, 27(2):687–713.

Biros, G. and Ghattas, O. (2005b). Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part II: The Lagrange Newton solver, and its application to optimal control of steady viscous flows. *SIAM Journal on Scientific Computing*, 27(2):714–739.

Bischof, C., Roh, L., and Mauer-Oats, A. (1997). ADIC: an extensible automatic differentiation tool for ANSI-C. *Urbana*, 51:61802.

Blacker, T., Bohnhoff, W., Edwards, T., Hipp, J., Lober, R., Mitchell, S., Sjaardema, G., Tautges, T., Wilson, T., Oakes, W., et al. (1994). CUBIT mesh generation environment. Technical report, Sandia National Labs., Albuquerque, NM. Cubit Development Team.

Blatter, H. (1995). Velocity and stress fields in grounded glaciers: A simple algorithm for including deviatoric stress gradients. *Journal of Glaciology*, 41(138):333–344.

Boris, J. and Book, D. (1973). Flux-corrected transport. I. SHASTA, a fluid transport algorithm that works. *Journal of Computational Physics*, 11(1):38–69.

Braess, H. and Wriggers, P. (2000). Arbitrary Lagrangian Eulerian finite element analysis of free surface flow. *Computer Methods in Applied Mechanics and Engineering*, 190(1-2):95–109.

Brakkee, E., Vuik, C., and Wesseling, P. (1998). Domain decomposition for the incompressible Navier-Stokes equations: solving subdomain problems accurately and inaccurately. *International Journal for Numerical Methods in Fluids*, 26(10):1217–1237.

Brandt, A. and Dinar, N. (1979). Multigrid solutions to elliptic flow problems. *Numerical Methods for Partial Differential Equations*, pages 53–147.

Brenner, S. (2004). Korn's inequalities for piecewise H^1 vector fields. *Mathematics of Computation*, 73:1067–1088.

Brenner, S. and Scott, L. (2008). *The mathematical theory of finite element methods*. Springer Verlag.

Brezzi, F. and Fortin, M. (1991). Mixed and hybrid finite element methods. *Springer Series In Computational Mathematics; Vol. 15*, page 350.

Brooks, A. and Hughes, T. (1982). Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Eng.*, 32:199–259.

Brown, J., Heimbach, P., Jacob, R., Karpeev, D., Lipscomb, W., Smith, B., Tautges, T., and Utke, J. (2009). SISIPHUS: Scalable Ice-Sheet Solvers and Infrastructure for Petascale, High-Resolution, Unstructured Simulations.

Budd, C., Huang, W., and Russell, R. (2009). Adaptivity with moving grids. *Acta Numerica*, 18(-1):111–241.

Bueler, E. and Brown, J. (2009). Shallow shelf approximation as a "sliding law" in a thermomechanically coupled ice sheet model. *Journal of Geophysical Research-Earth Surface*, 114(F3):F03008.

Burstedde, C., Ghattas, O., Gurnis, M., Stadler, G., Tan, E., Tu, T., Wilcox, L., and Zhong, S. (2008). Scalable adaptive mantle convection simulation on petascale supercomputers. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, page 62. IEEE Press.

Büskens, C. and Maurer, H. (2000). Sqp-methods for solving optimal control problems with control and state constraints: adjoint variables, sensitivity analysis and real-time control. *Journal of Computational and Applied Mathematics*, 120(1-2):85–108.

Butcher, J. (2006). General linear methods. *Acta Numerica*, 15:157–256.

Butcher, J., Jackiewicz, Z., and Wright, W. (2007). Error propagation of general linear methods for ordinary differential equations. *Journal of Complexity*, 23(4-6):560–580.

Butcher, J. and Podhaisky, H. (2006). On error estimation in general linear methods for stiff odes. *Applied numerical mathematics*, 56(3-4):345–357.

Cai, X. and Sarkis, M. (1999). A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing*, 21(2):797.

Cairncross, R., Schunk, P., Baer, T., Rao, R., and Sackinger, P. (2000). A finite element method for free surface flows of incompressible fluids in three dimensions. Part I. Boundary fitted mesh motion. *International Journal for Numerical Methods in Fluids*, 33(3):375–403.

Calov, R., Greve, R., Abe-Ouchi, A., Bueler, E., Huybrechts, P., Johnson, J., Pattyn, F., Pollard, D., Ritz, C., Saito, F., and Tarasov, L. (10). Results from the Ice-Sheet Model Intercomparison Project–Heinrich Event INtercOmparison (ISMIP HEINO). *Journal of Glaciology*, 56(197):371–383.

Certik, O. et al. (2011). SymPy Python library for symbolic mathematics. `http://code.google.com/p/sympy/`.

Chacón, L. (2008). An optimal, parallel, fully implicit Newton-Krylov solver for three-dimensional viscoresistive magnetohydrodynamics. *Physics of Plasmas*, 15:056103.

Chacón, L., Delzanno, G., and Finn, J. (2011). Robust, multidimensional mesh-motion based on Monge-Kantorovich equidistribution. *Journal of Computational Physics*, 230(1):87–103.

Chacón, L. and Lapenta, G. (2006). A fully implicit, nonlinear adaptive grid strategy. *Journal of Computational Physics*, 212(2):703–717.

Chapelle, D. and Bathe, K. (1993). The inf-sup test. *Computers and Structures*, 47:537–537.

Childs, H. and Miller, M. (2006). Beyond meat grinders: An analysis framework addressing the scale and complexity of large data sets. *Simulation Series*, 38(1):181.

Chow, P. (2007). *Stochastic partial differential equations*. Chapman & Hall/CRC.

Chow, S., Carey, G., and Anderson, M. (2004). Finite element approximations of a glaciology problem. *Mathematical Modelling and Numerical Analysis*, 38(5):741–756.

Ciarlet, P. (1978). *The finite element method for elliptic problems*. North-Holland.

Clarke, G. (2004). Subglacial processes. *Annual Review of Earth and Planetary Sciences*, 33(1):247.

Coffey, T. S., Kelley, C. T., and Keyes, D. E. (2003). Pseudotransient continuation and differential-algebraic equations. *SIAM Journal on Scientific Computing*, 25(2):553–569.

Colinge, J. and Rappaz, J. (1999). A strongly nonlinear problem arising in glaciology. *Mathematical Modelling and Numerical Analysis*, 33(2):395–406.

Collins, W., Bitz, C., Blackmon, M., Bonan, G., Bretherton, C., Carton, J., Chang, P., Doney, S., Hack, J., Henderson, T., et al. (2006). The community climate system model version 3 (CCSM3). *Journal of Climate*, 19(11):2122–2143.

Corbett, P., Feitelson, D., Fineberg, S., Hsu, Y., Nitzberg, B., Prost, J., Snir, M., Traversat, B., and Wong, P. (1995). Overview of the MPI-IO parallel I/O interface.

Cottrell, J., Hughes, T., and Bazilevs, Y. (2009). *Isogeometric Analysis: Toward Integration of CAD and FEA*. Wiley.

Cuthill, E. and McKee, J. (1969). Reducing the bandwidth of sparse symmetric matrices. In *ACM Annual Conference/Annual Meeting: Proceedings of the 1969 24th national conference*, pages 157–172. ACM.

Davies, A. (1988). Reentrant corner singularities in non-Newtonian flow. Part I: Theory. *Journal of Non-Newtonian Fluid Mechanics*, 29:269–293.

de Niet, A. (2007). *Solving large linear systems in an implicit thermohaline ocean model*. PhD thesis, University of Groningen.

de Niet, A. and Wubs, F. (2007). Two preconditioners for saddle point problems in fluid flows. *Int. J. Numer. Meth. Fluids*, 54:355–377.

De Smedt, B., Pattyn, F., and De Groen, P. (2010). Using the unstable manifold correction in a Picard iteration to solve the velocity field in higher-order ice-flow models. *Journal of Glaciology*, 56(196):257–261.

Deb, M., Babuška, I., and Oden, J. (2001). Solution of stochastic partial differential equations using Galerkin finite element techniques. *Computer Methods in Applied Mechanics and Engineering*, 190(48):6359–6372.

Delzanno, G., Chacón, L., Finn, J., Chung, Y., and Lapenta, G. (2008). An optimal robust equidistribution method for two-dimensional grid adaptation based on Monge-Kantorovich optimization. *Journal of Computational Physics*, 227(23):9841–9864.

Demkowicz, L., Oden, J., Rachowicz, W., and Hardy, O. (1989). Toward a universal *hp* adaptive finite element strategy. I: Constrained approximation and data structure. *Comput. Methods Appl. Mech. Engrg*, 77:79–112.

Demkowicz, L., Rachowicz, W., and Devloo, P. (2002). A fully automatic *hp*-adaptivity. *Journal of Scientific Computing*, 17(1):117–142.

Deville, M. and Mund, E. (1985). Chebyshev pseudospectral solution of second-order elliptic equations with finite element preconditioning. *Journal of Computational Physics*, 60:517.

Deville, M. and Mund, E. (1990). Finite-element preconditioning for pseudospectral solutions of elliptic problems. *SIAM Journal on Scientific and Statistical Computing*, 11:311.

Dohrmann, C. (2003). A preconditioner for substructuring based on constrained energy minimization. *SIAM Journal on Scientific Computing*, 25:246.

Dohrmann, C. and Bochev, P. (2004). A stabilized finite element method for the Stokes problem based on polynomial pressure projections. *International Journal for Numerical Methods in Fluids*, 46(2):183–201.

Dohrmann, C. and Lehoucq, R. (2006). A primal based penalty preconditioner for elliptic saddle point systems. *SIAM Journal on Numerical Analysis*, 44(1):270–282.

Dohrmann, C. and Widlund, O. (2010). Hybrid domain decomposition algorithms for compressible and almost incompressible elasticity. *International Journal for Numerical Methods in Engineering*, 82(2):157–183.

Donea, J., Huerta, A., Ponthot, J.-P., and Rodríguez-Ferran, A. (2004). *Arbitrary Lagrangian-Eulerian Methods*, chapter 14, pages 413–437. John Wiley & Sons, Ltd.

Drepper, U. (2007). What every programmer should know about memory. Linux Weekly News, `http://people.redhat.com/drepper/cpumemory.pdf`.

Durand, G., Gagliardini, O., de Fleurian, B., Zwinger, T., and Le Meur, E. (2009). Marine ice sheet dynamics: Hysteresis and neutral equilibrium. *Journal of Geophysical Research*, 114(F3):F03009.

Egholm, D., Knudsen, M., Clark, C., and Lesemann, J. (2011). Modeling the flow of glaciers in steep terrains: The integrated second-order shallow ice approximation (iSOSIA). *Journal of Geophysical Research*, 116(F2):F02012.

Eisenstat, S., Elman, H., and Schultz, M. (1983). Variational iterative methods for nonsymmetric systems of linear equations. *SIAM Journal on Numerical Analysis*, 20(2):345–357.

Eisenstat, S. C. and Walker, H. F. (1996). Choosing the forcing terms in an inexact Newton method. *SIAM Journal on Scientific Computing*, 17(1):16–32.

Elguedj, T., Gravouil, A., and Combescure, A. (2006). Appropriate extended functions for X-FEM simulation of plastic fracture mechanics. *Computer Methods in Applied Mechanics and Engineering*, 195(7-8):501 – 515.

Elman, H. (1999). Preconditioning for the steady-state Navier-Stokes equations with low viscosity. *SIAM Journal on Scientific Computing*, 20(4):1299–1316.

Elman, H. (2005). Preconditioning strategies for models of incompressible flow. *Journal of Scientific Computing*, 25(1):347–366.

Elman, H., Howle, V., Shadid, J., Shuttleworth, R., and Tuminaro, R. (2006). Block preconditioners based on approximate commutators. *SIAM Journal on Scientific Computing*, 27(5):1651–1668.

Elman, H., Howle, V., Shadid, J., Shuttleworth, R., and Tuminaro, R. (2008). A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 227(1):1790–1808.

Elman, H., Mihajlovic, M., and Silvester, D. (2011). Fast iterative solvers for buoyancy driven flow problems. *Journal of Computational Physics*, 230(10):3900 – 3914.

Elman, H. and Tuminaro, R. (2009). Boundary conditions in approximate commutator preconditioners for the Navier-Stokes equations. *Electronic Transactions on Numerical Analysis*, 35:257–280.

Embree, M. (1999). How descriptive are GMRES convergence bounds. Technical Report 08, Oxford University Computing Laboratory.

Erdogan, F. and Biricikoglu, V. (1973). Two bonded half planes with a crack going through the interface 1. *International Journal of Engineering Science*, 11(7):745–766.

Estep, D., Ginting, V., Ropp, D., Shadid, J., and Tavener, S. (2008). An a posteriori-a priori analysis of multiscale operator splitting. *SIAM Journal on Numerical Analysis*, 46(3):1116–1146.

Evans, K., Salinger, A., Barker, E., Holland, D., Knoll, D., LeMieux, J. F., Lipscomb, B., Nong, R., Price, S., Roddy, K., White, T., and Worley, P. (2010). A Scalable, Efficient, and Accurate Community Ice Sheet Model. `www.csm.ornl.gov/SEACISM`.

Evans, L. (1998). *Partial Differential Equations*. Graduate Studies in Mathematics, Vol. 19. American Mathematics Society.

Evans, L. (2007). The 1-Laplacian, the ∞-Laplacian and differential games. *Perspectives in Nonlinear Partial Differential Equations: In Honor of Haim Brezis*, 446:245.

Farhat, C., Lesoinne, M., LeTallec, P., Pierson, K., and Rixen, D. (2001). FETI-DP: a dual–primal unified FETI method—Part I: A faster alternative to the two-level FETI method. *International Journal for Numerical Methods in Engineering*, 50(7):1523–1544.

Farhat, C., Lesoinne, M., and Pierson, K. (2000). A scalable dual-primal domain decomposition method. *Numerical linear algebra with applications*, 7(7-8):687–714.

Fischer, P., Kruse, J., Mullen, J., Tufo, H., Lottes, J., and Kerkemeier, S. (2008). NEK5000–open source spectral element CFD solver.

Flowers, G. and Clarke, G. (2002a). A multicomponent coupled model of glacier hydrology 1. Theory and synthetic examples. *Journal of Geophysical Research*, 107(B11):2287.

Flowers, G. and Clarke, G. (2002b). A multicomponent coupled model of glacier hydrology 2. Application to Trapridge Glacier, Yukon, Canada. *Journal of Geophysical Research*, 107(B11):2288.

Fog, A. (2011a). Instruction tables: Lists of instruction latencies, throughputs and microoperation breakdowns for Intel, AMD and VIA CPUs.

Fog, A. (2011b). The microarchitecture of Intel and AMD CPU's: An optimization guide for assembly programmers and compiler makers.

Fowler, A. (2001). Modelling the flow of glaciers and ice sheets. *Continuum Mechanics and Applications in Geophysics and the Environment*, pages 201–221.

Fowler, M. (2004). Inversion of control containers and the dependency injection pattern.

Fritzson, P. and Fritzson, D. (1992). The need for high-level programming support in scientific computing applied to mechanical analysis. *Computers & structures*, 45(2):387–395.

Fujita, H. (2002). A coherent analysis of Stokes flows under boundary conditions of friction type. *Journal of Computational and Applied Mathematics*, 149(1):57–69.

Funk, M., Echelmeyer, K., and Iken, A. (1994). Mechanisms of fast flow in Jakobshavns Isbrae, Greenland; Part II: Modeling of englacial temperatures. *Journal of Glaciology*, 40(136):569–585.

Ganapathysubramanian, B. and Zabaras, N. (2007). Sparse grid collocation schemes for stochastic natural convection problems. *Journal of Computational Physics*, 225(1):652–685.

Gee, M., Siefert, C., Hu, J., Tuminaro, R., and Sala, M. (2006). ML 5.0 smoothed aggregation user's guide. Technical Report SAND2006-2649, Sandia National Laboratories.

Ghanem, R. and Spanos, P. (2003). *Stochastic Finite Elements: A Spectral Approach*. Dover Publications.

Glen, J. W. (1955). The creep of polycrystalline ice. *Proceedings of the Royal Society of London, Ser. A*, 228(1175):519–538.

Glowinski, R. and Rappaz, J. (2003). Approximation of a nonlinear elliptic problem arising in a non-Newtonian fluid flow model in glaciology. *Mathematical Modelling and Numerical Analysis*, 37(1):175–186.

Goldberg, D. (1991). What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48.

Goldberg, D. (2011). A variationally derived, depth-integrated approximation to a higher-order glaciological flow model. *Journal of Glaciology*, 57(201):157–170.

Goldberg, D., Holland, D., and Schoof, C. (2009). Grounding line movement and ice shelf buttressing in marine ice sheets. *Journal of Geophysical Research*, 114(F4):F04026.

Gottlieb, S., Ketcheson, D., and Shu, C. (2009). High order strong stability preserving time discretizations. *Journal of Scientific Computing*, 38(3):251–289.

Gottlieb, S., Shu, C., and Tadmor, E. (2001). Strong Stability-Preserving High-Order Time Discretization Methods. *SIAM Review*, 43:89.

Greve, R. (1997). A continuum-mechanical formulation for shallow polythermal ice sheets. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 355(1726):921–974.

Greve, R. and Blatter, H. (2009). *Dynamics of ice sheets and glaciers*. Springer Verlag.

Griewank, A. (2003). A mathematical view of automatic differentiation. *Acta Numerica*, 12(-1):321–398.

Griewank, A., Juedes, D., Mitev, H., Utke, J., Vogel, O., and Walther, A. (1999). ADOL-C: A package for the automatic differentiation of algorithms written in C/C++.

Griewank, A., Juedes, D., Srinivasan, J., et al. (1996). Adol-c, a package for the automatic differentiation of algorithms written in c/c++. *ACM Trans. Math. Software*, 22:131–167.

Griewank, A. and Walther, A. (2008). *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Society for Industrial and Applied Mathematics (SIAM).

Griffiths, D. (1997). The "no boundary condition" outflow boundary condition. *International Journal for Numerical Methods in Fluids*, 24:393–411.

Grisvard, P. (1985). *Elliptic problems in nonsmooth domains*. Pitman Advanced Pub. Program.

Gropp, W. (1999). Exploiting existing software in libraries: Successes, Failures, and Reasons Why. In *Proceedings of the SIAM Workshop on Object Oriented Methods for Interoperable Scientific and Engineering Computing*, pages 21–29. Soc for Industrial & Applied Math, SIAM.

Gropp, W., Keyes, D., Mcinnes, L., and Tidriri, M. (2000a). Globalized Newton-Krylov-Schwarz algorithms and software for parallel implicit CFD. *International Journal of High Performance Computing Applications*, 14(2):102.

Gropp, W. D., Kaushik, D. K., Keyes, D. E., and Smith, B. (2000b). Performance modeling and tuning of an unstructured mesh CFD application. In *Supercomputing '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing*, page 34, Washington, DC, USA. IEEE Computer Society.

Grote, M. and Huckle, T. (1997). Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18(3):838–853.

Gudmundsson, G. and Raymond, M. (2008). On the limit to resolution and information on basal properties obtainable from surface data on ice streams. *The Cryosphere*, 2:167–178.

Hackbusch, W. (1985). *Multi-grid methods and applications*. Springer Verlag.

Hairer, E. and Wanner, G. (1999). Stiff differential equations solved by Radau methods. *Journal of Computational and Applied Mathematics*, 111(1-2):93–111.

Hairer, E. and Wanner, G. (2010). *Solving ordinary differential equations II: Stiff and differential-algebraic problems*, volume 14. Springer Verlag.

Hansen, G. and Zardecki, A. (2007). Unstructured surface mesh adaptation using the Laplace-Beltrami target metric approach. *Journal of Computational Physics*, 225(1):165–182.

Harten, A. (1983). High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics*, 49(3):357–393.

Harten, A., Engquist, B., Osher, S., and Chakravarthy, S. (1987). Uniformly high order accurate essentially non-oscillatory schemes, III. *Journal of Computational Physics*, 71(2):231–303.

HDF5 Group (2011). HDF5 web site. `http://www.hdfgroup.org/HDF5/`.

Heimbach, P. and Bugnion, V. (2009). Greenland ice-sheet volume sensitivity to basal, surface and initial conditions derived from an adjoint model. *Annals of Glaciology*, 50(52):67–80.

Heimbach, P., Hill, C., and Giering, R. (2005). An efficient exact adjoint of the parallel mit general circulation model, generated via automatic differentiation. *Future Generation Computer Systems*, 21(8):1356–1371.

Heinrich, H. (1988). Origin and consequences of cyclic ice rafting in the Northeast Atlantic Ocean during the past 130,000 years. *Quaternary Research*, 29(2):142 – 152.

Henderson, A., Ahrens, J., and Law, C. (2004). *The Paraview Guide*. Kitware.

Henson, V. and Yang, U. (2002). BoomerAMG: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41(1):155–177.

Herman, F. and Braun, J. (2008). Evolution of the glacial landscape of the Southern Alps of New Zealand: Insights from a glacial erosion model. *J. Geophys. Res*, 113:F02009.

Hesthaven, J. and Warburton, T. (2008). *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Verlag.

Heys, J., Manteuffel, T., McCormick, S., and Olson, L. (2005). Algebraic multigrid for higher-order finite elements. *Journal of Computational Physics*, 204(2):520–532.

Higham, N. (2002). *Accuracy and stability of numerical algorithms*. Society for Industrial Mathematics.

Hinch, E. (1993). The flow of an Oldroyd fluid around a sharp corner. *Journal of Non-Newtonian Fluid Mechanics*, 50(2-3):161–171.

Hindmarsh, R. (2004). A numerical comparison of approximations to the Stokes equations used in ice sheet and glacier modeling. *Journal of Geophysical Research*, 109(F1):F01012.

Hirata, S. (2003). Tensor contraction engine: Abstraction and automated parallel implementation of configuration-interaction, coupled-cluster, and many-body perturbation theories. *The Journal of Physical Chemistry A*, 107(46):9887–9897.

Hughes, T., Cottrell, J., and Bazilevs, Y. (2005). Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer methods in applied mechanics and engineering*, 194(39-41):4135–4195.

Hughes, T., Engel, G., Mazzei, L., and Larson, M. (2000). The continuous Galerkin method is locally conservative. *Journal of Computational Physics*, 163(2):467–488.

Hughes, T., Feijóo, G., Mazzei, L., and Quincy, J. (1998). The variational multiscale method–a paradigm for computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 166(1-2):3–24.

Hughes, T., Franca, L., and Balestra, M. (1986). A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuška-Brezzi condition: A stable Petrov-Galerkin formulation of the Stokes problem accommodating equal-order interpolations. *Computer Methods in Applied Mechanics and Engineering*, 59(1):85–99.

Hughes, T., Franca, L., and Hulbert, G. (1989). A new finite element formulation for computational fluid dynamics: VIII. the Galerkin/least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering*, 73(2):173–189.

Hutchinson, J. (1968). Singular behaviour at the end of a tensile crack in a hardening material. *Journal of the Mechanics and Physics of Solids*, 16(1):13–31.

Hutter, K. (1982). A mathematical model of polythermal glaciers and ice sheets. *Geophysical and Astrophysical Fluid Dynamics*, 21:201–224.

Hutter, K. (1983). *Theoretical glaciology: material science of ice and the mechanics of glaciers and ice sheets*. Springer.

Hutter, K. and Jöhnk, K. (2004). *Continuum methods of physical modeling*. Springer.

Intel (2011). *Intel 64 and IA-32 Architectures Optimization Reference Manual*. Number 248966-024 in Intel Technical Report. Intel.

Ipsen, I. C. F. (2001). A note on preconditioning nonsymmetric matrices. *SIAM J. Sci. Comput.*, 23:1050–1051.

ITAPS (2011). Interoperable Technologies for Advanced Petascale Simulations. http://www.itaps.org/.

Iverson, N., Hooyer, T., and Baker, R. (1998). Ring-shear studies of till deformation: Coulomb-plastic behavior and distributed strain in glacier beds. *Journal of Glaciology*, 44(148):634–642.

Jacobson, H. and Raymond, C. (1998). Thermal effects on the location of ice stream margins. *Journal of Geophysical Research*, 103(B6):12111.

Jameson, A. and Caughey, D. (2001). How many steps are required to solve the Euler equations of steady, compressible flow: in search of a fast solution algorithm. *AIAA paper*, 2673:2001.

Jardin, S., Bateman, G., Hammett, G., and Ku, L. (2008). On 1D diffusion problems with a gradient-dependent diffusion coefficient. *Journal of Computational Physics*, 227(20):8769–8775.

Jiang, G. and Shu, C. (1996). Efficient implementation of weighted ENO schemes. *Journal of Computational Physics*, 126(1):202–228.

Jiang, Y., Liu, G., Zhang, Y., Chen, L., and Tay, T. (2011). A singular ES-FEM for plastic fracture mechanics. *Computer Methods in Applied Mechanics and Engineering*, In Press, Accepted Manuscript:– .

John, V. and Knobloch, P. (2007). On spurious oscillations at layers diminishing (SOLD) methods for convection-diffusion equations: Part I—A review. *Computer Methods in Applied Mechanics and Engineering*, 196(17-20):2197–2215.

John, V. and Knobloch, P. (2008). On spurious oscillations at layers diminishing (SOLD) methods for convection-diffusion equations: Part II—Analysis for p_1 and q_1 finite elements. *Computer Methods in Applied Mechanics and Engineering*, 197(21-24):1997–2014.

John, V. and Schmeyer, E. (2008). Finite element methods for time-dependent convection-diffusion-reaction equations with small diffusion. *Computer Methods in Applied Mechanics and Engineering*, 198(3-4):475 – 494.

Johnson, J. and Fastook, J. (2002). Northern hemisphere glaciation and its sensitivity to basal melt water. *Quaternary International*, 95(96):87–98.

Johnson, J., Prescott, P., and Hughes, T. (2004). Ice dynamics preceding catastrophic disintegration of the floating part of Jakobshavn Isbræ, Greenland. *Journal of Glaciology*, 50(171):492–504.

Johnson, J. and Staiger, J. (2007). Modeling long-term stability of the Ferrar Glacier, East Antarctica: Implications for interpreting cosmogenic nuclide inheritance. *Journal of Geophysical Research*, 112(F3):F03S30.

Joughin, I., Howat, I. M., Fahnestock, M., Smith, B., Krabill, W., Alley, R. B., Stern, H., , and Truffer, M. (2008). Continued evolution of Jakobshavn Isbrae following its rapid speedup. *Journal of Geophysical Research*, 113:F04006. doi:10.1029/2008JF001023.

Karniadakis, G. and Sherwin, S. (2005). *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press, Oxford.

Katz, R. (2008). Magma dynamics with the enthalpy method: Benchmark solutions and magmatic focusing at mid-ocean ridges. *Journal of Petrology*, 49(12):2099.

Kaushik, D., Gropp, W., Minkoff, M., and Smith, B. (2008). Improving the performance of tensor matrix vector multiplication in cumulative reaction probability based quantum chemistry codes. In *Proceedings of the 15th international conference on High performance computing*, pages 120–130. Springer-Verlag.

Kay, D., Loghin, D., and Wathen, A. (2002). A preconditioner for the steady-state Navier-Stokes equations. *SIAM Journal on Scientific Computing*, 24(1):237–256.

Kelley, C. (1995). *Iterative methods for linear and nonlinear equations*. Society for Industrial Mathematics.

Kelley, C. and Keyes, D. (1998). Convergence analysis of pseudo-transient continuation. *SIAM Journal on Numerical Analysis*, 35:508.

Ketcheson, D. (2008). Highly efficient strong stability preserving Runge-Kutta methods with low-storage implementations. *SIAM Journal on Scientific Computing*, 30(4):2113–2136.

Keyes, D. (2011). Exaflop/s: The why and the how. *Comptes Rendus Mécanique*, 339(2-3):70–77.

Kim, S. (2007). Piecewise bilinear preconditioning of high-order finite element methods. *Electronic Transactions on Numerical Analysis*, 26:228–242.

Kirby, R. (2004). Algorithm 839: FIAT, a new paradigm for computing finite element basis functions. *ACM Transactions on Mathematical Software (TOMS)*, 30(4):502–516.

Kirby, R. (2006). Optimizing FIAT with level 3 BLAS. *ACM Transactions on Mathematical Software (TOMS)*, 32(2):223–235.

Kirby, R., Knepley, M., Logg, A., and Scott, L. (2005). Optimizing the evaluation of finite element matrices. *SIAM Journal on Scientific Computing*, 27(3):741–758.

Kirk, B., Peterson, J. W., Stogner, R. H., and Carey, G. F. (2006). `libMesh`: A C++ library for parallel adaptive mesh refinement/coarsening simulations. *Engineering with Computers*, 22(3–4):237–254. `http://dx.doi.org/10.1007/s00366-006-0049-3`.

Klawonn, A. and Rheinbach, O. (2007a). Inexact FETI-DP methods. *International Journal for Numerical Methods in Engineering*, 69(2):284–307.

Klawonn, A. and Rheinbach, O. (2007b). Robust FETI-DP methods for heterogeneous three dimensional elasticity problems. *Computer Methods in Applied Mechanics and Engineering*, 196(8):1400–1414.

Klawonn, A. and Widlund, O. (2006). Dual-primal FETI methods for linear elasticity. *Communications on Pure and Applied Mathematics*, 59(11):1523–1572.

Knoll, D., Chacon, L., Margolin, L., and Mousseau, V. (2003). On balanced approximations for time integration of multiple time scale systems. *Journal of Computational Physics*, 185(2):583–611.

Knoll, D. and Keyes, D. (2004). Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397.

Knoll, D. and McHugh, P. (1998). Enhanced nonlinear iterative techniques applied to a nonequilibrium plasma flow. *SIAM Journal on Scientific Computing*, 19(1):291–301.

Knoll, D., Mousseau, V., Chacon, L., and Reisner, J. (2005). Jacobian-Free Newton-Krylov methods for the accurate time integration of stiff wave systems. *Journal of Scientific Computing*, 25(1):213–230.

Knupp, P. and Salari, K. (2002). *Verification of computer codes in computational science and engineering*. CRC Press.

Kondrat'ev, V. and Oleinik, O. (1988). Boundary-value problems for the system of elasticity theory in unbounded domains: Korn's inequalities. *Russian Mathematical Surveys*, 43:65.

Korelc, J. (1997). Automatic generation of finite-element code by simultaneous optimization of expressions. *Theoretical Computer Science*, 187(1-2):231–248.

Korelc, J. (2002). Multi-language and multi-environment generation of nonlinear finite element codes. *Engineering with Computers*, 18(4):312–327.

Kuzmin, D., Moller, M., and Turek, S. (2004). High-resolution FEM-FCT schemes for multidimensional conservation laws. *Computer Methods in Applied Mechanics and Engineering*, 193(45-47):4915–4946.

Larour, E., Morlighem, M., and Seroussi, H. (2010). Ice Sheet System Model. `issm.jpl.nasa.gov`.

Lemieux, J.-F., Price, S. F., Evans, K. J., Knoll, D., Salinger, A. G., Holland, D. M., and Payne, A. J. (2011). Implementation of the Jacobian-free Newton-Krylov method for solving the first-order ice sheet momentum balance. *Journal of Computational Physics*, 230(17):6531 – 6545.

Lemke, P., Ren, J., Alley, R., Allison, I., Carrasco, J., Flato, G., Fujii, Y., Kaser, G., Mote, P., Thomas, R., and Zhang, T. (2007). Observations: Changes in snow, ice and frozen ground. In Solomon, S., Qin, D., Manning, M., Chen, Z., Marquis, M., Averyt, K., Tignor, M., and Miller, H., editors, *Climate Change 2007: The Physical Science Basis. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*, volume WG I, chapter 4, pages 337–383. Cambridge University Press, Cambridge, UK.

LeVeque, R. (2002). *Finite volume methods for hyperbolic problems*. Cambridge Univ Press.

Li, J. and Widlund, O. (2006). BDDC algorithms for incompressible Stokes equations. *SIAM Journal on Numerical Analysis*, 44(6):2432–2455.

Limache, A., Sánchez, P., Dalcín, L., and Idelsohn, S. (2008). Objectivity tests for Navier-Stokes simulations: The revealing of non-physical solutions produced by Laplace formulations. *Computer Methods in Applied Mechanics and Engineering*, 197(49-50):4180–4192.

Lipscomb, G., Keunings, R., and Denn, M. (1987). Implications of boundary singularities in complex geometries. *Journal of Non-Newtonian Fluid Mechanics*, 24(1):85–96.

Liseikin, V. (2009). *Grid generation methods*. Springer Verlag.

Liu, T. (2000). *Hyperbolic and viscous conservation laws*. Number 72. Society for Industrial Mathematics.

Liu, X., Osher, S., and Chan, T. (1994). Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115(1):200–212.

Logg, A., Ølgaard, K., Rognes, M., Wells, G., Jansson, J., Kirby, R., Knepley, M., Lindbo, D., and Skavhaug, O. (2011). The FEniCS project. A continually updated technical report. `http://fenicsproject.org`.

Logg, A. and Wells, G. (2010). DOLFIN: Automated finite element computing. *ACM Transactions on Mathematical Software (TOMS)*, 37(2):1–28.

Lottes, J. and Fischer, P. (2005). Hybrid Multigrid/Schwarz Algorithms for the Spectral Element Method. *Journal of Scientific Computing*, 24(1):45–78.

Lynch, D. and Gray, W. (1980). Finite element simulation of flow in deforming regions. *Journal of Computational Physics*, 36(2):135–153.

MacAyeal, D. (1992). The basal stress distribution of Ice Stream E, Antarctica, inferred by control methods. *Journal of Geophysical Research*, 97:595–603.

MacAyeal, D. (1993). A tutorial on the use of control methods in ice-sheet modeling. *JGlac*, 39(131):91–98.

Maday, Y., Patera, A., and Rønquist, E. (1992). The $P_N - P_{N-2}$ method for the approximation of the Stokes problem. *Laboratoire d'Analyse Numérique, Paris VI*, 11(4).

Malas, T. M., Ahmadia, A. J., Brown, J., Gunnels, J. A., and Keyes, D. E. (2012). Optimizing the performance of streaming numerical kernels on the IBM BlueGene/P PowerPC 450 processor. *International Journal of High Performance Computing Applications*, accepted.

Mandel, J. and Dohrmann, C. (2003). Convergence of a balancing domain decomposition by constraints and energy minimization. *Numerical Linear Algebra with Applications*, 10(7):639–659.

Matthies, G., Skrzypacz, P., and Tobiska, L. (2008). Stabilization of local projection type applied to convection-diffusion problems with mixed boundary conditions. *Electronic Transactions on Numerical Analysis*, 32:90–105.

Matthies, G. and Tobiska, L. (2002). The inf-sup condition for the mapped $Q_k - P_{k-1}^{\text{disc}}$ element in arbitrary space dimensions. *Computing*, 69(2):119–139.

Matthies, H. and Keese, A. (2005). Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 194(12-16):1295–1331.

May, D. A. and Moresi, L. (2008). Preconditioned iterative methods for Stokes flow problems arising in computational geodynamics. *Physics of the Earth and Planetary Interiors*, 171(1-4):33 – 47. Recent Advances in Computational Geodynamics: Theory, Numerics and Applications.

McCalpin, J. D. (1991-2007). STREAM: Sustainable memory bandwidth in high performance computers. Technical report, University of Virginia, Charlottesville, Virginia. A continually updated technical report. `http://www.cs.virginia.edu/stream`.

Merlis, T. and Khatiwala, S. (2008). Fast dynamical spin-up of ocean general circulation models using Newton-Krylov methods. *Ocean Modelling*, 21(3-4):97–105.

Mishra, S., Schwab, C., and Šukys, J. (2011). Multi-level Monte Carlo finite volume methods for nonlinear systems of conservation laws in multi-dimensions. Technical Report 2011-02, ETH Zürich.

Mizukami, A. and Hughes, T. (1985). A Petrov-Galerkin finite element method for convection-dominated flows: An accurate upwinding technique for satisfying the maximum principle. *Computer Methods in Applied Mechanics and Engineering*, 50(2):181–193.

Mohammadi, S. (2008). *Extended finite element method*. Wiley Online Library.

Morland, L. (1987). Unconfined ice-shelf flow. In van der Veen, C. and Oerlemans, J., editors, *Dynamics of the West Antarctic ice sheet*, pages 99–116. Kluwer Academic Publishers.

Morlighem, M., Rignot, E., Seroussi, H., Larour, E., Dhia, H., and Aubry, D. (2010). Spatial patterns of basal drag inferred using control methods from a full-Stokes and simpler models for Pine Island Glacier, West Antarctica. *Geophysical Research Letters*, 37(14):L14502.

Motyka, R., Fahnestock, M., and Truffer, M. (2010). Volume change of Jakobshavn Isbrae, West Greenland:: 198519972007. *Journal of Glaciology*, 56(198):635–646.

Mousseau, V., Knoll, D., and Reisner, J. (2002). An implicit nonlinearly consistent method for the two-dimensional shallow-water equations with Coriolis force. *Monthly Weather Review*, 130(11):2611–2625.

Murphy, M., Golub, G., and Wathen, A. (2000). A note on preconditioning for indefinite linear systems. *SIAM Journal on Scientific Computing*, 21(6):1969–1972.

Nachtigal, N., Reddy, S., and Trefethen, L. (1992). How fast are nonsymmetric matrix iterations? *SIAM Journal on Matrix Analysis and Applications*, 13:778.

Nazarov, S. and Plamenevskiĭ, B. (1994). *Elliptic problems in domains with piecewise smooth boundaries*. Walter de Gruyter.

Oden, J., Demkowicz, L., Rachowicz, W., and Westermann, T. (1989). Toward a universal *hp* adaptive finite element strategy. II: A posteriori error estimation. *Comput. Methods Appl. Mech. Engrg*, 77:113–180.

Olshanskii, M., Peters, J., and Reusken, A. (2006). Uniform preconditioners for a parameter dependent saddle point problem with application to generalized Stokes interface equations. *Numerische Mathematik*, 105(1):159–191.

Olshanskii, M. and Reusken, A. (2006). Analysis of a Stokes interface problem. *Numerische Mathematik*, 103(1):129–149.

Orszag, S. (1980). Spectral methods for problems in complex geometries. *J. Comput. Phys.*, 37:70–92.

Ouazzi, A. and Turek, S. (2006). Efficient multigrid and data structures for edge-oriented FEM stabilization. *Numerical Mathematics and Advanced Applications*, pages 520–527.

Owens, R. and Phillips, T. (2002). *Computational Rheology*. Imperial College Press.

Papanastasiou, T., Malamataris, N., and Ellwood, K. (1992). A new outflow boundary condition. *International Journal for Numerical Methods in Fluids*, 14(5):587–608.

Park, H., Nourgaliev, R., Martineau, R., and Knoll, D. (2009). On physics-based preconditioning of the Navier-Stokes equations. *Journal of Computational Physics*, 228(24):9131–9146.

Patankar, S. and Spalding, D. (1972). A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *Int. J. Heat Mass Transfer*, 15:1787–1806.

Pattyn, F. (2002). Transient glacier response with a higher-order numerical ice-flow model. *Journal of Glaciology*, 48(162):467–477.

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(2382):10–1029.

Pattyn, F., Perichon, L., Aschwanden, A., Breuer, B., De Smedt, B., Gagliardini, O., Gudmundsson, G., Hindmarsh, R., Hubbard, A., Johnson, J., Kleiner, T., Konovalov, Y., Martin, C., Payne, A., Pollard, D., Price, S., Rückamp, M., Saito, F., Souček, O., Sugiyama, S., and Zwinger, T. (2008). Benchmark experiments for higher-order and full Stokes ice sheet models (ISMIP-HOM). *The Cryosphere*, 2(1):95–108.

Pham, D. C., Aipperspach, T., Boerstler, D., Bolliger, M., Chaudhry, R., Cox, D., Harvey, P., Harvey, P. M., Hofstee, H. P., Johns, C., Kahle, J., Kameyama, A., Keaty, J., Masubuchi, Y., Pham, M., Pille, J., Posluszn, S., and Yazawa, K. (2006). Overview of the architecture, circuit design, and physical implementation of a first-generation cell processor. *Solid-State Circuits, IEEE Journal of*, 41(1):179–196.

Pimentel, S. and Flowers, G. (2011). A numerical study of hydrologically driven glacier dynamics and subglacial flooding. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 467(2126):537.

Pimentel, S., Flowers, G., and Schoof, C. (2010). A hydrologically coupled higher-order flow-band model of ice dynamics with a coulomb friction sliding law. *Journal of Geophysical Research*, 115(F4):F04023.

Plummer, J. et al. (*submitted*, 2011). A high-resolution bed elevation map for Jakobshavn Isbræ, west greenland. *Journal of Glaciology*.

Podhaisky, H. (2006). Atlas of general linear methods with inherent Runge-Kutta stability. `http://www.math.auckland.ac.nz/~hpod/atlas/`.

Rachowicz, W., Oden, J., and Demkowicz, L. (1989). Toward a universal *hp* adaptive finite element strategy. III: Design of *hp* meshes. *Comput. Methods Appl. Mech. Engrg*, 77:181–212.

Rannacher, R. (2000). Finite element methods for the incompressible Navier-Stokes equations. *Fundamental Directions in Mathematical Fluid Mechanics*, page 191.

Rannacher, R. and Turek, S. (1992). Simple nonconforming quadrilateral Stokes element. *Numerical Methods for Partial Differential Equations*, 8(2):97–111.

Raymond, C. (2000). Energy balance of ice streams. *Journal of Glaciology*, 46(155):665–674.

Raymond, M. and Gudmundsson, G. (2009). Estimating basal properties of ice streams from surface measurements: a non-linear Bayesian inverse approach applied to synthetic data. *The Cryosphere*, 3:265–278.

Rees, T., Dollar, H., and Wathen, A. (2010). Optimal solvers for PDE-constrained optimization. *SIAM Journal on Scientific Computing*, 32:271.

Renardy, M. (1997). Imposing "no" boundary condition at outflow: Why does it work? *International Journal for Numerical Methods in Fluids*, 24(4):413–417.

Rice, J. (1968). A path independent integral and the approximate analysis of strain concentration by notches and cracks. *Journal of Applied Mechanics*, 35(2):379–386.

Rice, J. and Rosengren, G. (1968). Plane strain deformation near a crack tip in a power-law hardening material. *Journal of the Mechanics and Physics of Solids*, 16(1):1–12.

Roache, P. (1998). Verification and validation in computational science and engineering. *Computing in Science Engineering*, pages 8–9.

Roache, P. (2002). Code verification by the method of manufactured solutions. *Journal of Fluids Engineering*, 124:4.

Roache, P. (2004). Building PDE codes to be verifiable and validatable. *Computing in Science & Engineering*, 6(5):30–38.

Roache, P., Ghia, K., and White, F. (1986). Editorial policy statement on the control of numerical accuracy. *Journal of Fluids Engineering*, 108:2.

Saad, Y. (1993). A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing*, 14:461–461.

Saad, Y. and Schultz, M. (1986). GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869.

Saito, N. (2004). On the Stokes equations with the leak and slip boundary conditions of friction type: regularity of solutions. *Publ. Res. Inst. Math. Sci*, 40:345–383.

Sandu, A. (2006). On the properties of runge-kutta discrete adjoints. *Computational Science–ICCS 2006*, pages 550–557.

Sandu, A., Daescu, D., and Carmichael, G. (2003). Direct and adjoint sensitivity analysis of chemical kinetic systems with KPP: Part I–theory and software tools. *Atmospheric Environment*, 37(36):5083–5096.

Schieweck, F. (2008). Uniformly stable mixed *hp*-finite elements on multilevel adaptive grids with hanging nodes. *Modélisation mathématique et analyse numérique*, 42(3):493–505.

Schoof, C. (2006a). A variational approach to ice stream flow. *Journal of Fluid Mechanics*, 556:227–251.

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

Schoof, C. (2007). Ice sheet grounding line dynamics: Steady states, stability, and hysteresis. *Journal of Geophysical Research*, 112(F3):F03S28.

Schoof, C. (2010). Ice-sheet acceleration driven by melt supply variability. *Nature*, 468(7325):803–806.

Schoof, C. and Hindmarsh, R. (2010). Thin-film flows with wall slip: an asymptotic analysis of higher order glacier flow models. *The Quarterly Journal of Mechanics and Applied Mathematics*, 63:73–114.

Schoof, C., Hindmarsh, R., and Pattyn, F. (2009). MISMIP: Marine ice sheet model intercomparison project. *in preparation*.

Schötzau, D., Schwab, C., and Stenberg, R. (1998). Mixed *hp*-FEM on anisotropic meshes. *Mathematical Models and Methods in Applied Sciences*, 8:787–820.

Schötzau, D., Schwab, C., and Stenberg, R. (1999). Mixed *hp*-FEM on anisotropic meshes II: Hanging nodes and tensor products of boundary layer meshes. *Numerische Mathematik*, 83(4):667–697.

Schroeder, W., Bertel, F., Malaterre, M., Thompson, D., Pebay, P., O'Barall, R., and Tendulkar, S. (2005). Framework for visualizing higher-order basis functions. In *Visualization, 2005. VIS 05. IEEE*, pages 43–50. IEEE.

Schroeder, W. et al. (2006). Methods and framework for visualizing higher-order finite elements. *IEEE transactions on visualization and computer graphics*, pages 446–460.

Schroeder, W., Martin, K., Martin, K., and Lorensen, B. (1998). *The Visualization Toolkit*. Prentice Hall PTR.

Schwab, C. (1998). *p-and hp-Finite Element Methods: Theory and Applications in Solid and Fluid Mechanics*. Oxford University Press.

Sederberg, T., Zheng, J., Bakenov, A., and Nasri, A. (2003). T-splines and T-NURCCs. In *ACM SIGGRAPH 2003 Papers*, pages 477–484. ACM.

Seiler, L., Carmean, D., Sprangle, E., Forsyth, T., Abrash, M., Dubey, P., Junkins, S., Lake, A., Sugerman, J., Cavin, R., et al. (2008). Larrabee: a many-core x86 architecture for visual computing. *ACM Transactions on Graphics (TOG)*, 27(3):1–15.

Shamsundar, N. and Sparrow, E. (1975). Analysis of multidimensional conduction phase change via the enthalpy model. *J. Heat Transfer*, 97(3).

Shin, J., Hall, M., Chame, J., Chen, C., Fischer, P., and Hovland, P. (2010). Speeding up Nek5000 with autotuning and specialization. In *Proceedings of the 24th ACM International Conference on Supercomputing*, pages 253–262. ACM.

Shu, C. (2003). High-order finite difference and finite volume WENO schemes and discontinuous Galerkin methods for CFD. *International Journal of Computational Fluid Dynamics*, 17(2):107–118.

Silvester, D., Elman, H., Kay, D., and Wathen, A. (2001). Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow. *Journal of Computational and Applied Mathematics*, 128(1-2):261–279.

Simon, H., Zacharia, T., Stevens, R., et al. (2007). Modeling and Simulation at the Exascale for Energy and the Environment. Technical report, Department of Energy.

Smith, B., Bjørstad, P., and Gropp, W. (1996). *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*. Cambridge Univ Press, New York.

Smith, B. and Zhang, H. (2010). Sparse triangular solves for ILU revisited: data layout crucial to better performance. *International Journal of High Performance Computing Applications*, pages 1–6.

Spijker, M. (1983). Contractivity in the numerical solution of initial value problems. *Numerische Mathematik*, 42(3):271–290.

Stenberg, R. and Suri, M. (1996). Mixed finite element methods for problems in elasticity and Stokes flow. *Numerische Mathematik*, 72(3):367–389.

Tarantola, A. (2005). *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial Mathematics.

Tautges, T. J. (2001). CGM: a geometry interface for mesh generation, analysis and other applications. *Engineering with Computers*, 17(3):299–314. Journal Article.

Tautges, T. J. (2011). MeshKit. `https://trac.mcs.anl.gov/projects/fathom/wiki/MeshKit`.

Tautges, T. J., Meyers, R., Merkley, K., Stimpson, C., and Ernst, C. (2004). MOAB: a Mesh-Oriented database. Technical report, Sandia National Laboratories.

Tomkin, J. and Braun, J. (2002). The influence of alpine glaciation on the relief of tectonically active mountain belts. *American Journal of Science*, 302(3):169.

Toro, E. (2009). *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Verlag.

Toselli, A. and Schwab, C. (2003). Mixed *hp*-finite element approximations on geometric edge and boundary layer meshes in three dimensions. *Numerische Mathematik*, 94(4):771–801.

Trefethen, L. and Embree, M. (2005). *Spectra and pseudospectra: the behavior of nonnormal matrices and operators*. Princeton Univ Press.

Trottenberg, U., Oosterlee, C., and Schüller, A. (2001). *Multigrid*. Academic Press.

Truffer, M. and Echelmeyer, K. (2003). Of Isbræ and Ice Streams. *Annals of Glaciology*, 36(1):66–72.

Tufo, H. and Fischer, P. (2001). Fast parallel direct solvers for coarse grid problems. *Journal of Parallel and Distributed Computing*, 61(2):151–177.

Turek, S. and Ouazzi, A. (2007). Unified edge-oriented stabilization of nonconforming FEM for incompressible flow problems: Numerical investigations. *Journal of Numerical Mathematics*, 15(4):299–322.

Turek, S., Ouazzi, A., and Schmachtel, R. (2002). Multigrid methods for stabilized nonconforming finite elements for incompressible flow involving the deformation tensor formulation. *Journal of Numerical Mathematics*, 10(3):235–248.

Üffinger, M., Frey, S., and Ertl, T. (2010). Interactive high-quality visualization of higher-order finite elements. *Computer Graphics Forum*, 29(2):337–346.

Utke, J. (2004). Openad: Algorithm implementation user guide. Technical report, Technical Memorandum ANL/MCS–TM–274, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill.

Utke, J., Naumann, U., Fagan, M., Tallent, N., Strout, M., Heimbach, P., Hill, C., and Wunsch, C. (2008). Openad/f: A modular open-source tool for automatic differentiation of fortran codes. *ACM Transactions on Mathematical Software (TOMS)*, 34(4):1–36.

Utke, J., Naumann, U., and Lyons, A. (2011). OpenAD/F: User Manual. Technical report, Argonne National Laboratory. latest version available online at `http://www.mcs.anl.gov/OpenAD/openad.pdf`.

Van der Vorst, H. and Vuik, C. (1994). GMRESR: A family of nested GMRES methods. *Numerical Linear Algebra with Applications*, 1(4):369–386.

Vanka, S. (1986). Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. *Journal of Computational Physics*, 65(1):138–158.

Vuik, C. (1995). New insights in gmres-like methods with variable preconditioners. *Journal of Computational and Applied Mathematics*, 61(2):189–204.

Walkley, M., Gaskell, P., Jimack, P., Kelmanson, M., Summers, J., and Wilson, M. (2004). On the calculation of normals in free-surface flow problems. *Communications in Numerical Methods in Engineering*, 20(5):343–351.

Walter, S. (2011). AlgoPy – algorithmic differentiation in Python.

Walter, S. and Lehmann, L. (2010). Algorithmic differentiation of linear algebra functions with application in optimum experimental design (extended version). *Arxiv preprint arXiv:1001.1654*.

Wang, P. (1986). FINGER: A symbolic system for automatic generation of numerical programs in finite element analysis. *Journal of Symbolic Computation*, 2(3):305–316.

Washington, W., Bader, D., Collins, B., Drake, J., Taylor, M., Kirtman, B., Williams, D., and Middleton, D. (2009). Scientific grand challenges: Challenges in climate change science and the role of computing at the extreme scale. Technical report, Department of Energy, Office of Biological and Environmental Research and the Office of Advanced Scientific Computing Research.

Weertman, J. (1957). On the sliding of glaciers. *Journal of Glaciology*, 3:33–38.

Weertman, J. (1974). Stability of the junction of an ice sheet and an ice shelf. *Journal of Glaciology*, 13(67):3–11.

Weis, M., Greve, R., and Hutter, K. (1999). Theory of shallow ice shelves. *Continuum Mechanics and Thermodynamics*, 11(1):15–50.

White, R. (1982). An enthalpy formulation of the Stefan problem. *SIAM J. Numer. Anal*, 19(6).

Williams, S., Oliker, L., Vuduc, R., Shalf, J., Yelick, K., and Demmel, J. (2007). Optimization of sparse matrix-vector multiplication on emerging multicore platforms. In *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*. ACM New York, NY, USA.

Wriggers, P. (2008). *Nonlinear finite element methods*. Springer Verlag.

Wright, W. (2002). *General linear methods with inherent Runge-Kutta stability*. PhD thesis, University of Auckland.

Zabaras, N. and Ganapathysubramanian, B. (2008). A scalable framework for the solution of stochastic inverse problems using a sparse grid collocation approach. *Journal of Computational Physics*, 227(9):4697–4735.

Zalesak, S. (1979). Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of Computational Physics*, 31(3):335–362.

Curriculum Vitae

Contact

Name	Jedediah Aspen Kallen-Brown
Address	Box 84056
	Fairbanks, AK 99708, USA
Email	jed@59A2.org
Date of birth	4 June, 1983, in Alaska
Citizenship	United States

Education

2007 - 2011	Versuchsanstalt für Wasserbau, Hydrologie und Glaziologie (VAW), ETH Zürich: PhD-thesis “Computational methods for ice flow simulation with applications to Jakobshavn Isbræ”
2004 - 2006	University of Alaska Fairbanks: M.S. in Mathematics
2000 - 2004	University of Alaska Fairbanks: B.S. in Mathematics B.S. in Physics