The Dissertation Committee for James Michael Rath
certifies that this is the approved version of the following dissertation:

Multiscale Basis Optimization for Darcy Flow

Committee:

Todd Arbogast, Supervisor

Steve Bryant

Clint Dawson

Robert van de Geijn

Mary Wheeler

Multiscale Basis Optimization for Darcy Flow

by

James Michael Rath, B.S., M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2007

For everyone who
helped me along the way,
Thanks Y'all.

Acknowledgments

This work was supported in part by grant DMS-0408489 from the National Science Foundation. The University of Texas at Austin has generously supported me during my time here.

James Michael Rath

The University of Texas at Austin
May 2007

Multiscale Basis Optimization for Darcy Flow

Publication No. _____

James Michael Rath, Ph.D.
The University of Texas at Austin, 2007

Supervisor: Todd Arbogast

Simulation of flow through a heterogeneous porous medium with fine-scale features can be computationally expensive if the flow is fully resolved. Coarsening the problem gives a faster approximation of the flow but loses some detail. We propose an algorithm that obtains the fully resolved approximation but only iterates on a sequence of coarsened problems. The sequence is chosen by optimizing the shapes of the coarse finite element basis functions.

As a stand-alone method, the algorithm converges globally and monotonically with a quadratic asymptotic rate. Computational experience indicates the number of iterations needed is independent of the resolution and heterogeneity of the medium. However, an externally provided error estimate is required; the algorithm could be combined as an accelerator with another iterative algorithm. A single "inner" iteration of the other algorithm would yield an error estimate; following it with an "outer" iteration of our algorithm would give a viable method.

Contents

List of Figures

List of Symbols

Ω	domain for the PDE	7
V	space of trial functions (piecewise linear polynomials on simplices)	7
k	coefficient of the PDE (proportional to permeability)	7
A	operator/matrix representing $\nabla \cdot k\nabla$	7
f	source, gravity, and boundary data	7
u	solution of the problem $Au = f$	7
δV	subgrid bubble functions/degrees of freedom	9
Γ_N	subset of the boundary $\partial\Omega$ with Neumann boundary conditions	9
V_{corner}	corner shape functions/degrees of freedom	9
e_c	a coarse edge (or face)	9
β_{e_c}	list/vector of shape parameters along a coarse edge e_c	10
$v_{\beta_{e_c}}$	shape function with support on coarse patches abutting e_c with trace β_{e_c} along e_c	10
V_{e_c}	the linear map $V_{e_c} : \beta_{e_c} \to v_{\beta_{e_c}}$ that takes edge shape parameters β_{e_c} to edge shape functions $v_{\beta_{e_c}}$ in V; also, the range of this map	10
$V_{\beta,\text{edge}}$	edge bubble functions/degrees of freedom	10
$V_{\beta,\text{face}}$	face bubble functions/degrees of freedom	10
β	list/vector of shape parameters on all edges/faces	10
V_β	$\delta V \oplus V_{\beta,\text{face}} \oplus V_{\beta,\text{edge}} \oplus V_{\text{corner}}$; multiscale shape functions/degrees of freedom	10
V_0	$\delta V \oplus V_{\text{corner}}$ or V_β with $\beta = 0$	10
V_{edge}	union over all β of $V_{\beta,\text{edge}}$; edge degrees of freedom from V	10
V_{face}	union over all β of $V_{\beta,\text{face}}$; face degrees of freedom from V	10
P_e	projection that extracts edge/face information from a function in V	10
I_β	the natural inclusion from V_β to V	10
A_β	$I_\beta^T A I_\beta$ or A applied to the subspace V_β	12
f_β	$I_\beta^T f$ or f restricted to the subspace V_β^*	12
u_β	multiscale solution of the problem $A_\beta u_\beta = f_\beta$	7
u_0	multiscale solution with $\beta = 0$	31

Z_β	A-orthogonal projection from V onto $V_\beta^{\perp_A}$	12
e_β	error in the multiscale solution $e_\beta = u - I_\beta u_\beta$	12
r_β	residual of the multiscale solution $r_\beta = f - AI_\beta u_\beta$	14
$\|\cdot\|_A$	energy norm; $\|v\|_A = \left(v^T A v\right)^{1/2}$ for $v \in V$	14
$\dagger$	Moore-Penrose pseudoinverse	
$\dagger_A$	pseudoinverse where the range is measured in the A-inner product	31
β^*	set of shape parameters β for which $e_{\beta^*} = 0$	18
$F_M(\beta)$	level set functions measuring size of β; objective functions for Newton minimization	20
e'_β, r'_β	Jacobians of the error and residual functions $\frac{\partial e_\beta}{\partial \beta}$ and $\frac{\partial r_\beta}{\partial \beta}$	21
ω	damping factor for a (Newton) step	20
h	maximum diameter of a fine-scale element	7
H	maximum diameter of a coarse-scale element	9
$\mathcal{E}$	the collection of all error vectors: $\mathcal{E} = \{e_\beta \; \forall \beta\}$	27
P_{e_c}	linear operator that takes a vector $v \in V$, extracts edge information $P_e v$, restricts attention to a given coarse edge $(P_e v)\big\|_{e_c}$, and extends the result by zero to the rest of the degrees of freedom	27
$V_{\tilde{\beta}}$	a reparameterization of V_β where each shape parameter's influence is orthogonal to all others'	36
$\tilde{P}_{e_c}$	an orthogonal edge-information extraction operator that works on $V_{\tilde{\beta}}$	37
$V_{\hat{\beta}}$	a reparameterization of V_β where each shape parameter's influence is A-orthogonal to all others'	36
$\hat{Q}_{e_c}$	an A-orthogonal edge-information extraction operator that works on $V_{\hat{\beta}}$, a reparameterization of V_β	37
$P_{\tan}$	(A-)orthogonal projection onto the tangent space of a manifold	28

Chapter 1

Overview

Solving linear systems involves nonlinear operations, namely, division. At first blush, one might expect that iterative algorithms for solving linear systems can achieve superlinear convergence since Newton's method does for nonlinear ones. However, only a handful of algorithms for linear problems have this property.

In this dissertation we describe a new method for solving second order elliptic partial differential equations such as Darcy flow problems. The algorithm optimizes basis shapes used in solving a coarsened version of the problem; the process ends when the solution to the coarse problem coincides with the original problem (or nearly so).

Since we optimize the basis (and not the solution directly), we trade solving a linear system for solving a nonlinear optimization problem. Intuition may tell us that a nonlinear problem is more difficult to solve. However, we do this because we trade a large linear system for a small nonlinear one, and this nonlinear problem has a special structure we can further exploit.

Our analysis shows we can achieve global, monotone, asymptotically quadratic convergence with a cheap per-iteration cost. Some computational experience has also shown that the number of iterations needed is independent of both the resolution of the flow problem and the heterogeneity of the permeability field (that is, independent of the condition number of the problem). However, we assume an externally provided error estimate is available at each step. Our algorithm would be effective as an accelerator: an inner iteration of another iterative method would provide an error estimate after which an outer iteration of our method would act on that estimate.

1.1 Motivating problem

Most parts of our proposed algorithms can be described in a purely algebraic fashion and can be applied to any symmetric positive definite linear system. As a black-box solver, our algorithms may do quite well. However, our original motivating problem was solving Darcy flow problems where we face two challenges — high resolution and heterogeneous data — that make the use of conventional solvers impractical.

Darcy's law describes fluid flow through a porous medium. It is an empirical law that asserts bulk flow of a fluid through the medium is proportional to the gradient of the pressure across the medium (accounting for hydrostatic differences from gravity) [21, 8, 74, 35]:

$$\mathbf{u} = -\frac{\boldsymbol{\kappa}}{\mu} \left(\nabla p - \rho \mathbf{g} \right).$$

Darcy's law has found wide applicability in modeling subsurface flows, and has been generalized to model multicomponent and multiphase flows. (The above differential form is itself a generalization of the relation Darcy formulated.) Our primary interest is in using Darcy's law to model oil reservoir and groundwater contaminant flows.

Darcy's law alone is insufficient to describe the physics: conservation of mass (the continuity equation) and equations of state (relating density, viscosity, and permeability to phase fraction and temperature) are necessary. In the applications being considered, there is often a need for the velocity to be very accurate and to strictly (locally) observe mass conservation. For simplicity, our presentation describes only Galerkin methods; these do not produce conservative flow fields. However, there are post-processing methods that obtain conservative velocity fields; see [18, 77], for instance. Mixed methods also produce conservative flows [71, 29, 22, 32]; for this reason, we originally focussed on mixed methods. We have done substantial work to develop our algorithms for mixed elements, and intend to publish these shortly. Also, although our presentation ignores aspects of multiphase flow, the proposed ideas and software can readily be adapted to model such flows.

The challenge of heterogeneity

Geostatistical modeling is used to generate the necessary data (porosity and permeability) to specify the problem to be approximated [28]. This data is typically given at a very fine resolution [30], but the goal is to predict long-range flow behavior (such as break-through times, optimal pumping and injection rates, and total volumes produced). It is tempting, then, to approximate the problem at a very coarse scale. However, nature is not so kind and fine-scale features of the problem data can have substantial effects on the coarse-scale flow behavior [30, 1].

Therein lies one big difficulty: it is necessary to resolve the flow at very fine scales requiring the solution of computationally ill-conditioned problems [37]. Moreover, the resolution cannot be reduced to shrink the size of the system: (1) heterogeneity in the permeability (irregular, short spatial-scale jumps) means p-refinements (high-order approximations) will not help, and (2) spatially-limited resolution and spatially-uniform heterogeneity means h-refinements (coarse scaling) will not help either. Further, geometrically irregular features prevent the use of more specialized solvers. For instance, if we had a well-defined layer structure, we could use deflation coupled with any usual iterative solver to achieve fast convergence [81, 34, 4]. However, real geologic formations are not always so neat; see Figures 1.1 and 1.2 for two examples.

The fine-scale resolution necessary in simulations makes for poor conditioning, yet there is still another difficulty: the jumps in the permeability can sometimes be quite severe (spanning several orders of magnitude) between nearby locations. (Such contrasts can easily be seen in the seismics in Figure 1.1; the scale of such contrasts are numerically detailed in the simulated permeabilities in Figures 1.2, 5.1, and 5.5.) This makes our computation of an approximation even more poorly conditioned.

To restate, the more heterogeneous and fine-scale the problem, the higher the condition number and the more computationally expensive it becomes to solve our flow problem. All direct/iterative linear solvers in common use for this problem have behavior which worsens with increasing condition number [38, 72]. As a means of circumventing this computational conundrum, *upscaling* techniques have been developed that perform computations on a coarser scale (for faster computations) but still retain information about the fine-scale flow and problem data (for accurate flow predictions) [30]. In upscaling the problem, some information is always lost; an averaging procedure is used in upscaling to determine the influence of fine-scales on the coarse-scale problem. However, the result is often of such good quality that it seems appropriate to use it as a starting point for the full fine-scale computation; this idea we develop here.

Proposed solution technique

We use one kind of upscaling — variational multiscale subgrid upscaling [46, 5] — to construct an accelerator for solving the full fine-scale problem. Through this we hope to broaden the range of practical-interest problems that are computationally feasible. That is, our solver appears to be perform well with high-resolution data and is insensitive to geometric irregularities and high contrasts in the data.

In upscaling, the flexibility of (or number of degrees of freedom in) our approximation is reduced in order to obtain a smaller or better algebraic problem to solve. Our proposed algorithm attempts to reintroduce the necessary flexibility into the upscaled model to be

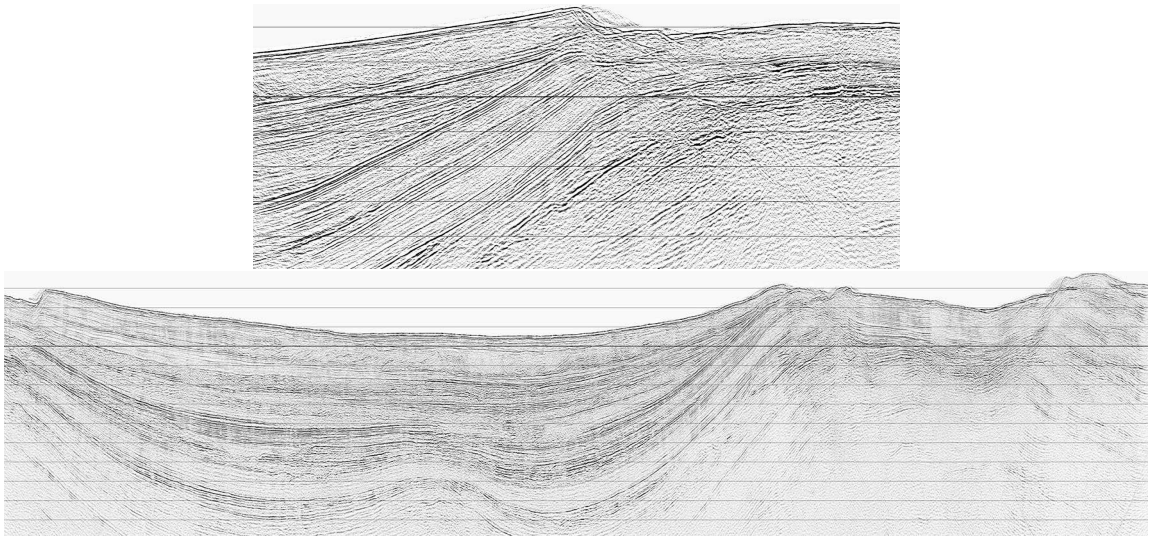

Figure 1.1: Sample seismics from the Keathley Canyon in the northern Gulf of Mexico (adapted from [47, 41]). This is an area of increasing gas and oil exploration; the USGS survey from which the data is borrowed was intended to help characterize the nature of gas hydrates present on the ocean floor and its near subsurface. (These hydrates are a potential energy source as well as a hazard to drilling.) The region's geology is driven by salt tectonics. We show these data here because they "illustrate a rich pattern of unconformities, pinch-outs, on-laps, and faults between the basin center and structural high at the edge of the basin." That is, subsurface features can have great geometric irregularties and high material contrasts on short and long scales.

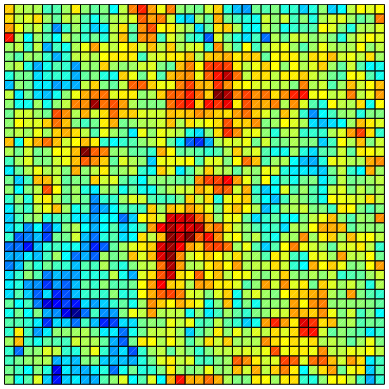

Figure 1.2: A typical sample of a geostatistically generated permeability field (from [57]). The covariance is homogeneous and isotropic with a power law structure ($\beta = 1/2$). The base-10 logarithm is plotted. Red indicates high permeability; the greatest is 3330 mD. Blue indicates low permeability; the smallest is 0.712 mD. The permeabilities span about three and a half orders of magnitude.

able to capture the fine-scale solution. We do this by considering a parameterized family of macro-elements for the coarse-scale shape functions; the family includes all possible fine-scale shapes. A sequence of upscaled problems is solved in which the problem structure (through the shape parameters, not the problem data) is gradually evolved towards a problem which has a solution that coincides with the fine-scale solution. The sequence is chosen using nonlinear optimization techniques — Newton's method and its ilk. These have the potential of superlinear convergence. Each step in the optimization only requires solving an upscaled problem, determining its fine-scale residual, and calculating a simple projection of the residual.

Each of these operations we assume to be inexpensive. The upscaled problem is effectively as expensive to compute as a coarse problem — a much smaller system than the original. Calculating the residual requires only a (sparse) matrix-vector multiply and a vector-vector addition. The required projection is an orthogonal one onto a low-dimensional (or one-dimensional) subspace, and can be calculated by solving a small linear system and a few vector-vector operations (or a dot product and "axpy" in the one-dimensional case).

Provided with a sufficiently accurate error estimate at each step, we can prove that our algorithm converges globally and has an asymptotic quadratic convergence rate. We introduce some approximations to make the algorithm more practical but as of yet are unsure of their effect on the global convergence. With regards to an error estimate, we conjecture that a simple smoother (such as Jacobi or Gauss–Seidel) would be sufficient to make for a viable method, but we are unsure of the effect of an imperfect error estimate.

1.2 Outline

In Chapter 2, we describe the application of lowest-order Galerkin elements to Darcy flow, and give a description of how variational multiscale subgrid upscaling is used to coarsen the resulting system. This is included to introduce some terminology and notation, and to remind the reader of how this method works. It further introduces our variant with macroscale coarse shape functions.

In Chapter 3, we introduce an algorithm that demonstrates our nonlinear approach. Although this algorithm is probably not practical, it lays the way for a different, geometry-based approach in Chapter 4. We cannot prove much about the first approach but can say much more about the geometric one. In both chapters, we introduce some modifications to make the algorithms computationally feasible. Some changes are exact; others are approximate but provably do not affect performance. However, there are yet other approximations that seem reasonable, but we cannot prove they do not have undesirable effects.

Chapter 5 demonstrates our algorithms on problems of practical interest in simulat-

ing Darcy flow. Evidence is given for the proven and conjectured properties laid out in the previous chapters.

In Chapter 6, we compare our algorithms with other similarly-featured algorithms. There are algorithms which share our small computational complexity, our superlinear convergence rate, or our insensitivity to heterogeneity, but none share all these properties (or even any one in quite the same way).

Finally, in Chapter 7, we make some comments about the algorithms and lay out some further research directions.

A table of symbols can be found for easy reference just after the table of contents and list of figures. A bibliography follows the last chapter.

Chapter 2

Problem description

In this chapter we describe the use of lowest-order Galerkin elements to discretize the Darcy flow problem, and how we coarsen this discretization using a specially modified multiscale basis.

2.1 Underlying fine-scale problem

Suppose we have a convex polygonal domain Ω with a fine mesh of triangles or tetrahedra (with maximum diameter h). Let V be the set of piecewise linear finite elements on the fine mesh. A diagram of the degrees of freedom of V is shown for a sample domain in Figure 2.1.

Suppose k is a symmetric, uniformly positive definite matrix.[1] Let $A\colon V \to V^*$ be the operator defined by $\langle Au, v\rangle = (k\nabla u, \nabla v)_{L^2}$ for all $u, v \in V$, and let $f \in V^*$ represent source, gravity, and boundary data.[2] We will refer to k as the "permeability", but the k in our equation is only proportional to permeability; other factors (such as viscosity) affect this term. We have changed another symbol as well: u here is the "unknown"; it corresponds to pressure.

Our goal is to solve the problem $Au = f$. As mentioned in the introduction, this problem is challenging because the permeability k comes to us at high resolution (small h), and because the permeability has high contrasts on short scales (the ratio of the maximum permeability $k_{\max}$ to minimum permeabiltiy $k_{\min}$ is large). The high resolution and contrast result in an ill-conditioned A: the condition number of A grows as $O(k_{\max}/k_{\min}\, h^{-2})$. The problem cannot simply be coarsened as details on all scales are important to correct flow

[1] We conjecture that the permeability k need follow a sort of super diagonal dominance for the results on insensitivity to jump size (a diagonal k will do, for instance), but such diagonal dominance is demonstrably unnecessary for the convergence results.

[2] Discretizing a time-dependent Darcy flow problem where the medium is slightly compressible results in an additional term $(cu, v)_{L^2}$ to the bilinear form defining A. The coefficient c is non-negative; the term simply adds to the diagonal of A and otherwise does not affect our analysis.

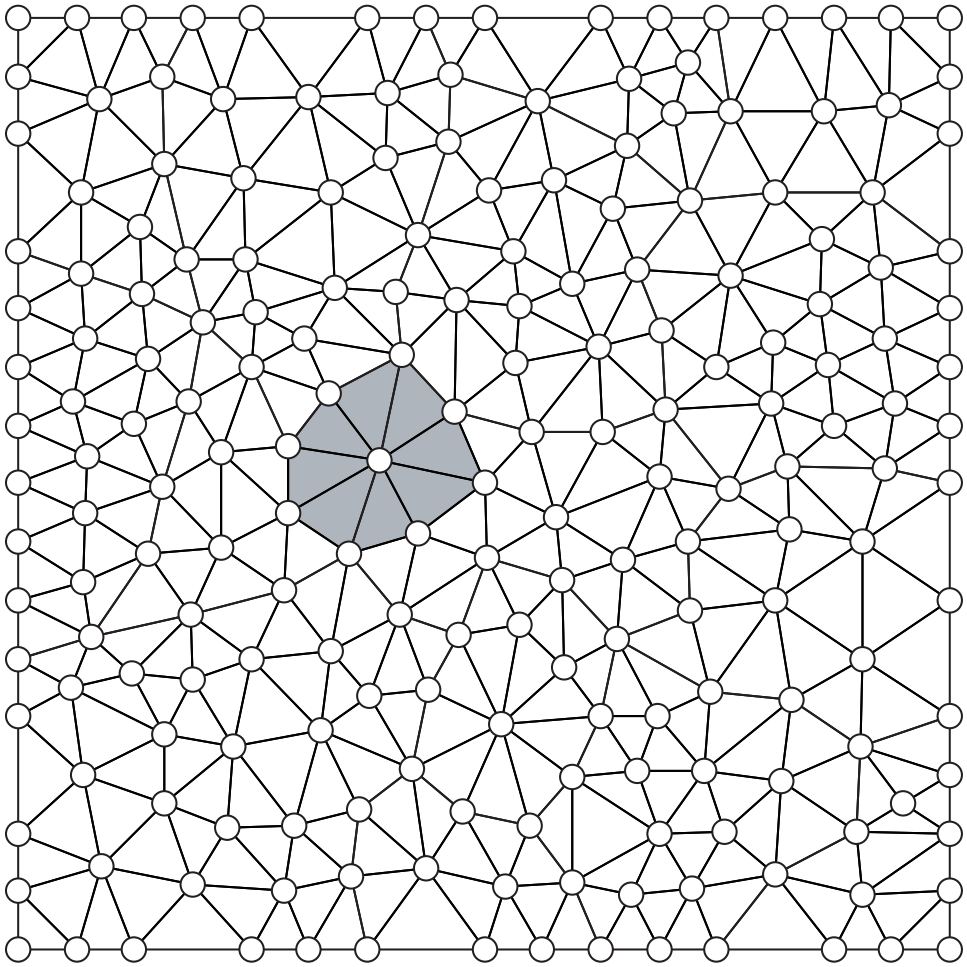

Figure 2.1: A sample domain with multiscale degrees of freedom shown. Each open circle represents a degree of freedom of V. The gray patch indicates the support for the degree of freedom at its center. For simplicity, the domain is square, and the case $\Gamma_N = \Gamma$ is shown so that there are degrees of freedom all along the boundary of the domain.

prediction or, rather, prediction of quantities that depend on the flow such as injection and production rates, breakthrough times of injected fluids, and total production. See, for instance, the introductions in [30, 1].

2.2 Coarsening the problem

We assume that a coarsened mesh is given to us.[3] We require, though, that the coarse mesh is formed by picking coarse elements as patches of fine elements, that is, by agglomerating fine elements into coarse ones (with maximum diameter H). Some rules governing this agglomeration are necessary to make our proposed algorithms computationally feasible; for instance, picking simply connected coarse patches would be wise. However, the following analysis shows no such rules are necessary.

The space V has a degree of freedom at each vertex of the mesh. We will segregate these degrees of freedom into groups, and construct from these groups a multiscale basis whose span approximates V. In defining this basis we follow the lead of the of the variational multiscale finite element method. Notable differences include the use of edge and face bubbles in the coarse basis (and not just corners), and the parameterization of the edge and face bubbles as macro elements. (To step ahead for a moment, see Figure 2.2 for a 2-D example.)

Let δV be the subspace of V defined by the span of shape functions whose support lies in a single coarse patch. This space δV might be described as subgrid bubbles or interior bubbles.

Suppose the intersection (in 2-D) of three or more coarse patches is a single vertex; call that vertex a corner. Also consider as corners those vertices that are the intersection of two or more coarse patches and Γ_N, the part of $\partial\Omega$ on which Neumann boundary conditions are imposed. (In 3-D, we want the intersection of four or more patches, or the intersection of three or more and the boundary.) Consider a shape function that is one at a corner and has a support contained in the union of coarse patches abutting the corner. Fixing a single shape for each corner, let V_{corner} be the subspace of V that is the span of such shape functions.

Suppose the intersection (in 2-D) of two coarse patches is a curve on which more than two vertices lie; call that curve a coarse edge e_c. (Note that we want more than two vertices — otherwise there are no vertices interior to the edge, and V_{corner} covers the degrees of freedom on the edge.) We also consider those edges that are the intersection of a coarse patch and Γ_N. (In 3-D, we want the intersection of three or more patches, or the intersection

[3] Any of a number of procedures are available in this regard. For instance, there is the smoothed aggregation of algebraic multigrid [12] and domain decomposition methods [73].

of two or more and the boundary.) Consider a shape function that is zero at both ends of the edge, possibly non-zero in the interior of the edge, and has a support contained in the union of coarse patches abutting the edge. Record the heights of the shape function $v_{\beta_{e_c}}$ along coarse edge e_c in a vector β_{e_c}. Require that the heights of the shape function in the interior of the coarse patches that abut e_c depend linearly on β_{e_c}. That is, fix a one-to-one linear map $V_{e_c} : \beta_{e_c} \to v_{\beta_{e_c}}$ for each coarse edge[4]; note that setting all interior nodes to zero will do. Let $V_{\beta,\text{edge}}$ be the subspace of V that is the span of such shape functions, "edge bubbles."

In 3-D, we will also need face bubbles. Call the (non-empty) intersection of two coarse patches (or a coarse patch and Γ_N) a coarse face f_c. Defining face bubble shape functions in a manner similar to those for edge bubbles gives the space $V_{\beta,\text{face}}$. Let β be the vector whose entries list the entries of all β_{e_c} and β_{f_c}. (For simplicity, throughout the rest of this document e_c will represent either a coarse edge or face as appropriate.)

Finally, let $V_\beta = \delta V \oplus V_{\beta,\text{face}} \oplus V_{\beta,\text{edge}} \oplus V_{\text{corner}}$. See Figure 2.2 for a 2-D example of a coarsened mesh and the associated degrees of freedom for V_β.

Note V is the union over all β of V_β. For later convenience, define $V_0 = \delta V \oplus V_{\text{corner}}$; that is, this is the space V_β with $\beta \equiv 0$. Also define V_{edge} as the union over all β of $V_{\beta,\text{edge}}$, and V_{face} as the union over all β of $V_{\beta,\text{face}}$. Another way to define V_{edge} is as the direct sum of the ranges of the V_{e_c} over all e_c (with V_{face} defined similarly).

Define the projection P_e that extracts edge and face information. In terms of the direct sum $V = \delta V \oplus V_{\text{face}} \oplus V_{\text{edge}} \oplus V_{\text{corner}}$, if $u = \delta u + u_{\text{face}} + u_{\text{edge}} + u_{\text{corner}}$, then $P_e u = u_{\text{face}} + u_{\text{edge}}$. In general, P_e is not an orthogonal projection; usually $\delta V \oplus V_{\text{corner}}$ is not orthogonal to $V_{\text{face}} \oplus V_{\text{edge}}$. In the special case where basis shapes of V_β have minimal support — where the basis shapes for δV and V_{corner} come from the nodal basis for V, and the basis shapes for $V_{\beta,\text{edge}}$ and $V_{\beta,\text{face}}$ are linear combinations of as small as number as possible of nodal basis shapes from V — then P_e is an orthogonal projection (considering elements of V as vectors). We will briefly return to this idea of a change of basis in Section 4.4.

Let $I_\beta \colon V_\beta \to V$ be the natural inclusion.[5] Considering I_β as a matrix, a quick lemma is it is of full rank. For the nodal bases, there is a non-zero entry in each column of I_β corresponding to a degree of freedom of $V_{\beta,\text{face}} \oplus V_{\beta,\text{edge}} \oplus V_{\text{corner}}$ that appears in no other column of I_β. If β_{e_c} or β_{f_c} is identically zero on an edge or face, then the corresponding column of I_β simply is not present.

We use the inclusion I_β to construct a coarsened version of our problem. Let the

[4]For convenience, we will use V_{e_c} interchangeably to denote the map and its range.

[5]For a purely algebraic interpretation of our algorithm, one would instead pick any one-to-one map I_β from a chosen coarse space V_β to the underlying fine space V.

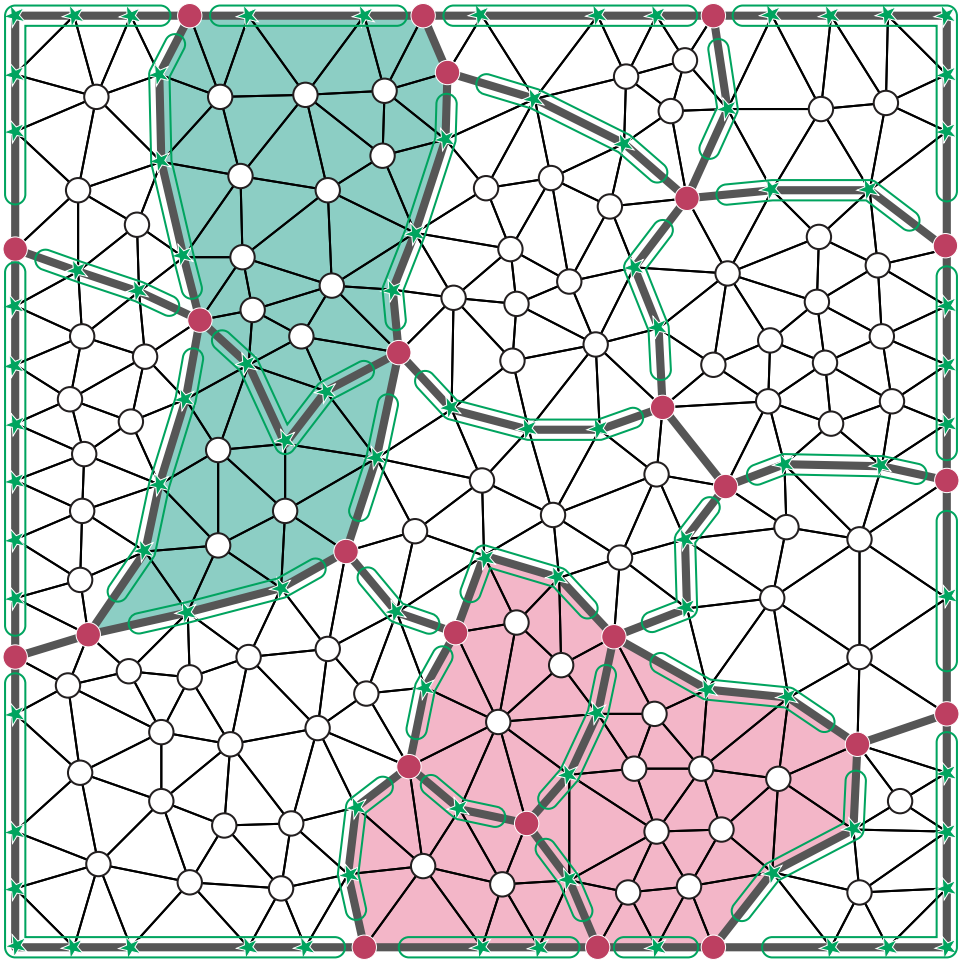

Figure 2.2: A sample coarsened domain with multiscale degrees of freedom shown. Open circles are subgrid degrees of freedom from δV. Filled red circles are corner degrees of freedom from V_{corner}. Extended green ovals are edge degrees of freedom from $V_{\beta,\text{edge}}$; stars represent degrees of freedom from the shape parameters β. The green patch represents the support for the edge in the middle of the patch; likewise, the red patch is for the corner's support. For simplicity, the case $\Gamma_N = \Gamma$ is shown so that there are degrees of freedom all along the boundary of the domain.

matrix A_β be

$$A_\beta = I_\beta^T A I_\beta$$

and the vector f_β be

$$f_\beta = I_\beta^T f.$$

Then

$$A_\beta u_\beta = f_\beta$$

is the coarsened or multiscale problem for a given β. The matrix A_β is symmetric positive definite because I_β is of full rank and A is symmetric positive definite. The coarsened problem results from the same Galerkin procedure restricted to the subspace $V_\beta \subset V$.[6]

Even though the multiscale problem has nearly as many degrees of freedom as the fine problem, it is easier to solve. The support of subgrid degrees of freedom from δV in one coarse patch does not overlap that of others in another coarse patch. The support does, of course, overlap that of some coarse degrees of freedom (those of some faces, edges, and corners). We can solve for the influence of a coarse degree of freedom on its support's coarse patches, that is, form the Schur complement of the subgrid into the coarse. In doing so, we must solve a collection of subgrid problems along with the coarse one. However, each subgrid problem is independent; we effectively decompose the whole system into pieces each of which has fewer degrees of freedom (lower resolution). Also, each coarse patch will almost surely have less heterogeneity than the whole domain, and the coarse-scale problem is an "averaged" version of the fine one so has less hetereogeneity, too.

2.3 Some important projections

We will make use of a number of projections onto V_β and related spaces. Define Z_β as

$$Z_\beta = I - I_\beta A_\beta^{-1} I_\beta^T A.$$

This is the A-orthogonal projection onto $V_\beta^{\perp_A}$, and its complement $I - Z_\beta$ is the A-orthogonal projection onto V_β. Its transpose $Z_\beta^T = I - A I_\beta A_\beta^{-1} I_\beta^T$ is its conjugate $Z_\beta = A^{-1} Z_\beta^T A$. The transpose is the A^{-1}-orthogonal projection onto $V_\beta^\perp$, and its complement $I - Z_\beta^T$ is the A^{-1}-orthogonal projection onto AV_β. These relationships are diagrammed in Figure 2.3.

We make an error in solving the coarsened problem versus the original problem. Let e_β be the error vector

$$e_\beta = u - I_\beta u_\beta = Z_\beta u.$$

[6]Keeping in mind the purely algebraic approach, technically it is the range of I_β that is the subset, that is, $I_\beta V_\beta \subset V$. If I_β is not the natural inclusion, we will sometimes abuse the notation for convenience, and we note it here to avoid confusion.

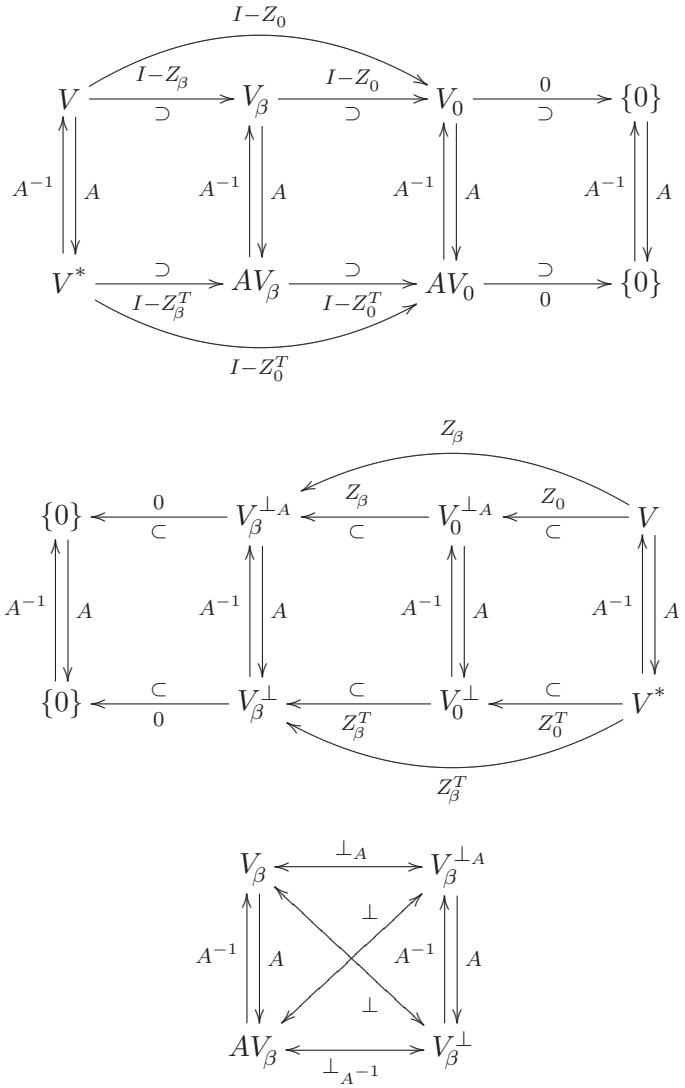

Figure 2.3: Relationships among subsets of V induced by the projections Z_β and its relatives. For every β (including 0), Z_β is an A-orthogonal projection, and Z_β^T is an A^{-1}-orthogonal projection. Because of the nested spaces $V_0 \subset V_\beta$, we have $Z_\beta Z_0 = Z_\beta$. The bottom diagram is a vertical slice through the top two diagrams along V_β; similar diagrams hold for the other spaces.

As a means for measuring sizes of errors, define $\|v\|_A = \left(v^T A v\right)^{1/2}$ for $v \in V$. The residual for the coarsened problem is

$$r_\beta = f - A I_\beta u_\beta = Z_\beta^T f,$$

and the error and residual are related through $A e_\beta = r_\beta$. That $e_\beta \in V_\beta^{\perp_A}$ can be written as $I_\beta^T A e_\beta = 0$, and that $r_\beta \in V_\beta^\perp$ as $I_\beta^T r_\beta = 0$. These relationships also can be expressed through $Z_\beta I_\beta = 0$ and/or $I_\beta^T A Z_\beta = 0$.

As noted in the first section, any given coarsened space V_β may give a poor approximation (and a large e_β). The hope lies in that the union over all β of the V_β is V; that is, we hope we can adjust β to reduce the error — to find a β so that u lies close to V_β. In the next chapters we lay out algorithms that achieve the goal of reducing the error as much as one desires.

Chapter 3

A naive application of Newton's method

We want an algorithm to find a β so that $e_\beta = 0$. That is, we want to find a root of the error e_β as a function of the shape parameters β. Newton's method is a root-finding algorithm; we can use it here. Its employ is non-trivial: e_β is a vector rational function of β.

In this chapter we describe a straight-forward application of Newton's method and the implications of this approach. Although we do not believe this is the best approach, it illuminates a path to a better method, and with further research it may yet be a viable method of its own.

The first section is a teaser on how to apply our idea to a toy 3×3 linear system; it shows our idea in action in a simple context. In the second section we describe the algorithm as applied to the finite element discretization Darcy flow of chapter 2, and follow that with a short discussion of convergence and a section on the computational infeasibility of the algorithm. Next is a section on modifying the algorithm to make it more practical. A last section describes a Jacobi-like algorithm. It is included here for completeness to use as a basis of comparison in the examples of Chapter 5.

3.1 A toy example in $\mathbb{R}^3$

Solving linear systems requires nonlinear operations (division), but a direct application of Newton's method results in a one-step procedure. That is, suppose we wish to solve the linear system $Au = f$, and we use the residual

$$F(u) = f - Au$$

as our objective function for Newton's method; the root of this function is the solution we seek. The Jacobian of the objective is

$$F'(u) = -A$$

so that the Newton step at iteration i is

$$u_{i+1} = u_i - (-A)^{-1}(f - Au_i) = u,$$

the exact solution. This fast convergence — one iteration — occurs because to solve the linear system $Au = f$, we must solve the linear system $e = A^{-1}r$ at each iteration. This direct application of Newton's method tells us nothing new; we have to find a different way to look at the problem.

Writing the unknown u using polar coordinates offers us that different view. We introduce the 3×3 linear system

$$\begin{array}{ccc} A & u & = & f \end{array}$$

$$\begin{bmatrix} 10 & -6 & 4 \\ -6 & 17 & 0 \\ 4 & 0 & 9 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 10 \\ 5 \\ -1 \end{bmatrix}$$

to provide some simple context. If we write the entries of the unknown u using polar coordinates, we get

$$\begin{array}{ccc} u & = & U_\sigma \rho \end{array}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos\theta \sin\phi \\ \sin\theta \sin\phi \\ \cos\phi \end{bmatrix} \rho$$

where the magnitude ρ in common to each entry has been factored out. The angles θ and ϕ, abbreviated in the list $\sigma = (\theta, \phi)$, indirectly tell us the shape/direction U_σ of the unknown.

Again using the residual as our objection function, we apply Newton's method to find a zero. We could write the residual as a function of both shape and magnitude,

$$r(\sigma, \rho) = f - AU_\sigma \rho.$$

This would leave Newton's method to optimize on all the parameters; instead, we will split the problem into two pieces, one linear and one nonlinear.

We compute the magnitude as a function of the shape: for a given shape we solve a

coarse problem for the "best" magnitude in the energy norm that comes from applying the Galerkin procedure

$$\big(U_\sigma^T A U_\sigma\big)\rho = U_\sigma^T f. \tag{3.1}$$

The system $U_\sigma^T A U_\sigma$ is a 1×1 system; it is easier to solve than the original 3×3 system.

For the nonlinear problem, we write the residual as a function of the shape alone,

$$r(\sigma) = f - A U_\sigma \rho,$$

where ρ is determined by (3.1). This is an overdetermined system. The collection of shapes σ can be parameterized with two degrees of freedom (the angles θ and ϕ), but the residual has three components. The collection of residuals over all shapes is a surface (an ellipsoid) in $\mathbb{R}^3$ so the Jacobian is always singular.

Applying Newton's method to finding a zero of the residual gives the algorithm:

1. Pick a shape σ;
2. Solve the coarsened problem
$$\big(U_\sigma^T A U_\sigma\big)\rho = U_\sigma^T f$$
for the magnitude ρ;
3. Calculate the objective/residual $r(\sigma) = f - A U_\sigma \rho$;
4. Calculate Jacobian $r'(\sigma)$;
5. Compute the Newton step
$$\delta\sigma = -\big(r'\big)^\dagger r;$$
6. Update the shape σ:
$$\sigma \leftarrow \sigma + \delta\sigma;$$
7. And repeat until the residual is small.

The pseudoinverse (†) is used because, as noted above, the Jacobian is singular.

3.2 An algorithm

In parallel to the 3×3 system above, applying Newton's method to finding a zero of e_β in our full problem gives the algorithm:

1. Pick a β and solve the multiscale problem $A_\beta u_\beta = f_\beta$;
2. Calculate the fine-scale error $e_\beta = u - I_\beta u_\beta$;

3. Compute the Newton step $\delta\beta = -(e'_\beta)^\dagger e_\beta$;
4. Update $\beta \leftarrow \beta + \delta\beta$; and
5. Repeat until the error is small.

One can also use the residual r_β in a similar manner:

1. Pick a β and solve the multiscale problem $A_\beta u_\beta = f_\beta$;
2. Calculate the fine-scale residual $r_\beta = f - AI_\beta u_\beta$;
3. Compute the Newton step $\delta\beta = -(r'_\beta)^\dagger r_\beta$;
4. Update $\beta \leftarrow \beta + \delta\beta$; and
5. Repeat until the residual is small.

The steps taken by the error formulation are exactly the same as those taken by the residual formulation since Newton's method is affine invariant (affine covariant).[1]

Two basic questions about the algorithm we can answer right away are: Is there a root for Newton's method to find? If Newton's method finds a root, does this give us the desired solution to the original problem? Fortunately, the answer to both questions is "yes". A root means that the error is zero $e_\beta = 0$ so that $u_\beta = u$; a root means that the residual is zero with same effect (A is non-singular and $Ae = r$). Also, there is a β^* for which $e_{\beta^*} = 0$. For instance, pick $\beta^*_{e_c} = P_e u|_{e_c}$; then $u \in V_{\beta^*}$ so that $e_{\beta^*} = Z_{\beta^*} u = 0$. Note that β^* is not unique; see below.

3.3 Convergence

We would like to know under what circumstances Newton's method converges and, if it converges, how fast it converges. We begin by examining how the shape parameters β affect the error e_β.

First, the error does not depend on the scale of β. The shapes β act as a direction and their magnitude is unimportant so β is (almost) contractible: for a non-zero C, the shapes β and $C\beta$ produce the same error $e_\beta = e_{C\beta}$ because $V_\beta = V_{C\beta}$. We can even go further and scale β edge-by-edge by a collection of non-zero C_{e_c}, one per coarse edge. Because Newton's method is affine contravariant, the size of the step $\delta\beta$ is proportional to the size of β. That is, the scale of β on a given edge does not interfere with step selection. However, there is a price to be paid. The size-redundancy in the shape parameters means the problem is over-determined; for this reason, we use the pseudo-inverse and not the inverse in the

[1] See, for example, [27] for an exposition on affine invariance including affine covariance and contravariance, among others

above algorithms. There is a further reason the problem is overdetermined: the error is A-orthogonal to V_β. We have $I_\beta^T A e_\beta = I_\beta^T r_\beta = 0$. There is a small upside, though: the root β^* can be specified in many different ways; that is, it is easier to find one root when there are many.

Beyond the two issues of non-uniqueness of β mentioned above, there are almost always no others. We use the phrase "almost always" to refer to u and not β here. An easy lemma is the following:

Lemma 3.3.1. *Given a β such that $I_\beta u_\beta = u$, if there is an e_c such that $P_e u|_{e_c} = 0$, then there are many shapes γ with $I_\gamma u_\gamma = u$.*

Proof. Construct γ with $\gamma_{e_c'} = \beta_{e_c'}$ for all $e_c' \neq e_c$, but allow γ_{e_c} to be anything desired. Then $e_\gamma = 0$ just as $e_\beta = 0$ because $u \in V_\gamma$ just as $u \in V_\beta$. □

Although this makes analysis of the naive Newton's method more difficult, computationally this is a happy situation as it means a solution is easier to find.

Our objective function, the error e_β, is well-defined for every shape β. However, it is only continuous almost everywhere (in the usual Euclidean norm).[2] In fact, e_β is a rational function of the entries of β; the inclusion I_β is linear in β, the matrix A_β is quadratic, and its inverse A_β^{-1} is rational. Also, e_β is bounded:

$$\|e_\beta\| = \|Z_\beta u\| \leq \|Z_\beta\| \|u\| = \|u\|.$$

There are essential discontinuities in e_β but only at those β where there is an e_c such that $\beta_{e_c} = 0$ — there is a change in rank of I_β. Because e_β is rational, it is also differentiable (even analytic) almost everywhere.

Because of the smoothness of e_β, it is easy to see that if β starts sufficiently close to β^*, and β^* is not at a discontinuity of e_β, then the algorithm provides quadratic convergence. That is, since the error e_β is analytic almost everywhere, it almost surely satisfies the hypotheses of the Newton–Kantorovich [49, 50, 51] or the Newton–Mysovskikh [62, 63] theorem (see also [52, 67, 24, 27]) in some neighborhood of β^* using appropriate quotient spaces. The Jacobian e_β' is singular because of the effect of V_0 and the (near) contractibility of β. We could also attack this directly as an overdetermined root-finding problem as in [23].

[2] There are other natural norms on the shape parameters. For instance, one can describe equivalence classes $[\beta]$ by the errors they generate: two shape parameters are equivalent if they produce the same error. A distance metric is given by

$$d([\beta],[\gamma]) = \|e_\beta - e_\gamma\|_A = \|(Z_\beta - Z_\gamma)u\|_A \leq \|Z_\beta - Z_\gamma\|_A \|u\|_A = \|u\|_A d(V_\beta, V_\gamma).$$

This metric has a number of properties such as e_β is trivially uniformly continuous as a function of β.

We said that if β starts sufficiently close to β^*, then we get convergence to the solution. Just how close is "close", on the other hand, has been difficult to see; how it depends on the problem parameters such as h, k, and u is unknown. Whether or not global convergence can be had via damping or line searches or trust regions is another open question; see some of the results in Section 3.5 below for some discussion.

The algorithm has quadratic convergence, but this is only an asymptotic rate. The pre-asymptotic convergence rate, however, is unknown. It can be at least as slow as linear; for instance, see the numerical example in Figure 5.2 in Section 5. Also, the convergence is not guaranteed to be monotone far from the root; in this case, sufficient damping[3] would guarantee monotonicity as the Newton step is guaranteed to be a descent direction for any positive definite quadratic form applied to the objective function (the error e_β).

Consider the level set function $F(\beta) = \frac{1}{2}\|e_\beta\|_A^2 = \frac{1}{2}e_\beta^T A e_\beta$. An alternative view of our algorithm is that we minimize $F(\beta)$. Using Newton's method to carry out the minimization gives the same steps as zeroing e_β. Likewise, minimizing the square of the A^{-1}-norm of r_β gives the same steps. Even further, since Newton's method is affine invariant, minimizing any level set function $F_M(\beta) = \frac{1}{2}e_\beta^T M e_\beta$ will do (where M is symmetric positive definite). In particular, with the identity we minimize the ordinary l^2 norm. Also, every norm on finite dimensional spaces is equivalent, so we are guaranteed asymptotic quadratic convergence in all of them. Further, the Newton direction $\delta\beta$ is special in that it is a descent direction for any of these objective functions F_M (it can be defined that way).

3.4 Computational infeasibility

At each step, we need to solve two linear systems. There is a multiscale solve for u_β given β. (If a direct solver is used for the subgrid, these problems need be solved only once — subsequent steps use the same matrix but with different right-hand sides; then only a coarse solve is necessary for each Newton step.) This solve is much less expensive than for the original problem as we have reduced the number of unknowns by the order of η^D where $\eta = H/h$ and D is the space dimension (two or three).

Once we have u_β, we need a residual evaluation followed by a Newton step solve. The Jacobian has a dimension of only the number of edge shape parameters. This, too, is much less expensive as the number of unknowns involved is from a slice of dimension one smaller (of order $\partial\Omega$ versus Ω).

Since we have separated the original large problem into two smaller ones, solvers

[3]That is, use the update $\beta \leftarrow \beta + \omega\,\delta\beta$ for some $0 < \omega < 1$ instead of $\beta \leftarrow \beta + \delta\beta$.

work faster. The catch is the operation count for computing the Jacobian r'_β. Since

$$r_\beta = f - AI_\beta u_\beta = Z_\beta^T f,$$

then

$$r'_\beta = -A(I'_\beta u_\beta + I_\beta u'_\beta)$$

by the product rule. We also have that

$$A_\beta u_\beta = f_\beta \qquad \text{and} \qquad A_\beta = I_\beta^T A I_\beta$$

so

$$A'_\beta u_\beta + A_\beta u'_\beta = (I'_\beta)^T f \qquad \text{and} \qquad A'_\beta = (I'_\beta)^T A I_\beta + I_\beta^T A I'_\beta.$$

Then

$$\begin{aligned} A_\beta u'_\beta &= (I'_\beta)^T (f - AI_\beta u_\beta) - I_\beta^T A I'_\beta u_\beta \\ &= (I'_\beta)^T r_\beta - I_\beta^T A I'_\beta u_\beta, \end{aligned}$$

and finally, recalling that $Z_\beta = I - I_\beta A_\beta^{-1} I_\beta^T A$,

$$r'_\beta = -A(Z_\beta I'_\beta u_\beta + I_\beta A_\beta^{-1} (I'_\beta)^T r_\beta).$$

A straight-forward computation of r'_β (or $e'_\beta = A^{-1} r'_\beta$) requires a number of coarse solves: each application of Z_β to a column of $I'_\beta u_\beta$, and the more explicit $A_\beta^{-1}(I'_\beta)^T r_\beta$. Not counting zero right-hand sides, there are as many right-hand sides as twice the number of edge degrees of freedom (length of β). This is, essentially, like forming the Schur complement of the degrees of freedom in V_β into their complement in V. Along with the other solves required by the algorithm, this is as expensive as solving the original problem $Au = f$ itself.

3.5 Practical application of Newton's method

Two severe problems confront us in devising a practical implementation of the above algorithm: the expense of Jacobian evaluation, and ensuring convergence from any starting shape. We make some comments on each problem; some obvious remedies do not seem to alleviate the situation. Lastly, we mention possible termination criteria.

Jacobian approximation

We do not need the Jacobian r'_β in and of itself; we only need to be able to solve the system $(r'_\beta)^\dagger r_\beta$ for the Newton step. Rather than solving the system directly, we could use an iterative solver like GMRES. This only requires being able to compute matrix vector products $r'_\beta v$ for various vectors v. These products, in turn, can be calculated using automatic differentiation or approximated using finite differences. (The Jacobian times a vector is a scaled directional derivative.) What impact this further approximation has on convergence is unknown (both the rate and the domain of attraction of the solution). Also, how to set the tolerances for GMRES and finite differences is not clear. The speed performance is also impacted as each directional derivative requires an objective function (e_β or r_β) evaluation.

Alternatively, we could use a secant method such as Broyden's method to avoid computing the Jacobian. Of course, one needs to decide what to use as the initial approximation to the Jacobian. Finite differences or automatic differentiation could be used to calculate an initial Jacobian, but as noted above, this would be prohibitively expensive without further approximations. More simply, one could start with the identity as the initial Jacobian; aside from universal (mindless) applicability, the identity is easy to invert. Broyden's "bad" update allows us to (at each step) compute a rank-one update to this inverse. In a similar vein, the identity is easy to factorize; with Broyden's "good" update one could compute a rank-one update to the factorization of the approximate Jacobian. As for the convergence of Broyden's method, one need only prove that the approximate Jacobian undergoes bounded deterioration iteration by iteration; how to approach this problem is unknown.

Both of the above strategies have been tried with some success. Further research is necessary to see if they are viable methods.

Perhaps instead one could be more crafty in the computation of r'_β. For instance, to a fair approximation, a column of r'_β has only some "large" non-zeros entries. If the degree of freedom to which the column corresponds sits on a particular coarse edge, then the "large" non-zero entries correspond to degrees of freedom in the coarse patch abutting that edge. One can then cut Ω into non-overlapping coarse patches and compute several columns at once (with a column per patch). By rejiggering patches, one can compute a few more columns, and so on. (The above applies equally to finite differences and automatic differentiation.)

"Global" convergence

Global convergence is the more important problem in that it regards the underlying exact algorithm. If the exact algorithm does not work, then we cannot expect an approximation to work (like with the approximate Jacobian above).

At first blush, it looks like we do not necessarily need global convergence in the sense of convergence from an arbitrary initial guess for β. We already have a good initial guess from ordinary multiscale elements (with their attendant error estimates and proven practical value); we need only converge from there. This initial guess can be improved further using any method that improves on multiscale elements; for instance, "Norwegian" or "Scandinavian" elements [1]. Unfortunately, some "real world" examples have shown that this is just not good enough to always get convergence with the full Newton steps. Thus it would seem our "good" initial guess cannot be guaranteed to be (uniformly) "close" in the sense of the Newton–Kantorovich or Newton–Mysovskikh theorems' requirements; that is, an additional scheme is necessary to ensure global convergence.

There are a number of schemes one can choose from. For instance, there are line-search and trust-region methods. Good reviews can be found in the textbooks [24, 54]. We have tested a scheme first proposed by Deuflhard in [25, 26] and coded in [65] and [27, Section 3.3]. See the examples in Section 5 for some sample results.

One choice that is required for Deuflhard's method is the size of the initial damping ω_0 (the damping for the first step). The algorithm can sometimes detect and recover from a poor choice, but cannot avoid a situation where this damping is required to be small (for "highly nonlinear" problems). If the initial damping is small, it can take many iterations before the damping can be reduced so that convergence is quadratic (many iterations to get close to the root β^*). Compare the left and right middle diagrams in Figure 5.2 for instance. This difficulty is a feature of the underlying root-finding problem, and not just of Deuflhard's algorithm (whether or not the initial damping is specified or computed).

The initial damping appears sensitive to problem parameters. For instance, Figure 5.10 shows a clear trend in the size of the damping as the size of the jumps in the permeability field is increased. The size of the initial damping gets smaller (the problem gets more nonlinear) as the permeability gets more heterogeneous. Some open questions are: is the damping ω proportional to the maximum eigenvalue of the gradient of the step field? Does this mean that $\omega \to 1$ as we approach the solution $\beta \to \beta^*$? Conversely, does $\omega \to 0$ as we get far from β^*? Or as $\beta_0 \to \beta_{\text{zero}}$, the equivalence class [0] of β's that produce the same (worst-size) error as $\beta = 0$? Also, would the damping ω being proportional to the maximum eigenvalue of the gradient of the step field mean that $\omega_0 \propto k_{\max}/k_{\min}$? What is the relationship between ω_0 and H/h? Does $\omega_0 \to 0$ as $H/h \to \infty$? What about when $H/h \to 1$?

Finally, sufficient damping would lead to global convergence as the Newton direction is a descent direction for $\|e_\beta\|_A$. That is, the Newton path can only go down, and there is a single minimum for the error $e_\beta = 0$ (with the only other critical point being $e_\beta = u$). Whether or not an adaptive damping scheme can achieve the required damping

automatically, though, is an open question.

As an alternative to damping, one could consider combining Newton's method with a linear, monotone convergent method. For instance, Newton's method is special because it generates a descent direction for any of the level set functions F_M. One could use the gradient of one of these to get started (assuming it is damped/scaled sufficiently to guarantee global convergence). Then one could use Newton's method to polish it off. As a second alternative, one could use conjugate gradients, multigrid, the linearized scheme (Section 3.6 below), or some other such method to get started, then use Newton's method to polish it off. Any of these has its problems, too, such as when to hand off between the two methods (the initial method and Newton's method).

Termination criteria

The algorithm never actually reaches the solution so we need a criterion for deciding when to stop. As with other iterative methods, we can use a residual termination criterion. That is, if the residual is smaller than some predetermined tolerance, or if the algorithm has reduced the size of the residual by some predetermined scale, then we stop. However, Newton's method presents another, more natural, termination criterion. In Newton's method, since β converges quadratically to β^*, the step size $\|\delta\beta\|$ converges linearly to zero. The step $\delta\beta$ is a proxy for the error $\beta^* - \beta$; stopping when the step size is smaller than a predetermined level results in the error itself being smaller than that level.

3.6 A Jacobi-like method

Jacobi's method for solving $Au = f$ estimates the error in a proposed solution $\hat{u}$ as $D^{-1}r$ where D is the diagonal of A and $r = f - A\hat{u}$ is the residual. One performs the update:

$$\hat{u} \leftarrow \hat{u} + \omega D^{-1} r.$$

We can adopt this to update basis shapes indirectly by changing

$$I_\beta u_\beta \leftarrow I_\beta u_\beta + \omega D^{-1} r_\beta$$

and then extracting edge information

$$\beta_{e_c} = P_e I_\beta u_\beta|_{e_c}.$$

This was more or less the original formulation of our method before we realized root finding methods like Newton's method were applicable. However, like the naive application

of Newton's method, this Jacobi-like method is probably not a practical one. There are the same questions about the magnitude of damping for this scheme as for the naive Newton scheme. Although it probably converges globally with sufficiently strong damping, the method only obtains a linear convergence rate. It likely does not out-perform Jacobi's method itself.

We present it here for reference to use as a comparison in the chapter on practical test examples.

Chapter 4

A geometric method: constrained Newton

The previous method suffered for two reasons. First, there is the seeming inherent difficulty in computing the Jacobian. Second, there is the necessity of damping the Newton step because of the highly non-linear problem we try to solve. In this chapter we lay out a different algorithm that avoids the computation of the Jacobian through an application of the chain rule. In doing so we also eliminate the second problem; there is a direct analogy between two dynamical systems and the two algorithms: the first is stiff — requiring either very short stepping via damping or implicit stepping — and the second not.

We begin with a geometric description of the collection of error vectors. Next we show how the chain rule can be used to avoid the use of Jacobians; this introduces the need for a projection onto the tangent space of the errors. Following that is a discussion of how to compute this projection, and a section where we compute those points where Newton's method pauses (has a zero step). We then make explicit the analogy between Newton's method and a dynamical system before finally proving global, monotone, asymptotically quadratic convergence of the method.

Last, we discuss modifications to make this algorithm practical. Some of these modifications do not impact our result; others may, but we do not know how. Since estimating the error is an unavoidable step in our algorithm, we suggest employing the algorithm as an accelerator, that is, as an outer iteration for another solver.

The last section is a summary of results and a list of conjectures on other notable features of the algorithm.

4.1 A geometric description of the collection of error vectors

The idea is that every error vector obeys the Galerkin orthogonality conditions, and every vector that obeys the Galerkin orthogonality conditions is an error vector. The key for the reverse direction lies in picking a β that generates a given error. For this we can exploit the definition of the error: $e_\beta = u - I_\beta u_\beta$. Flipping this around gives $I_\beta u_\beta = u - e_\beta$, and from the multiscale solution $I_\beta u_\beta$ we can always back out a β that generates it by examining its shape along coarse edges.

First we introduce a couple of definitions for convenience.

Definition 4.1.1. *Let $\mathcal{E}$ be the collection of all error vectors: $\mathcal{E} = \{e_\beta \; \forall \beta\}$.*

Definition 4.1.2. *Let P_{e_c} be the operation that takes a vector $v \in V$, extracts edge information $P_e v$, restricts attention to a given coarse edge $(P_e v)\big|_{e_c}$, and extends the result by zero to the rest of the degrees of freedom.*

Note that P_{e_c} is the composition of linear operators so it is linear, too. It is also a projection. Using this edge-information extracting projection, we can write a purely geometric description of the collection of error vectors $\mathcal{E}$.

Theorem 4.1.3. *The collection of all error vectors $\mathcal{E}$ is the intersection of $V_0^{\perp_A}$ and $\{e \mid e^T A P_{e_c}(u - e) = 0 \; \forall e_c\}$.*

Proof. ($\subset$) For every β we have $e_\beta \in V_\beta^{\perp_A}$ by the Galerkin property. In particular, $I_\beta u_\beta = u - e_\beta \in V_\beta$ and $P_{e_c}(u - e_\beta) \in V_\beta$ for all e_c. Thus $e_\beta^T A P_{e_c}(u - e_\beta) = 0$ for all e_c.

Also, $V_0 \subset V_\beta$ so $V_\beta^{\perp_A} \subset V_0^{\perp_A}$. Since $e_\beta \in V_\beta^{\perp_A}$ then $e_\beta \in V_0^{\perp_A}$. Thus $\mathcal{E}$ is a subset of $V_0^{\perp_A} \cap \{e \mid e^T A P_{e_c}(u - e) = 0 \; \forall e_c\}$.

($\supset$) The converse is a little trickier. Pick any vector $e \in V_0^{\perp_A}$ such that $e^T A P_{e_c}(u - e) = 0$ for every e_c. Construct β by setting $\beta_{e_c} = P_e(u - e)|_{e_c}$ on coarse edges e_c. We claim that $e = e_\beta$.

The basis shape $v_{\beta_{e_c}}$ can be written as $v_{\beta_{e_c}} = v_0 + P_{e_c}(u - e)$ for some $v_0 \in V_0$ because $\beta_{e_c} = P_e(u - e)|_{e_c}$. However, $e \perp_A V_0$ and $e \perp_A P_{e_c}(u - e)$ by assumption so $e \perp_A v_{\beta_{e_c}}$. The edge e_c was arbitrary so $e \perp_A v_{\beta_{e_c}}$ for every e_c. Since e is also A-orthogonal to V_0, then e is A-orthogonal to all of V_β. That is, e is in $V_\beta^{\perp_A}$.

That $e \in V_\beta^{\perp_A}$ means $Z_\beta e = e$. Since $u - e \in V_\beta$ by construction, we get

$$e_\beta = Z_\beta u = Z_\beta(u - e + e) = Z_\beta(u - e) + Z_\beta e = 0 + e = e.$$

Thus the claim $e = e_\beta$ is true, and $e \in \mathcal{E}$. □

A few quick remarks are in order.

The collection of all error vectors $\mathcal{E}$ is a subset (and a submanifold) of the ellipsoid $\{e \mid e^T A e = u^T A e\}$. (Or, rather, the intersection of this ellipsoid and $V_0^{\perp_A}$.) That is, the A-orthogonality of the projection Z_β means $Z_\beta^T A Z_\beta = A Z_\beta$ which gives $e_\beta^T A e_\beta = u^T A e_\beta$ by hitting both sides with u. We will make use of this (smooth) embedding when constructing approximations to parts of our algorithm. Note that this ellipsoid is only of dimension one smaller than the ambient space $V_0^{\perp_A}$. It also has a simple geometry: the volume it encloses is convex and simply connected.

The collection of residuals follows the same form. The residual obeys $A e_\beta = r_\beta$ so the collection of residual vectors is just a sheared, scaled, and/or rotated version of the error vectors. The collection of residuals shares the geometry of $\mathcal{E}$ (is diffeomorphic to and affine congruent to $\mathcal{E}$).

We have managed a purely geometric description of the collection of error vectors $\mathcal{E}$. It makes no reference to the shape parameters β, the spaces V_β, or the multiscale problem $A_\beta u_\beta = f_\beta$ and its solution u_β. We have written an analogue of the Galerkin orthogonality conditions through the edge-information extraction operation P_{e_c}. We will use geometry to help forge a new algorithm.

4.2 The chain rule and the projection onto the tangent space

Instead of using Newton's method to calculate how the shape parameters β change,

$$\delta\beta = -(e'_\beta)^\dagger e_\beta,$$

we can calculate what effect this step will have on the error. That is,

$$\begin{aligned} \delta e_\beta &= e'_\beta \, \delta\beta && \text{so that} \\ \delta e_\beta &= -(e'_\beta)(e'_\beta)^\dagger e_\beta. \end{aligned}$$

But the quantity $(e'_\beta)(e'_\beta)^\dagger$ we recognize as the orthogonal projection $P_{\tan}$ onto the tangent space of the manifold of the collection of e_β; we can write $\delta e_\beta = -P_{\tan} e_\beta$. The above calculations can only be defined almost everywhere because they involved e'_β. However, we know that the error vectors form a geometric shape with a smoothly varying tangent space. Thus one can instead *define*

$$\delta e_\beta = -P_{\tan} e_\beta$$

which is now continuous (and otherwise smooth) everywhere. Note that we can also map this back to β space to obtain a $\delta\beta$ that is well-defined everywhere too. (Although $\delta\beta$ could be well-defined everywhere — even at $\beta = 0$ — it is still only continuous among equivalence

classes; it has essential discontinuities at some zero-measure subset of β.)

The tangent space and a choice of projections

In the following two subsections we will explore how one can compute the projection onto the tangent space. We will focus on computing the tangent projection as the complement of the normal projection, and will start with a simpler case — a problem with a single coarse edge — after which we will treat the full problem. We will also discuss variants using the residual in place of the error, and variants that use an A- or A^{-1}-orthogonal projection.

Let N_0 be the dimension of V_0. Let N_e be the dimension of $V_{\text{face}} \oplus V_{\text{edge}}$; this is the length of β or the number of shape parameters. Let n_e be the number of coarse edges. Except in certain degenerate cases, the dimension of the tangent space is $N_e - n_e$ and that of the normal space is $N_0 + n_e$. Thus it seems that by computing the tangent projection as the complement of the normal, we are choosing to solve a harder problem.[1] We do this for two reasons. First, it is easier to characterize the tangent plane by its normals rather than computing an independent set that spans the tangent plane. The columns of the Jacobian e'_β span the tangent space, but as we have already noted, computing the Jacobian is impractical. Second, by a judicious choice of an oblique projection we will be able to avoid reference to the subgrid V_0. The normal projection will then be onto a space with a dimension equal to the number of coarse edges — much smaller than before and now also much smaller than the dimension of the tangent space.

A simple case: one coarse edge

In the case where there is but one coarse edge, we can rewrite the description of the collection of error vectors without reference to P_{e_c}:

$$\{e_\beta \mid e_\beta^T A(u - e_\beta) = 0\} \quad \cap \quad V_0^{\perp_A}.$$

Because there is only one e_c, the action of P_{e_c} on $(u - e_\beta)$ only serves to remove the effect of V_0 and not that of other edges. Since we already consider the intersection of the above ellipsoid with $V_0^{\perp_A}$, the result is the same. In the same vein, the collection of residual vectors can be written as:

$$\{r_\beta \mid r_\beta^T A^{-1}(f - r_\beta) = 0\} \quad \cap \quad V_0^{\perp}.$$

Rather than consider the tangent space (and the projection thereto), we will com-

[1] We are assuming that $N_0 \gg N_e \gg n_e$. That is, there are many more degrees of freedom in the interior of coarse patches than on their boundary, and there are many more degrees of freedom on coarse edges than there are edges.

pute the projection onto the normal space. We can recover the tangent projection as the complement of the normal projection. We do this because the normal space is readily computable and (if we play our cards right) is of a smaller dimension.

The normal to $\{e_\beta \mid e_\beta^T A(u - e_\beta) = 0\}$ is the vector $r_\beta - \frac{1}{2}f$:

$$\begin{aligned} (e+\delta e)^T A(u-(e+\delta e)) &= 0 \\ e^T A(u-e) + \delta e^T A(u-e) - e^T A\delta e - \delta e^T A \delta e &= 0 \\ \delta e^T A(u-2e) &= 0 \\ \delta e^T (f - 2r) &= 0. \end{aligned} \tag{4.1}$$

The span of the vector $r_\beta - \frac{1}{2}f$ with AV_0 is the normal space. Likewise for the collection of residuals, we have the span of $e_\beta - \frac{1}{2}u$ and V_0.

As a quick review, if the columns of a matrix M are linearly independent and span a space W, then $M(M^TM)^{-1}M^T$ is the orthogonal projection onto W. Of course, we need not explicitly form such a matrix to calculate the effect of the projection of a vector, but we do need to be able to solve systems of the sort $M^TMx = y$.

In the error case, a natural choice for the columns of M are the columns of AI_0 along with $r_\beta - \frac{1}{2}f$. The matrix M^TM represents a sparse system, but involves A^2 along with a dense row and column and so loses our special multiscale structure; it may be difficult to solve. That is, this is not as easy as solving a coarsened/multiscale problem, and we would like to avoid such a computation.

The situation is better for the residual formulation: the columns of M are the columns of I_0 along with $e_\beta - \frac{1}{2}u$. Computing the effect of the projection requires solving a coarse system if corner shapes overlap. If, on the other hand, corner shapes have minimal support, then all the columns of M are columns of the identity matrix along with $e_\beta - \frac{1}{2}u$, and the projection is even easier to compute. Because of the ease in computing the tangent projection for the residual formulation, we will keep this in mind. However, we still face the difficulty of computing $e_\beta - \frac{1}{2}u$ in the first place. We would like to avoid this, too.

By changing perspective a little bit, we can always avoid the coarse solves needed for the projection.[2] The problem in the error case comes from the vector $r_\beta - \frac{1}{2}f$ not being orthogonal to AV_0; if it were, we could compute the projections separately. That is, if we found a normal to $\{e_\beta \mid e_\beta^T A(u - e_\beta) = 0\}$ that lay in $V_0^{\perp_A}$ (considered now as the ambient space for $\mathcal{E}$), we could simply compute the projection onto the normal space as the projection onto this single vector.

This is indeed possible. A hint comes from the calculations for the normal in equation (4.1). The last line tells us $r_\beta - \frac{1}{2}f$ is normal to the tangent space, but the line just

[2] In this section, we will not attempt to avoid computing $e_\beta - \frac{1}{2}u$. This will be addressed later.

above it tells us $e_\beta - \frac{1}{2}u$ is A-normal to the tangent space.[3]

We will need two insights to carry out our procedure. One insight is that the Moore–Penrose pseudoinverse is not the only pseudoinverse available to us. We wanted to compute the effect of $(e'_\beta)(e'_\beta)^\dagger$. As stated before, if we use the Moore–Penrose inverse ($\dagger$), this is the orthogonal projection onto the tangent space of the collection of error vectors (the range of e'_β). That is, for any matrix M the Moore–Penrose inverse has the property that $MM^\dagger$ is the orthogonal projection onto the range of M. We can define a new pseudoinverse ($\dagger_A$) so that $MM^{\dagger_A}$ is the A-orthogonal projection onto the range of M (and keep the other properties of the pseudoinverse the same). Using this new pseudoinverse, $(e'_\beta)(e'_\beta)^{\dagger_A}$ is the A-orthogonal projection onto the tangent space of the collection of error vectors.

The Moore–Penrose inverse has another property that we will find useful later. For a matrix M with full column rank:

$$x = M^\dagger b = \operatorname{argmin} \|b - Mx\|.$$

We can instead define a pseudoinverse ($\dagger_A$) with the property that:

$$x = M^{\dagger_A} b = \operatorname{argmin} \|b - Mx\|_A,$$

and again keep the other properties of the pseudoinverse the same. Fortunately, the two definitions (projection-based and minimization-based) coincide; see Sections 2.6, 2.7, and 3.4 of [10]. One can relate the two pseudoinverses $\dagger$ and $\dagger_A$ through the formula:

$$M^{\dagger_A} = \left(A^{1/2}M\right)^\dagger A^{1/2}.$$

We can similarly define a third pseudoinverse ($\dagger_{A^{-1}}$) with respect to $\|\cdot\|_{A^{-1}}$ and A^{-1}-orthogonal projections.

Our second insight comes by noting that e_β, the vector whose projection we desire, already lies in $V_0^{\perp_A}$, a natural ambient space for the manifold of error vectors. If u did too, then the normal vector $e_\beta - \frac{1}{2}u$ would lie in $V_0^{\perp_A}$ (as would f and $r_\beta - \frac{1}{2}f$ lie in $V_0^\perp$). In general, of course, u does not lie in $V_0^{\perp_A}$, but if we remove the components of u that are A-orthogonal to V_0, we will have as we wish. We can add an extra initialization step to our algorithm to accomplish this. If we perform a multiscale computation with $\beta = 0$ and call the result u_0, then we can update $u \leftarrow u - u_0$ and store u_0 to be added back at the very

[3]The A-normal to $\mathcal{E}$ can be considered an affine normal [56, 64]. It is only *an* affine normal (there are many others), but it is the natural one to an ellipsoidal surface (the "usual" normal to the sphere).

One could also call this the conjugate normal (in the spirit of conjugate gradients). The only parallel though is in the concept that there is an alternative normal, one that uses the induced A-inner product instead of the usual Euclidean one so in this context the term "conjugate normal" is a neologism.

end of our computation. This initial step costs only a multiscale solve (which we assume is inexpensive), and guarantees that $u \in V_0^{\perp_A}$. (At the same time we update $f \leftarrow f - Au_0$ and get $f \in V_0^{\perp}$.)[4]

To sum up and review so far we list some results. For the orthogonal projection, the normals are

in the error case:	columns of AI_0	along with the vector $r_\beta - \frac{1}{2}f$,	and
in the residual case:	columns of I_0	along with the vector $e_\beta - \frac{1}{2}u$.	

For the A-orthogonal projection, the normals are

in the error case:	columns of I_0	along with the vector $e_\beta - \frac{1}{2}u$.

For the A^{-1}-orthogonal projection, the normals are

in the residual case:	columns of AI_0	along with the vector $r_\beta - \frac{1}{2}f$.

By precomputing u_0 and f_0 we get to assume

$$u, e_\beta \in V_0^{\perp_A}$$
$$f, r_\beta \in V_0^{\perp} = (AV_0)^{\perp_{A^{-1}}}.$$

This does not help us compute the orthogonal projections in either the error or residual case; that is, we do not have that $e_\beta \in V_0^{\perp}$ nor $r_\beta \in V_0^{\perp_A}$ so that solving a system is still necessary to compute the projection. However, it does help for the A-orthogonal and A^{-1}-orthogonal projections; we can get away with a matrix-vector multiply, a dot product, and a division.

Taking $V_0^{\perp_A}$ as the ambient space in the error case, the ellipsoid of error vectors is of one dimension smaller than the ambient space. Now that we have identified a normal vector to the ellipsoid, we can easily remove the orthogonal component in the direction of the normal; this is the action of the A-orthogonal projection onto the tangent space.

$$P_{\tan} e_\beta = \left(I - \frac{(e_\beta - \frac{1}{2}u)(e_\beta - \frac{1}{2}u)^T A}{(e_\beta - \frac{1}{2}u)^T A (e_\beta - \frac{1}{2}u)} \right) e_\beta$$

The tangent projection can be computed with a matrix-vector multiply along with dot products and vector scalings and additions; the linear system solve is replaced with a simple

[4]The vector $r_\beta - \frac{1}{2}f$ generally will not be in $V_0^{\perp_A}$ — it is in $V_0^{\perp}$ — unless V_0 is empty, that is, there is no subgrid. This may be a useful case in a purely algebraic approach to the problem even if not useful here; it allows one to avoid identifying a subgrid in a problem that does not naturally have one. See Chapter 7 for an example in a dense matrix, the Hilbert matrix.

scalar division. However, this form seems to imply that we need to know both the error e_β and the solution u. The need for u can be skirted; see Section 4.7 for a fix.

We can compute the tangent projection likewise for the residual case:

$$P_{\tan} r_\beta = \left(I - \frac{(r_\beta - \frac{1}{2}f)(r_\beta - \frac{1}{2}f)^T A^{-1}}{(r_\beta - \frac{1}{2}f)^T A^{-1}(r_\beta - \frac{1}{2}f)} \right) r_\beta$$

again only using dot products and vector operations along with a matrix-vector multiply (assuming we can compute the effect of applying A^{-1}).

Next we will consider the general case, but keep the above discussion in mind for the approximation of the general case.

The general case: many edges

We now need a collection of independent normals, one per edge. Recalling equation (4.1) and theorem 4.1.3, we can compute the normals in the case of many edges.

A normal to $e^T A P_{e_c}(u - e) = 0$ can be had from the following:

$$\begin{aligned} (e + \delta e)^T A P_{e_c}(u - (e + \delta e)) &= 0 \\ e^T A P_{e_c}(u - e) + \delta e^T A P_{e_c}(u - e) - e^T A P_{e_c} \delta e - \delta e^T A P_{e_c} \delta e &= 0 \\ \delta e^T (A P_{e_c}(u - e) - P_{e_c}^T A e) &= 0. \end{aligned} \tag{4.2}$$

This gives $AP_{e_c}(u-e) - P_{e_c}^T Ae$ as a normal, one per edge. The span of these along with AV_0 is the normal space. (We will see in section 4.4 that the vectors $AP_{e_c}(u - e) - P_{e_c}^T Ae$ are indeed independent.) As in the single edge case, we will also be interested in the A-normal space. This space is the span of the $P_{e_c}(u - e) - P_{e_c}^* e$ along with V_0 (where we use $P_{e_c}^*$ to denote the adjoint of P_{e_c} in the A-inner product: $P_{e_c}^* = A^{-1} P_{e_c}^T A$).

Let N be a matrix whose columns are those of AI_0 along with the vectors $AP_{e_c}(u - e) - P_{e_c}^T Ae$. Let N_A be a matrix whose columns are those of I_0 along with the vectors $P_{e_c}(u - e) - P_{e_c}^* e$. Writing out the form for the normal projections gives

$$N\big(N^T N\big)^{-1} N^T \qquad \text{and} \qquad N_A\big(N_A^T A N_A\big)^{-1} N_A^T A$$

for the normals and A-normals respectively.

Because P_{e_c} is not an (A-)orthogonal projection, the situation is worse than that for a single edge. We cannot avoid a coarse solve where the number of unknowns equals the number of coarse edges plus $\dim V_0$ — not by considering the residual nor the error formulation and not by considering orthogonality nor A/A^{-1}-orthogonality. (Keep this conundrum in mind for the later section on computing fixed points of Newton.)

If we had orthogonality between AV_0 and the normals to the shapes described by the Galerkin conditions (or A-orthogonality between V_0 and the A-normals), then at least it is smaller than the size of the coarse system in the multiscale solve — there are no corners; it is also much smaller than the system solved in the naive Newton ($(r'_\beta)^\dagger r_\beta$). However, it is still dense. On the other hand, it does have a special structure that we might be able to exploit. There is weak coupling between the edges, and the algebraic form $P_{e_c} + P^*_{e_c}$ might yield fruit.[5]

Though this problem would make computations more difficult, in practice we will avoid this situation through a judicious approximation: the collection of error vectors $\mathcal{E}$ lies in the ellipsoid $e^T A(u-e) = 0$ which has but a single A-normal that already lies in $V_0^{\perp_A}$. That is, we go back to the single-edge case. See section 4.7.

4.3 An algorithm

This brings us to a new algorithm. Applying Newton's method to finding a zero of e_β is equivalent to finding a zero of $f(e) = e$ with the constraints $e \in \mathcal{E}$ and $e \in V_0^{\perp_A}$. Thus we compute:

0. Calculate and store the multiscale solution u_0 with $\beta = 0$. Update the right-hand-side data $f \leftarrow f - AI_0u_0$.
1. Pick a β and solve the multiscale problem $A_\beta u_\beta = f_\beta$.
2. Calculate the fine-scale error $e_\beta = u - I_\beta u_\beta$.
3. Compute the effect of the Newton step on the error $\delta e_\beta = -P_{\tan} e_\beta$
4. Update the error $e_\beta \leftarrow \exp(e_\beta, \delta e_\beta)$.
5. Infer the updated β by extracting the edge information from $I_\beta u_\beta = u - e_\beta$ by setting $\beta_{e_c} = P_{e_c} I_\beta u_\beta\big|_{e_c}$;
6. Return to step (1) with the new β and repeat until the error is small; and
7. Construct the approximate solution $u \approx I_\beta u_\beta + I_0 u_0$.

The exponential map $\exp(p, d)$ follows the geodesic of the manifold $\mathcal{E}$ from the point p in the direction d for a distance $\|d\|$. Several of the above steps are obviously computationally infeasible. See Section 4.7 for a means of approximating these steps.

We will only consider the error formulation going forward. This is for two reasons: the tangent projection is equally difficult to compute for both the collection of residuals and

[5]The systems could be dense for a dense A if we were applying our solver in a pure algebraic fashion. If I_β is chosen so that there is no subgrid, then $V_0^\perp = V_0^{\perp_A}$ and only a coarse-system sized solve is needed.

the collection of errors, and we cannot impute the shape parameters β from the residual r_β like we can for the error e_β (without computing or otherwise estimating $e_\beta = A^{-1} r_\beta$).

In the following sections, we assume without loss of generality that $u \in V_0^{\perp_A}$. Step (0) in the algorithm (a throwaway multiscale computation) takes care of this.

As a minor remark, unconstrained Newton's method takes one step: $\delta e = -e$. The same goes for finding a root of $g(e) = Ae$, and for minimizing $F = \frac{1}{2} e^T e$ or $G = \frac{1}{2} e^T A e = \frac{1}{2} r^T A^{-1} r$. However, subjecting any of the four to the constraints gives our method.

4.4 Places where the Newton step is zero

In this section, we carefully examine how we extract edge shapes. We used P_e and P_{e_c} in our geometric description of the collection of error vectors $\mathcal{E}$. We also used edge information extraction to calculate normals to $\mathcal{E}$ and so, indirectly, $P_{\tan} e_\beta$. But where this is zero is where the Newton step is zero. By a reparameterization of V_β that gives the same subspace (but different edge information projections), we will be able to explicitly calculate where the Newton step is exactly zero.

We will only make use of the reparameterization of V_β here and in Section 4.6. Of the points we calculate in this section, we will not make use of precise knowledge of their location; we only need know that there is a finite number of them (they are isolated points) and that they involve a choice — one per edge. Of the edge projections we introduce in this section, we only need know of their existence. Specifically, as a practical matter, we will never need to compute the edge projections or the location of these points.

As a quick analogy to guide our thinking in this section, consider a sphere (of any dimension). The only place where a transversal is normal to the tangent space (at either end) is if it is zero or passes through the center. This is exactly the case of one coarse edge (with $A = I$) where $e_\beta = 0$ and $e_\beta = u$ are the only places where the Newton step is zero. When we generalize to many edges (tensor product of spheres), there is a question of whether we get a tensor product of points or whether the "poles" get swept out into submanifolds of one dimension smaller (or something else entirely). It turns out that the former is true: the collection of points where e_β lies in the normal space to the manifold of e_β (that is, e_β is orthogonal to the tangent space of $\mathcal{E}$ at e_β — $T_{e_\beta} \mathcal{E}$) is a finite set of isolated points; there are $2^{\#e_c}$ of them — a two-way choice per edge.

Motivation for reparameterization

The operators P_{e_c} have a number of useful properties that we have already found useful. We summarize them here.

1. P_{e_c} is a projection; its range is V_{e_c}.
2. P_{e_c} selects information from an edge independently of others: $P_{e_c}P_{\hat{e_c}} = 0$ if $e_c \neq \hat{e_c}$.
3. P_{e_c} ignores the subgrid, $P_{e_c}V_0 = 0$, but keeps V_β invariant, $P_{e_c}V_\beta \subset V_\beta$.
4. Between the collection of P_{e_c} and $P_0 = I - \sum_{e_c} P_{e_c}$, their ranges cover all V; that is, V can be decomposed in a direct sum of the ranges.

We want an additional property for P_{e_c}, that of orthogonality, while keeping all the above. This will complete our ability to consider edges independently of one another.

If we use a basis for V_β where corner, edge, and face shape functions have minimal support, then P_{e_c} is diagonal (and so is symmetric making P_{e_c} an orthogonal projection). Note that in choosing a new basis, the shape parameters β that specified a space V_β with our original basis are different from the shape parameters $\tilde{\beta}$ that now specify that same space; that is, $V_\beta = V_{\tilde{\beta}}$ but $\beta \neq \tilde{\beta}$ and V_{e_c} has changed. This will serve as our motivation for reparameterizating V_β. By doing so we will be able to construct edge information extracting (A-)orthogonal projections.

Constructing new edge projections

As we noted in section 2.2, the space V can be decomposed as a direct sum of V_0, the subgrid, and the V_{e_c}, the edges. If we form a block matrix with the columns of I_0 and the columns of the V_{e_c}, we will get a non-singular matrix. That is, choose an ordering for the edges $e_{c_1}, e_{c_2}, \ldots$. Then form the block matrix

$$\mathbf{V} = \left[\, I_0 \,\middle|\, V_{e_{c_1}} \,\middle|\, V_{e_{c_2}} \,\middle|\, \ldots \,\right].$$

This matrix has a QR factorization because it is non-singular. Let $\mathbf{V} = \tilde{Q}\tilde{R}$ if the ordinary inner product is used to compute the factorization, and let $\mathbf{V} = \hat{Q}\hat{R}$ if the A-inner product is used. We will use the "Q" part to get our reparameterization.

Suppose $n_0 = \dim V_0$, $n_1 = \dim V_{e_{c_1}}$, $n_2 = \dim V_{e_{c_2}}$, and so on. Then the first n_0 columns of $\tilde{Q}$ and $\hat{Q}$ still span V_0. However, the next n_1 columns of either do not span $V_{e_{c_1}}$; their span is something new: call it $\tilde{V}_{e_{c_1}}$ and $\hat{V}_{e_{c_1}}$.[6] We can move down the line of e_c's in this fashion defining new "edge" spaces.

Definition 4.4.1. *Let $V_{\tilde{\beta}}$ be the space spanned by V_0 and $\tilde{V}_{e_c}\tilde{\beta}_{e_c}$ for all e_c. Let $V_{\hat{\beta}}$ be the space spanned by V_0 and $\hat{V}_{e_c}\hat{\beta}_{e_c}$ for all e_c.*

[6] Again, for convenience, we will use the same symbol for a map from shape parameters to V, the space (the range of the map), and the matrix representing the map (whose columns are taken from the QR factorizations above).

Note that $\tilde{R}$ and $\hat{R}$ can be used to map from our originally defined shape parameters β to the new parameters $\tilde{\beta}$ and $\hat{\beta}$; this is a correspondence (a bijection).

Definition 4.4.2. *Let $\tilde{P}_{e_c}$ be an orthogonal edge-information extraction operator that works on $V_{\tilde{\beta}}$. Let $\hat{Q}_{e_c}$ be an A-orthogonal edge-information extraction operator that works on $V_{\hat{\beta}}$.*

These two projections have all the properties listed above for P_{e_c}, but they are also (A-)orthogonal projections. The projections $\tilde{P}_0 = I - \sum_{e_c} \tilde{P}_{e_c}$ and $\hat{Q}_0 = I - \sum_{e_c} \hat{Q}_{e_c}$ are also (A-)orthogonal. (Note that $\hat{Q}_0 = I - Z_0$, too.)

Computing the points

Now that we have set up an A-orthogonal edge-information extraction projection, computing the points where e_β is A-orthogonal to $T_{e_\beta}\mathcal{E}$ is relatively easy. As a start, we note that the new edge projection can also be used to describe $\mathcal{E}$.

Corollary 4.4.3. *The collection of all error vectors $\mathcal{E}$ is the intersection of $V_0^{\perp_A}$ and $\{e \mid e^T A \hat{Q}_{e_c}(u - e) = 0 \; \forall e_c\}$.*

From this new representation for $\mathcal{E}$, we see that A-normals are columns of I_0 and $\hat{Q}_{e_c}(u-e) - \hat{Q}_{e_c}^* e$. However, $\hat{Q}_{e_c}$ is an A-orthogonal projection so $\hat{Q}_{e_c} = \hat{Q}_{e_c}^*$. Thus we have the simpler form for the A-normal as $\hat{Q}_{e_c}(e - \frac{1}{2}u)$. (These are clearly independent.) Once we have this, the points follow immediately.

Theorem 4.4.4. *Assuming $\hat{Q}_{e_c} u \neq 0$ for every e_c, there are $2^{\#e_c}$ points where the Newton step is zero. They are those e where $\hat{Q}_{e_c} e = 0$ or $\hat{Q}_{e_c} e = \hat{Q}_{e_c} u$ for every e_c with $\hat{Q}_0 e$ chosen so that $e \in V_0^{\perp_A}$.*

Proof. That the Newton step is zero means $P_{\tan} e_\beta = 0$. In turn, this means e_β is in the normal space to $\mathcal{E}$ at e_β; thus there exist a vector λ_0 and scalars λ_{e_c} such that

$$e_\beta = I_0 \lambda_0 + \sum_{e_c} \lambda_{e_c} \hat{Q}_{e_c}(e_\beta - \tfrac{1}{2}u).$$

Applying $\hat{Q}_{e_c}$ to the above equation gives

$$\hat{Q}_{e_c} e_\beta = \lambda_{e_c} \hat{Q}_{e_c}(e_\beta - \tfrac{1}{2}u) \tag{4.3}$$

because $\hat{Q}_{e_c}$ is linear, $\hat{Q}_{e_c} I_0 = 0$, and $\hat{Q}_{e_{c_1}} \hat{Q}_{e_{c_2}} = 0$ if $e_{c_1} \neq e_{c_2}$. Adding $-\frac{1}{2}\hat{Q}_{e_c} u$ to both sides of the above equation and collecting the terms in $e_\beta - \frac{1}{2}u$ gives

$$(1 - \lambda_{e_c})\hat{Q}_{e_c}(e_\beta - \tfrac{1}{2}u) = -\tfrac{1}{2}\hat{Q}_{e_c} u.$$

Assuming $\hat{Q}_{e_c} u \neq 0$ for all e_c, we can conclude $1 - \lambda_{e_c} \neq 0$. Returning to equation (4.3) and collecting terms in e_β gives

$$(1 - \lambda_{e_c})\hat{Q}_{e_c} e_\beta = -\tfrac{1}{2}\lambda_{e_c}\hat{Q}_{e_c} u$$

so that

$$\hat{Q}_{e_c} e_\beta = -\frac{\lambda_{e_c}}{2(1 - \lambda_{e_c})}\hat{Q}_{e_c} u. \tag{4.4}$$

We know from the above corollary that e_β also lies on the cylindrical ellipsoid

$$e_\beta^T A \hat{Q}_{e_c}(u - e_\beta) = 0.$$

Since $\hat{Q}_{e_c}$ is an A-orthogonal projection, we get

$$(\hat{Q}_{e_c} e_\beta)^T A \hat{Q}_{e_c}(u - e_\beta) = 0$$

or

$$(\hat{Q}_{e_c} e_\beta)^T A(\hat{Q}_{e_c} u) - (\hat{Q}_{e_c} e_\beta)^T A(\hat{Q}_{e_c} e_\beta) = 0.$$

Substituting in what we know about $\hat{Q}_{e_c} e_\beta$ from equation (4.4), and again assuming $\hat{Q}_{e_c} u \neq 0$ so that $(\hat{Q}_{e_c} u)^T A(\hat{Q}_{e_c} u) \neq 0$, we get

$$\left(-\frac{\lambda_{e_c}}{2(1 - \lambda_{e_c})}\right) - \left(-\frac{\lambda_{e_c}}{2(1 - \lambda_{e_c})}\right)^2 = 0.$$

Thus either $\lambda_{e_c} = 0$ or $\lambda_{e_c} = 2$; it follows that either $\hat{Q}_{e_c} e_\beta = 0$ or $\hat{Q}_{e_c} e_\beta = \hat{Q}_{e_c} u$.

A quick comment about the subgrid and λ_0: the $\hat{Q}_{e_c}$ are A-orthogonal to each other and to $\hat{Q}_0$. Thus we can pick $\hat{Q}_0 e_\beta = 0$ and $\lambda_0 = 0$; then $e \in V_0^{\perp_A}$. Equivalently, hit the very first equation with $I_0^T A$. □

Strictly speaking, we have only proven the theorem for the geometry of $\mathcal{E}$ with the A-normal fiber bundle. However, the various geometries of $\mathcal{E}$ with normals and A-normals (along with the corresponding ones for the collection of residuals) are all affine diffeomorphic. Since this geometry has a discrete set, it must be that the other geometries do, too.

Regarding the condition that $\hat{Q}_{e_c} u \neq 0$, in the single edge case, $P_{e_c} u = 0$ is equivalent to $u \in V_0$. That is, $V_0^{\perp_A}$ misses the ellipsoid $e^T A(u - e) = 0$. In the many edge case, things are more complicated, but we pass on worrying about it since the collection of u such that there is an e_c where $\hat{Q}_{e_c} u = 0$ is a set of measure zero (in the usual Lebesgue measure).

Geometry of $\mathcal{E}$

We said above that $e^T A\hat{Q}_{e_c}(u-e)=0$ describes a cylindrical ellipsoid. By using that $\hat{Q}_{e_c}$ is an A-orthogonal projection, we can complete the square to get

$$\big(\hat{Q}_{e_c}(e-\tfrac{1}{2}u)\big)^T A\big(\hat{Q}_{e_c}(e-\tfrac{1}{2}u)\big)=\tfrac{1}{4}(\hat{Q}_{e_c}u)^T A(\hat{Q}_{e_c}u).$$

Note that the right-hand side is strictly positive, and $A\hat{Q}_{e_c}$ is positive semidefinite. The vector $e-\frac{1}{2}u$ is restricted by the equation to an ellipsoid in the direction of the range of $\hat{Q}_{e_c}$, but it is free in every other direction. Thus we use the name cylindrical ellipsoid.

The degrees of freedom constrained by one $\hat{Q}_{e_c}$ on one edge are independent of the degrees of freedom constrained by the projection on another edge (the axes of the cylinders are independent). Along with $e\in V_0^{\perp_A}$ then (which gets all the rest), we have an ellipsoidal toroid (a tensor product of ellipsoids). This is a manifold so so is $\mathcal{E}$; it is, in fact, a smooth (C^∞) manifold.

4.5 Analogy with dynamical systems

We adopt an approach from stability analysis of dynamical systems to prove our algorithm has monotone, global convergence. The analogy we make is important so we separate it out in its own section.

Define the Newton path[7] as the solution to the initial value problem

$$\begin{aligned}\beta(0)&=\beta_0\\ \frac{d\beta}{dt}&=-(e'_\beta)^\dagger e_\beta\end{aligned}\tag{4.5}$$

At a given β, this path heads in the direction of the Newton step at a rate proportional to the step size. Exactly the same (implied) path of β's is followed by the initial value problem

$$\begin{aligned}e(0)&=e_{\beta_0}\\ \frac{de}{dt}&=-P_{\tan}e_\beta\end{aligned}\tag{4.6}$$

because of the chain rule. This path, too, follows the direction of the Newton step.

A few remarks are in order. The fixed points of the above ordinary differential equations correspond to the fixed points of Newton's method. Each initial value problem actually represents two dynamical systems: one with the pseudoinverse ($\dagger$) and the orthogonal projection onto the tangent space, and another with the modified pseudoinverse ($\dagger_A$)

[7]The usage here of "Newton path" differs slightly from that in the literature, but the idea is similar.

and the A-orthogonal projection. Newton's method results from applying Euler's method to these initial value problems with a step $\Delta t = 1$.

The first initial value problem is stiff [40]; the non-zero singular values of e'_β can be as widely spread as those of A. A reasonable numerical approximation of a trajectory will require small steps — just as applying Newton's method directly to finding a root of e_β required damping[8] — or an implicit method. On the other hand, the second initial value problem is decidedly not stiff; a projection is as well-conditioned as can be. Full sized steps are acceptable; no damping is required.

As a minor note, the trajectories stay on the manifolds $\mathcal{E}$ and a normalized collection of shapes β. For the second differential equation, this is easy to see: $\frac{de}{dt}$ is in the tangent space to $\mathcal{E}$. For the first one, the range of $(e'_\beta)^\dagger$ is orthogonal to the kernel of e'_β. The kernel of e'_β represent directions in β for which e_β does not change; these include scalar multiples of β on edges. Hence the direction keeps β on a tensor product of spheres (one per edge).

4.6 Almost sure global monotone convergence

Now that we have a dynamical system to analyze, we can use stability analysis [39, 80] to determine whether its fixed points are stable or unstable and what their basins of attraction are. We will find that there is but one stable, attracting fixed point — $e_\beta = 0$ — and its basin of attraction is almost all points.

The first subsection will treat the continuum case and prove the above statement. The second subsection will treat Newton's method. As noted above in section 4.5, Newton's method is a discrete approximation to the continuum: it results from applying Euler's method with a step $\Delta t = 1$.

Continuum case

We introduce positive definite functions — Lyapunov functions — to measure stability around $e_\beta = 0$ of the dynamical system in equation (4.6). Let $L(e) = \frac{1}{2}e^T e$, and use this for the system with the orthogonal projection onto the tangent space. Let $L_A(e) = \frac{1}{2}e^T A e$, and use this with the A-orthogonal projection. With these Lyapunov functions, we can prove the following theorem.

Theorem 4.6.1. *For almost every u, the Newton path almost surely converges to $\beta = \beta^*$ where $e_{\beta^*} = 0$.*

[8] Quoting [27, p124], "Even far away from the solution ... the Newton direction is an outstanding direction; only its length may be too large for highly nonlinear problems."

Proof. The rate of change of L and L_A along their respective Newton paths can readily be computed. For L it is

$$\begin{aligned}\frac{dL}{dt} &= L' \cdot \frac{de}{dt} \\ &= -e_\beta^T (P_{\tan} e_\beta) \\ &= -(P_{\tan} e_\beta)^T (P_{\tan} e_\beta) \\ &= -\|P_{\tan} e_\beta\|^2,\end{aligned}$$

and for L_A it is

$$\begin{aligned}\frac{dL}{dt} &= L' \cdot \frac{de}{dt} \\ &= -e_\beta^T A (P_{\tan} e_\beta) \\ &= -(P_{\tan} e_\beta)^T A (P_{\tan} e_\beta) \\ &= -\|P_{\tan} e_\beta\|_A^2.\end{aligned}$$

Thus we always have $\frac{dL}{dt} \leq 0$ and $\frac{dL_A}{dt} \leq 0$. Moreover, $\frac{dL}{dt} = 0$ and $\frac{dL_A}{dt} = 0$ if and only if $P_{\tan} e_\beta = 0$ with $P_{\tan}$ orthogonal for the first and A-orthogonal for the second. We will consider the A-orthogonal case going forward; the orthogonal case follows from the diffeomorphism between the two cases.

Theorem 4.4.4 tells us exactly how to characterize these points. If $\hat{Q}_{e_c} u \neq 0$ for every edge e_c (the "almost every u" assumption), then either $\hat{Q}_{e_c} e_\beta = \hat{Q}_{e_c} u$ or $\hat{Q}_{e_c} e_\beta = 0$ for every e_c. We claim the only stable fixed point is $e_\beta = 0$ where $\hat{Q}_{e_c} e_\beta = 0$ for every e_c; every other one is either semi-stable or unstable. (In fact, the only unstable one is $e_\beta = u$ where $\hat{Q}_{e_c} e_\beta = \hat{Q}_{e_c} u$ for every e_c; the others are all semi-stable.)

A stable fixed point is a local minimum of L along the manifold $\mathcal{E}$. If $e_\beta \neq 0$, then e_β being a fixed point means there is at least one edge e_c where $\hat{Q}_{e_c}(e_\beta - u) = 0$. Thus $\hat{Q}_{e_c}(I_\beta u_\beta) = 0$. Considering the space of shape parameters β — where it is feasible to move every direction — almost any direction will do to find a smaller error (and L). For instance, pick the direction $\hat{\beta}^*_{e_c}$ to move in β-space (where β^* is a shape that gives the solution, and $\hat{\beta}^*_{e_c}$ is the corresponding shape in the reparameterized space restricted to e_c).

To sum up, there is but one stable fixed point at $L_{(A)} = 0$, and we know $\frac{dL_{(A)}}{dt} < 0$ save for the other (semi-stable or unstable) fixed points. We can be more discriminating by examining edges one at a time.

We claim the basin of attraction of the stable point $e_\beta = 0$ is almost all β (almost all of $\mathcal{E}$). This is the "almost sure" in the conclusion of the theorem. To prove this, we

consider $L_{A,e_c}(e) = \frac{1}{2}\|\hat{Q}_{e_c}e\|_A^2 = L_A(\hat{Q}_{e_c}e)$. It follows that

$$\frac{dL_{A,e_c}}{dt} = -(\hat{Q}_{e_c}e_\beta)^T A(P_{\tan}e_\beta).$$

Since $\hat{Q}_{e_c}$ and $P_{\tan}$ are both A-orthogonal projections, and further since $\hat{Q}_{e_c}$ and $P_{\tan}$ commute, we get

$$\frac{dL_{A,e_c}}{dt} = -\|P_{\tan}(\hat{Q}_{e_c}e_\beta)\|_A^2.$$

This is only zero if $\hat{Q}_{e_c}e_\beta = \hat{Q}_{e_c}u$ or $\hat{Q}_{e_c}e_\beta = 0$ by theorem 4.4.4 (the conditions on other edges are satisfied trivially); this is otherwise negative.

The subset of $\mathcal{E}$ where there is an edge where $\hat{Q}_{e_c}e_\beta = \hat{Q}_{e_c}u$ or $\hat{Q}_{e_c}e_\beta = 0$ is of measure zero; at all other points L_{A,e_c} is strictly decreasing. As this applies to all edges, almost any starting point for a trajectory must have $L_{A,e_c} \to 0$ as $t \to +\infty$ for all e_c and $L_A = \sum_{e_c} L_{A,e_c} \to 0$ as well. □

We make three remarks. First, the basin of attraction of $e_\beta = 0$ is not all space save the other fixed points; it is slightly less. There are trajectories that have the other fixed points as limit points.

Second, evolution along trajectories is not a contraction of L (even though level sets are nested). This is easy to see in the single edge case: two points on "opposite" sides of the "pole" $e_\beta = u$ will head in opposite directions around the sphere towards $e_\beta = 0$. That is, this line of reasoning cannot be used to conclude that the method converges.

Last, even though evolution along trajectories is not a contraction, along a single trajectory the Lyapunov function $L_{(A)}$ always decreases: $\frac{dL_{(A)}}{dt} < 0$. Thus our proof showed $\|e_\beta\|$ decreases monotonically to zero along the Newton path with an orthogonal $P_{\tan}$. Likewise $\|e_\beta\|_A$ decreases monotonically to zero along the Newton path with an A-orthogonal $P_{\tan}$. It appears that on the two different paths only the associated norm decreases. However, the situation is actually much better: on both paths, the level set functions $F_M(e) = \frac{1}{2}e^T Me$ monotonically decrease (where M is a symmetric positive definite matrix) once we get close to the origin. We have the following theorem.

Theorem 4.6.2. *Suppose B and M are symmetric positive definite matrices, and $P_{\tan}$ is the B-orthogonal projection onto the tangent space of $\mathcal{E}$. There is a neighborhood in $\mathcal{E}$ of the origin in which the Newton direction (using the B-orthogonal projection) always reduces the M-norm.*

Proof. At a given $e \in \mathcal{E}$, the Newton direction is $-P_{\tan}e$. Let

$$\begin{aligned} G &= \{\ e \in \mathcal{E} \mid -P_{\tan}e \text{ is a descent direction for } e \text{ in the } M\text{-norm}\ \} \\ &= \{\ e \in \mathcal{E} \mid e^T M P_{\tan} e \geq 0\ \}. \end{aligned}$$

The equality holds because $\|e\|_M^2 = e^T M e$. The set G contains more than just the origin as nearby points with $\|e\|_M$ small also belong; this will follow from some simply geometrical estimates. If κ is the maximum principle curvature of $\mathcal{E}$ at the origin in the B induced metric, then the central angle made by e is approximately $\kappa\|e\|_B$. The B-inner product of e and $P_{\tan}e$ is bounded below by

$$\|e\|_B^2\big(1 - \tfrac{1}{4}\kappa^2\|e\|_B^2\big).$$

The B-norm and M-norm are equivalent, though, so that the above quantity is bounded below by

$$\frac{\lambda^B_{\min}}{\lambda^M_{\max}}\,\|e\|_M^2\left(1 - \frac{1}{4}\frac{\lambda^B_{\max}}{\lambda^M_{\min}}\,\kappa^2\|e\|_M^2\right)$$

where the λ's are the eigenvalues of B and M as noted. For sufficiently small $\|e\|_M$, the above is positive. □

The estimate in the proof notwithstanding, we conjecture that these neighborhoods are actually quite large — half of $\mathcal{E}$. Regardless of their size, though, the intersection of these neighborhoods (over all M for a fixed B) is just the origin. That is, no matter how close we are to the origin (the solution), there is always an M-norm that will increase when we step in the Newton direction. However, as we get close to the origin, these non-decreasing norms degenerate.

This theorem applies with B and M equal to A or I. For instance, when $B = A$ and $M = I$, we have the A-orthogonal tangent projection measured in the ordinary Euclidean norm.

Discrete case

As we saw above, the Newton direction $\delta e_\beta = -P_{\tan}e_\beta$ is always tangent to a path along which a Lyapunov function decreases. Its size is limited, too: $\|\delta e_\beta\| \leq \|e_\beta\|$ because a projection can only decrease its size (using the ordinary norm or the A-norm as appropriate). However, $\|e_\beta\|$ is the as-the-crow-flies distance direct to the origin ($e_\beta = 0$). Further, the as-the-crow-flies distance is less than the geodesic distance along the manifold $\mathcal{E}$. Thus, even as the Newton direction is a descent direction for the Lyapunov function, the Newton step also can never overshoot the origin, the minimum of the Lyapunov function. The following

corollary results.

Corollary 4.6.3. *For almost every u, and from almost every initial value β_0 for the shape parameters, the Newton method of section 4.3 converges to $\beta = \beta^*$ where $e_{\beta^*} = 0$. The convergence is asymptotically monotone and quadratic.*

Monotonicity follows from the remarks after theorem 4.6.1 and the reasoning above. The quadratic convergence rate follows from Newton's method. The only twist here is that the domain of our objective function is a smooth manifold and not ordinary Euclidean space $\mathbb{R}^n$. See [3].

Given the equivalence of the dynamical systems, we can conclude that the direct application of Newton's method to finding a root of e_β also converges globally and monotonically for sufficiently strong damping. Note that the undamped method does not share this property; generally $\delta\beta$ is too large and results in overshooting the root. On the other hand, δe_β is naturally size limited; it cannot overshoot.

4.7 A practical geometric method

We face a number of practical problems in implementing the above. First we discuss the problem of the use of the error e_β in a central place in the algorithm; this is the only difficult problem with the algorithm. The other problems we face are much easier to tackle. These include computing an approximate tangent projection, initializing the algorithm, computing an approximation to geodesics on $\mathcal{E}$ (the exponential map), along with several others.

Error estimation and use as an accelerator

Our method requires knowledge of the error, but if we knew the error, we would have no need for our algorithm. We need to estimate the error instead (in an accurate fashion).

For instance, we could apply a few iterations of a smoother[9] to the residual to obtain an error estimate. Easy criteria have already been developed for inexact Newton methods to judge whether we maintain convergence (and superlinear or quadratic convergence at that). See, for example, Sections 2.1.5, 2.2.4, and 2.3.3 from [27] (or Sections 3.2.3, 3.3.4, and 3.4.3 for the global results). This issue seems ripe for quantification. However, judging whether global convergence is retained (in general or in the case of a particular smoother) will take some further analysis.

More generally, we can view our method as an accelerator for *any* iterative method. Use just a single (or a few) iteration(s) of your favorite iterative method to estimate the error. This is the "inner" iteration. Take this error estimate and use it in a single iteration of

[9] Jacobi, Gauss–Seidel, (S)SOR, ILU, IC, among others, for instance.

the geometric Newton's method. This is the "outer" iteration. Again, we can use already proven results on inexact Newton methods to judge whether superlinear convergence is achieved, but judging whether global convergence is retained will take further work.

As a few examples, one could also use conjugate gradients or even an approximate direct method such as an incomplete factorization. Also, at any iteration we have solved a (multiscale) finite element problem. Perhaps we can apply some ideas from *a-posteriori* analysis to estimate the error or further improve an estimate.

As another example, one could pair our method with multigrid. However, as a step in our algorithm we already effectively compute a coarse grid correction. It seems reasonable, then, to just pair with a smoother.

Simply pairing with a smoother seems sensible for another reason: smoothers act locally. The residual is zero on the subgrid (V_0) and non-zero on edges; conveniently enough, we only care about the error on edges. As noted before, to a fair approximation edges are independent; block Jacobi with blocks by coarse edges (ignoring the subgrid in computations) seems like a good idea — inexpensive and reasonably accurate.

Computing the projection onto the tangent space

Computing the A-orthogonal projection onto the normal space requires solving a system approximately the size of a multiscale problem; the edge projection P_{e_c} fouls up orthogonality to AV_0 (or A-orthogonality to V_0). This might be an acceptable cost except that this system does not have the special structure of a multiscale problem. The blocks involving coarse degrees of freedom are dense. On the other hand, the coarse system has weak coupling between edges. The system could be approximated by a diagonal one because of the weak coupling if the effect of the subgrid is ignored, too.

A much simpler approximation, conceptually and computationally, is using the enclosing ellipsoid $e_\beta^T A(u - e_\beta) = 0$ to approximate $\mathcal{E}$. This ellipsoid has but a single A-normal in $V_0^{\perp_A}$ so the "solve" becomes just a dot product and a scalar division. We do not know the precise effect of this approximation, but it has been used in our code with seeming success. The practical examples in Chapter 5 bear out that it does work well in practice.

The formula for the enclosing-ellipsoid approximation to the tangent projection could be computed using only matrix-vector multiplies and vector dot products. However, there are several equivalent forms for this expression; we can use the identities $u = e_\beta + I_\beta u_\beta$, $e_\beta^T A e_\beta = u^T A e_\beta$, $Ae_\beta = r_\beta$, and $Au = f$ to generate a variety of formulas. Though they are equivalent when using the exact quantities, they may or may not produce the same result when using approximate quantities. As noted in the above section, we must approximate the error e_β. Rounding errors may also have an effect. One is naturally lead to the questions: under what conditions are they equivalent in the face of approximation? If they are not

equivalent, which one is best to use?

For certain, the solution u is not computationally available; however, we will assume e_β is (via the previously discussed approximation). To tackle this first problem, instead of calculating $e_\beta - \frac{1}{2}u$ we can calculate $\frac{1}{2}(e_\beta - I_\beta u_\beta)$ since $u = e_\beta + I_\beta u_\beta$. Thus

$$P_{\tan} \approx I - \frac{(e_\beta - \frac{1}{2}u)(e_\beta - \frac{1}{2}u)^T A}{(e_\beta - \frac{1}{2}u)^T A(e_\beta - \frac{1}{2}u)}$$

becomes

$$P_{\tan} \approx I - \frac{(e_\beta - I_\beta u_\beta)(e_\beta - I_\beta u_\beta)^T A}{(e_\beta - I_\beta u_\beta)^T A(e_\beta - I_\beta u_\beta)}$$

or

$$P_{\tan} \approx I - \frac{(e_\beta - I_\beta u_\beta)(r_\beta - \frac{1}{2}f)^T}{(e_\beta - I_\beta u_\beta)^T(r_\beta - \frac{1}{2}f)}$$

using $Ae_\beta = r_\beta$ and $Au = f$ for the last formula.[10]

For further flexibility, note that we do not need to compute the tangent projection of any vector; we just need to compute it for the error e_β. For instance, one could apply the formula

$$(e_\beta - \tfrac{1}{2}u)^T A e_\beta = \tfrac{1}{2}u^T A e_\beta = \tfrac{1}{2}f^T e_\beta$$

in computing the numerator of the above fraction. In an exercise in substitution, one can arrive at the symmetric-looking formula:

$$P_{\tan} e_\beta \approx \frac{1}{C}\big((f^T I_\beta u_\beta)e_\beta + (f^T e_\beta)I_\beta u_\beta\big)$$

where $C = (e_\beta - I_\beta u_\beta)^T A(e_\beta - I_\beta u_\beta)$.[11] Other such formulas are certainly possible. The one above is useful because it only relies on the computationally available quantities f and $I_\beta u_\beta$ along with the assumed error approximation e_β (it does *not* rely on the unknown u).

Assumption that the solution is non-zero on edges

The assumption that $\hat{Q}_{e_c} u \neq 0$ for every e_c is almost sure to happen (with a uniform measure of solutions u). If it does not happen, it actually makes things easy from a computational point of view: any shape will do for that edge. We might worry that our procedure breaks down; the only thing that can is the tangent space projection. However, our approximation

[10] Another item of note is that the denominator is constant independent of β: it is a multiple of $u^T A u$ in each of the three formulas above. Is it better to just approximate the constant once and for all? Probably not.

[11] As noted before, this is a constant. It is equal to $u^T A u$. If the error is small $e_\beta \approx 0$, then it seems reasonable to approximate $u^T A u \approx (I_\beta u_\beta)^T A(I_\beta u_\beta)$.

using the whole ellipsoid always is well defined — unless, of course, $u \in V_0$ and the algorithm ends at the first step.

"Near" violations are not a problem; there still is a shape on the edge. Small edge magnitudes relative to other edges means there is a small error relative to other edges. In other words, mistakes in the shape make for small errors when multiplied by the small coarse edge values in u_β; that is, $I_\beta u_\beta$ will be small along the edge, too.

Initialization

We know using $\beta = 0$ is bad (it is a stationary point), but any other initial value is almost sure to produce a non-zero descent direction. A natural choice is a β that gives ordinary coarse shapes for the multiscale problem. Rounding errors and other approximations will also tend to prevent us from landing exactly on a fixed point. Also, practical computational experience with multiscale methods show they often produce excellent approximations to the true solution.

The shape of L restricted to $\mathcal{E}$ is well-approximated by a quadric. When we start near the solution, Newton's method will converge quadratically to it. On the off chance we do start near a fixed point, by symmetry, we will move quadratically fast away from it. That is not so impressive, of course, because it does not imply we move quickly towards the solution; this can still be quite slow. For our algorithm, it would seem to be the best we can do. On the other hand, if we somehow we could detect this condition, it might behoove us to use a few iterations of another iterative solver to first move us away from the fixed point, then switch back to Newton to polish it off.

Approximating geodesics

The exponential map $\exp(p, d)$ where p is a point on a manifold and d is a tangent vector takes one along a geodesic (shortest distance path) from p for a distance $\|d\|$ in the direction of $d/\|d\|$. One can write an ordinary differential equation which describes geodesics: the acceleration along the curve is normal to the surface of a magnitude of the directional curvature of the surface; there is no lateral acceleration along a geodesic. In general, this is a difficult map to compute.

On the other hand, there is a very simple approximation: taking p as a vector (say, from an embedding of the manifold into $\mathbb{R}^n$), then

$$\exp(p, d) \approx p + d.$$

That is, approximate the geodesic by its initial tangent. Our manifold $\mathcal{E}$ already has a

natural embedding (which we have been using all along); it is very easy to compute

$$\exp(e_\beta, \delta e_\beta) \approx e_\beta + \delta e_\beta.$$

This approximation, though, is not a "retraction". That is, the new vector $e_\beta + \delta e_\beta$ is no longer on the manifold. However, when combined with imputing the coefficients, it can be considered as a retraction on the shape parameters β. Substituting a retraction for the exponential map does not affect the asymptotic convergence rate [3]. Perhaps it affects robustness, but we have not seen any poor behavior in our computational experience so far.

The tangent approximation is essentially what is used in the naive Newton's method of chapter 3. We could go back and substitute $\exp(\beta, \delta\beta)$ for $\beta + \delta\beta$ since computing the exponential map on a sphere (or tensor product of spheres) is possible to do simply with trigonometric functions. (Or one could substitute a higher order approximation to the exponential.)

Successor multiscale solution

In the implied update to $I_\beta u_\beta \leftarrow u - e_\beta$ in step (5) of the algorithm, we can avoid the need for the unavailable solution u. If e_{prev} is the error from the previous step and e_{next} is the error in the current step, then

$$e_{\text{prev}} = u - (I_\beta u_\beta)_{\text{prev}}$$

$$e_{\text{next}} = u - (I_\beta u_\beta)_{\text{next}}$$

so that

$$(I_\beta u_\beta)_{\text{next}} = (I_\beta u_\beta)_{\text{prev}} + e_{\text{prev}} - e_{\text{next}}.$$

Thus we can substitute to get rid of the unknown u in favor of the (presumably) computationally available (or, rather, already computed or approximated) errors. This substitution is exact if the errors are. We hope, of course, that this substitution has minimal effect when, say, the errors are approximated. Some computational experience indicates that this is not the case — this substitution does not cause any (further) problems.

Note that if we use a linear approximation to the geodesic $\exp(e_\beta, \delta e_\beta) \approx e_\beta + \delta e_\beta$, then this update is just

$$(I_\beta u_\beta)_{\text{next}} = (I_\beta u_\beta)_{\text{prev}} - \delta e_\beta$$

with similar results for higher order approximations to the geodesic.

Imputing β from $I_\beta u_\beta$

Imputing β from $I_\beta u_\beta$ is an easy calculation. We can always construct β by going edge-by-edge and defining

$$\beta_{e_c} = \big(P_e(I_\beta u_\beta)\big)\big|_{e_c}.$$

Computing $(P_e v)|_{e_c}$ is an easy calculation: ignore entries in the vector v corresponding to the subgrid and just look at entries corresponding to edges. If corner shape functions have support in a single fine patch, ignore the corner entries too (and just copy edge entries); if not, subtract from edge entries an amount corresponding to $\sum_{\text{corners}} \alpha_{\text{corner}} v_{\text{corner}}$ where α_{corner} is the entry in $I_\beta u_\beta$ at the central corner of v_{corner} (we use a Lagrangian basis — v_{corner} has height one there).

At the same time we calculate the new shape β, we can normalize it:

$$\beta_{e_c} \leftarrow \frac{\beta_{e_c}}{\|\beta_{e_c}\|}.$$

This will avoid poor scaling of A_β.

Solvers for subgrid and coarse subproblems

As noted in chapter 2, to solve a multiscale problem we split it into two pieces. We first solve for the influence of the coarse degrees of freedom on the subgrid, then substitute this in a coarse problem and solve it.

Our algorithm calls for solving a sequence of multiscale problems. As the sequence evolves, though, only the coarse shapes change; the subgrid shapes stay fixed. If we use direct solvers for the subgrid problems, we only need to compute a factorization of these matrices once. When the coarse shapes change, computing new influence functions can be done cheaply via the already computed factorizations; that is, the right-hand side data for the subgrid problems changes, but the problems themselves do not. Using direct solvers for the subgrid problem is also reasonable since we consider these problems to be "small".

On the other hand, there does not seem to be a clear choice for the coarse problem. We cannot reuse a factorization from iteration to iteration (not even a low-rank update is available), and the size and ill-conditioning of the coarse problem may also preclude using a direct solver (on the other hand, it might not). The alternative, of course, is using an iterative solver such as preconditioned conjugate gradients. Another possibility would be to use our algorithm in a recursive fashion.

If an iterative solver is used (for either the subgrid or the coarse), we must ask what effect this will have on our algorithm. That is, we are left with an inexact $I_\beta u_\beta$. It seems, though, that this can be accounted for by theorems on inexact Newton methods just as it

would be for estimating the error.

Other problems

Another issue is that of parallelism in the algorithm. The subgrid solves and updates can naturally be done in parallel since these are independent of one another. Dot products and residual evaluations can also be done in parallel (with a minimum of communication) as can coarse matrix assembly. On the other hand, coarse solves may be tricky if the problem is large and the data are spread across many processors; combining a domain decomposition method with preconditioned conjugate gradients seems reasonable. However, if we balance the sizes of the subgrid and coarse problems so that there are about the same size, and if we make the subgrid problems large so that they are assigned to a single processor, then the coarse solve can be assigned to a single processor. This would double its work load relative to other processors but only at this one step in the algorithm. It seems likely that the other processors could be kept busy most of the time.

Like the naive application of Newton's method in chapter 3, quadratic convergence implies the step size converges linearly to zero and so can be used in an easy-to-understand stopping criterion: the step size is approximately the error size. We need not introduce any opaque tolerance to set. What the cumulative effect of the other approximations has we do not know. In our computational experience so far it has performed well (it has not lead to any premature or delayed exits from the algorithm).

Note that the exact method requires no damping under any circumstance. Most of the approximations we introduce seem unlikely to change this; computational experience bears this out. On the other hand, approximating the error (using this method as an accelerator) may requiring damping. Further research and computational experience is needed.

4.8 Summary

We have developed a variant of Newton's method to optimize the basis shapes for a flow problem so that they match the shape of the solution. The exact version of the algorithm converges globally and monotonically with a quadratic asymptotic rate. The algorithm computes no Jacobians, needs no damping, and otherwise has no opaque parameters to set. The algorithm is readily approximated to be computationally inexpensive, and computational experience indicates these approximations do not affect performance. The algorithm needs an externally provided error estimate at each iteration; this portends our algorithm's use as an accelerator for that external error estimation procedure (whatever it may be).

To sum up the algorithm: pick coarse edge shapes, solve a multiscale problem, calculate a residual and an error estimate, calculate the tangent space projection, impute new shapes, and repeat as necessary. The projection operation is well-conditioned and easy computed (or approximated). The multiscale solve is broken into many pieces: lots of subgrid problems (subsection of permeability field is almost sure to have lower heterogeneity and the subgrid problem is itself of lower resolution) and a coarse problem (permeability is "averaged" and so is less heterogeneous and the problem is a lower resolution one). The error estimate is externally provided; however this estimate is given, we conjecture that we improve on it.

We have a number of conjectures on other notable features of the algorithm. Computational experience along with some incomplete theoretical results indicate the algorithm is robust with respect to the ill-conditioning of the underlying fine problem. The contributors to the ill-conditioning are the resolution, h, or the number of degrees of freedom; and the heterogeneity, $k_{\max}/k_{\min}$, or the relative roughness of the eigenvectors of A. Neither of these appears to impact the number of iterations needed for convergence.

Chapter 5

Onward and upward: interesting "real world" examples

We present applications of our algorithms to problems of a difficulty more interesting to practitioners.

In every one of the following examples we simulate a quarter five-spot-like problem. We use a 2-D square domain with homogeneous Dirichlet conditions, a source and sink in opposite corners, and no gravity. Piecewise bilinear elements on a uniform square grid were used to discretize the problem;[1] the coarse grid was a uniform square grid as well. The resolution (the fine and coarse grid spacing) and/or the coefficient k were varied from problem to problem.

As was noted in Section 3.5, because Newton's method has quadratic convergence, a useful termination criterion is the step size being small. We can choose to use the step size in β rather than in e_β. When the edge shape functions are normalized to a unit size, the relative and absolute accuracy in β is the same (the error is not a moving target if we change the right-hand side data). We also chose to use a root-mean-square measure rather than just the l_2 norm so that the termination criterion was independent of the (coarse and fine) resolution; that is, we are able to make an apples-to-apples comparison of the number of iterations required across problems of varying resolution (h and H). We chose a termination tolerance of the square-root of the precision used (half the digits available) figuring that rounding errors might prevent us from achieving better accuracy; when IEEE double precision was used, this means we achieved full single precision accuracy.[2]

[1] Though Chapter 2 describes our methods for piecewise linears on triangles, the ideas work virtually the same. And, as can be seen in this chapter, the results are the same.

[2] If the edge shape functions are normalized in magnitude, then relative and absolute accuracy of them are comparable. If this accuracy is of size ϵ, then $\|e_\beta\| \lesssim \|u\|\epsilon$. Also, the root-mean-square error is equivalent to L_2 norm of the trace (on coarse edges) of the shape functions.

It was checked to be sure that the algorithms were always in the regime of quadratic convergence when they halted (and did not terminate prematurely when the error and step size were of different orders). See, for example, the convergence histories in the next section. In every computation, uniform shapes ($\beta = \mathbf{1}$) are used as an initial guess.

These results are as a stand-alone method. In each of the examples, we computed the full Jacobian r'_β for the naive Newton method, and we computed the exact error e_β for the geometric method despite that this makes both algorithms computationally infeasible. We reasoned that if the algorithms did not work well with this information, then any approximation to them would not work well either. That is, we wished to separate aspects of the underlying algorithms and the effects of introducing further approximations. Further research on the geometric method as an accelerator is certainly in order.

However, with regards to the other computational considerations raised in Sections 3.5 and 4.7, we implemented all the other recommendations to make for a practical algorithm. Notably, we used approximations to the tangent projection and the exponential map in the geometric method, and we used adaptive damping for the naive Newton method. (A direct solver was used to get a reference solution.)

5.1 Convergence histories

We present convergence histories on a small-ish problem to demonstrate notable differences in the behavior of the three algorithms. A 10×10 fine grid was used with a 2×2 coarse grid (there were 16 shape parameters in β). In one set of computations, constant coefficients ($k = 1$) were used. In another set the coefficients from Figure 5.1 were used; these are moderately heterogeneous.

As can be seen in Figure 5.2, the Jacobi-like algorithm gets linear convergence, and the two Newton methods get quadratic (asymptotic) convergence. The naive Newton method does experience a delay in the onset of quadratic convergence, though, for heterogeneous coefficients; the initial convergence seems linear (at best). The geometric method, on the other hand, experiences no such degradation with the increase in heterogeneity. (We have not been able to prove this, but see Section 5.3 for further discussion.)

The convergence is monotone all around. We have only been able to prove this for the geometric method; see Section 4.6 in Chapter 4 above.

The rate constant for the Jacobi-like algorithm is about 0.43/iteration for constant coefficients, and is about 0.87/iteration for the heterogeneous coefficients. This convergence rate is not so bad (depending on your point of view). And each iteration is fast — sort of: there is a coarse solve per iteration. Before the advent of the geometric method, the cost advantage over the naive Newton looked acceptable (where there was a required Jacobian

evaluation), but now it seems there is *no* advantage to the simplicity of the Jacobi-like method.

It is an open question how the rate constant of the Jacobi-like method depends on the level of heterogeneity (and how it depends on h and H/h). The corrector damping was set to 2/3 for constant coefficient problem. There was some "tuning" needed for heterogeneous coefficients; ultimately a damping of 1/8 was used.

The adaptively damped naive Newton method used no initial damping $\omega_0 = 1$ for the constant coefficient problem. For the heterogeneous coefficients, some tuning was needed; a value of $\omega_0 = 1/10$ was used ("mild nonlinearity").

Figure 5.3 shows how the shapes along coarse edges evolve. Iterates from the geometric method as applied to the heterogeneous coefficients are shown. The initial, uniform shape is easily visible. On each edge, the second iterate has already made significant progress towards the solution shape. Second and later iterates are indistinguishable from the solution shape.

5.2 Problem size insensitivity ($h \to 0$)

The number of iterations required by the two Newton methods appears to be insensitive to the size of the underlying problem (or the size of h, the discretization spacing). We have not been able to prove this, but the computational evidence is compelling. It also comports with the fact that when Newton's method is used to solve approximations to nonlinear partial differential equations, the number of iterations used is insensitive to the resolution

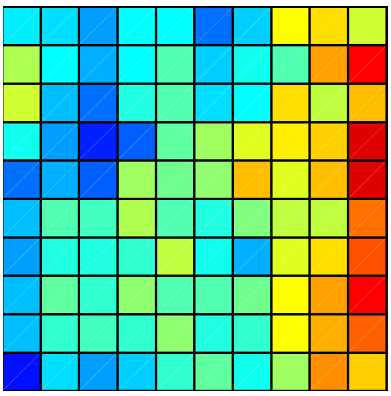

Figure 5.1: A sample 10×10 permeability field. The base-10 logarithm is plotted. Red indicates high permeability; the greatest is 1160 mD. Blue indicates low permeability; the smallest is 0.725 mD. The permeabilities span over three orders of magnitude. Data were subsampled from those shown in Figure 1.2.

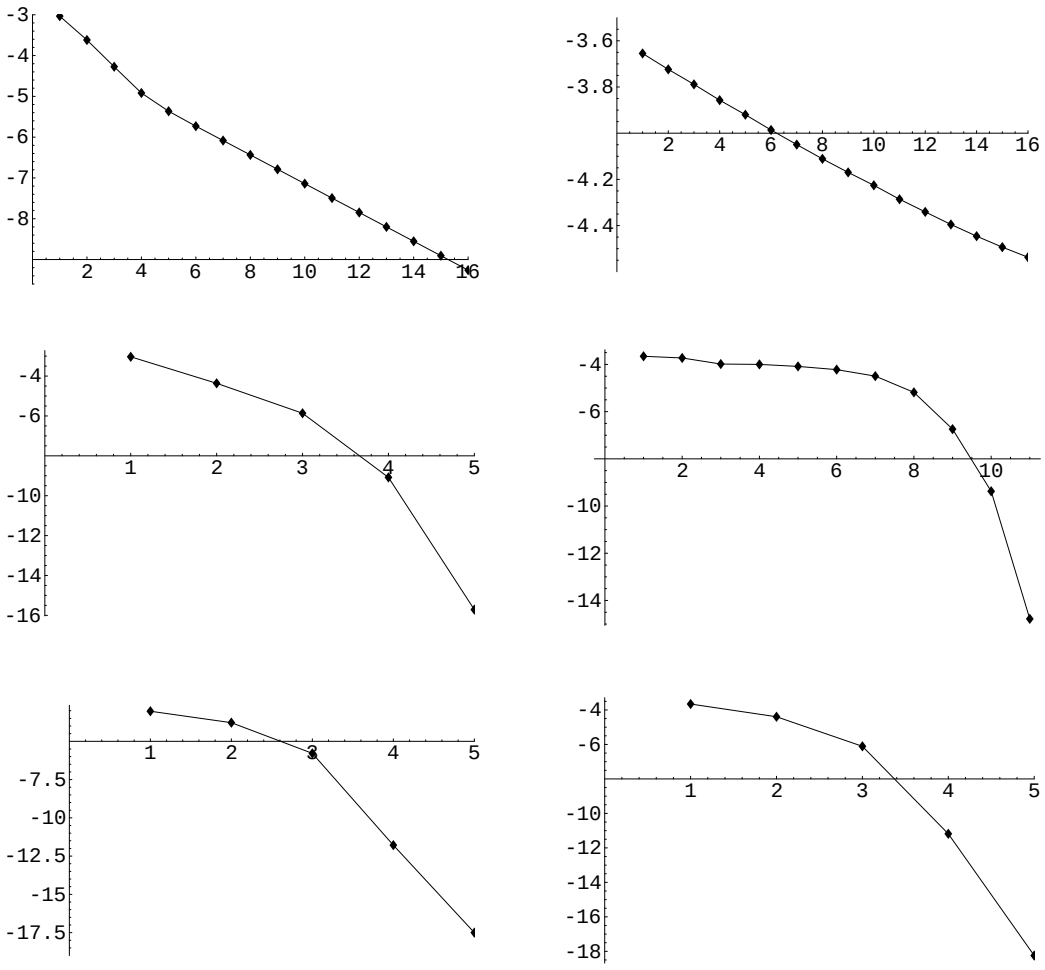

Figure 5.2: Sample convergence histories for three different algorithms on two problems with differing coefficients. The diagrams on the left are for constant coefficients; the diagrams on the right are for the permeability field shown in Figure 5.1. The diagrams in the top row are for the Jacobi-like algorithm, those in the second row for the naive Newton, and those in the last row for the geometric Newton. In each diagram, the base-10 logarithm of the l_2-norm of the error is plotted versus the iteration number. There are notable differences in the scales used, though.

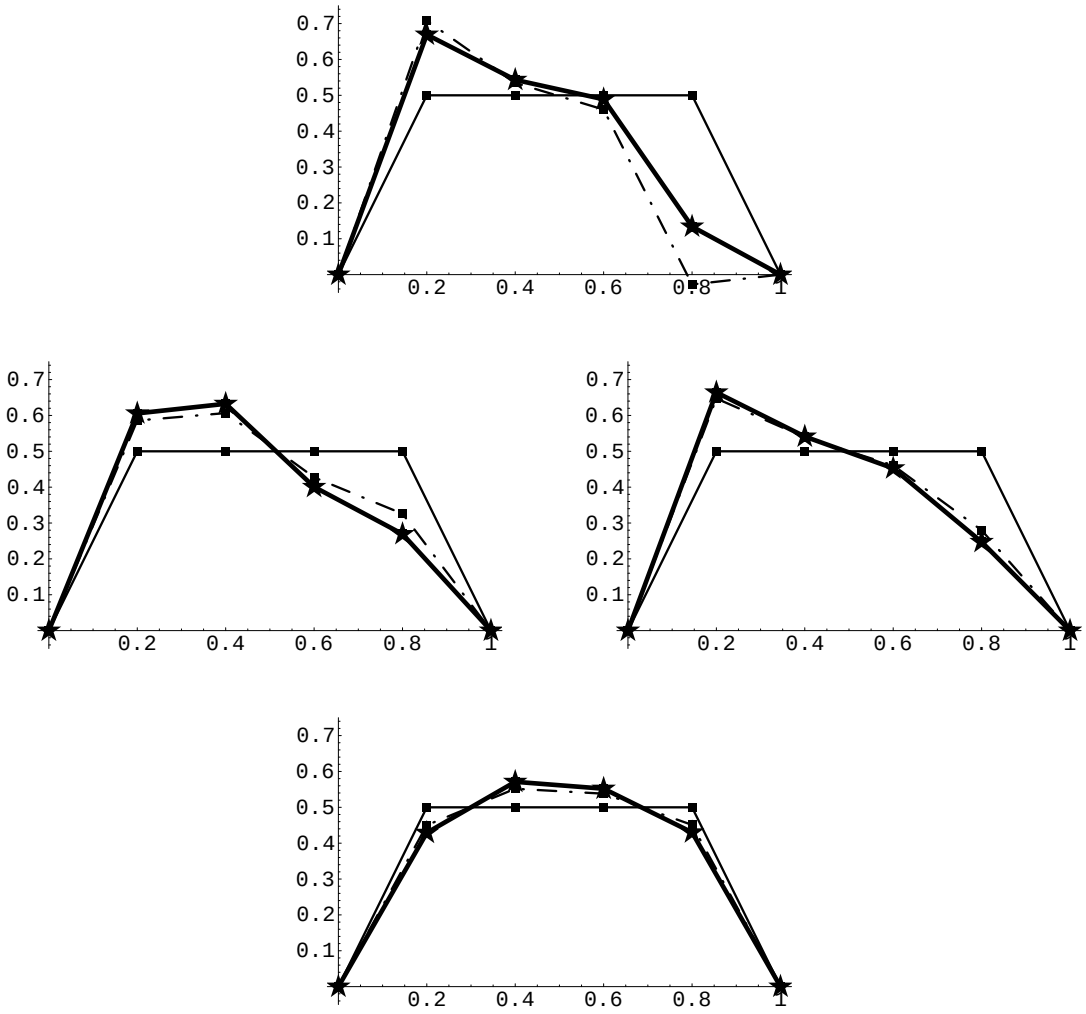

Figure 5.3: Sample convergence histories of edge shapes for the geometric algorithm on the heterogeneous coefficients. A plot for each of the four coarse edges is shown. The thin line is the initial shape, the dotted line is the shape after one iteration, and the solid line is the solution shape. (The second and later iterates are indistinguishable from the solution.)

of the discretization. This property of asymptotic mesh independence was noticed as early as 1978 by McCormick in [60]. See also the bibliographic note in [27, p377] or the review in [82].[3]

Figure 5.4 shows that the number of iterations seems independent of the resolution when using the naive Newton method applied to the constant coefficients problem. (The geometric method was not tested for the constant coefficient problem because of its excellent behavior with much more heterogeneous problems.) Both the naive Newton and the geometric Newton methods were applied to the heterogeneity field shown in Figure 5.5; the results of these computations are shown in Figure 5.6. The naive Newton method seems relatively insensitive to the resolution for the heterogeneous coefficients; at worst, there is a weak trend. The geometric method, on the other hand, seems completely insensitive to the resolution.

The data shown below in Figure 5.9 for the geometric method as applied to the channel/barrier permeability field (Figure 5.8) also demonstrate insensitivity to the resolution $1/h$.

5.3 Heterogeneity insensitivity

The number of iterations required by the two Newton methods appears to be insensitive to the heterogeneity of the permeability k. As mentioned in Section 6, this seems to be a

[3]The asymptotic mesh independence mentioned in these articles applies to nonlinear partial differential equations. This contrasts with the linear equation studied here. Also different is that the algorithms applied to nonlinear equations are not guaranteed global convergence whereas our (exact) algorithm is.

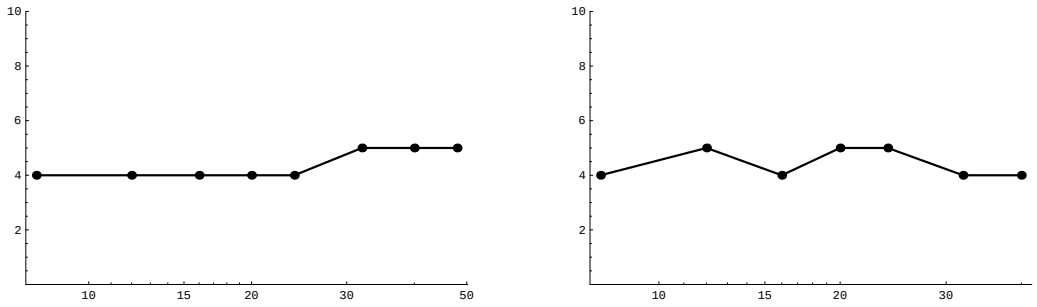

Figure 5.4: The damped naive Newton method was applied to a quarter five-spot flow problem with constant coefficients. In one case (shown at left), the coarse grid was left fixed as the fine grid and subgrid were refined. In the other case (shown at right), the subgrid was left fixed as the fine and coarse grids were refined. In each diagram, the number of iterations needed for convergence to a fixed tolerance is shown on the vertical axis. The resolution of the fine grid $1/h$ is shown on the horizontal axis.

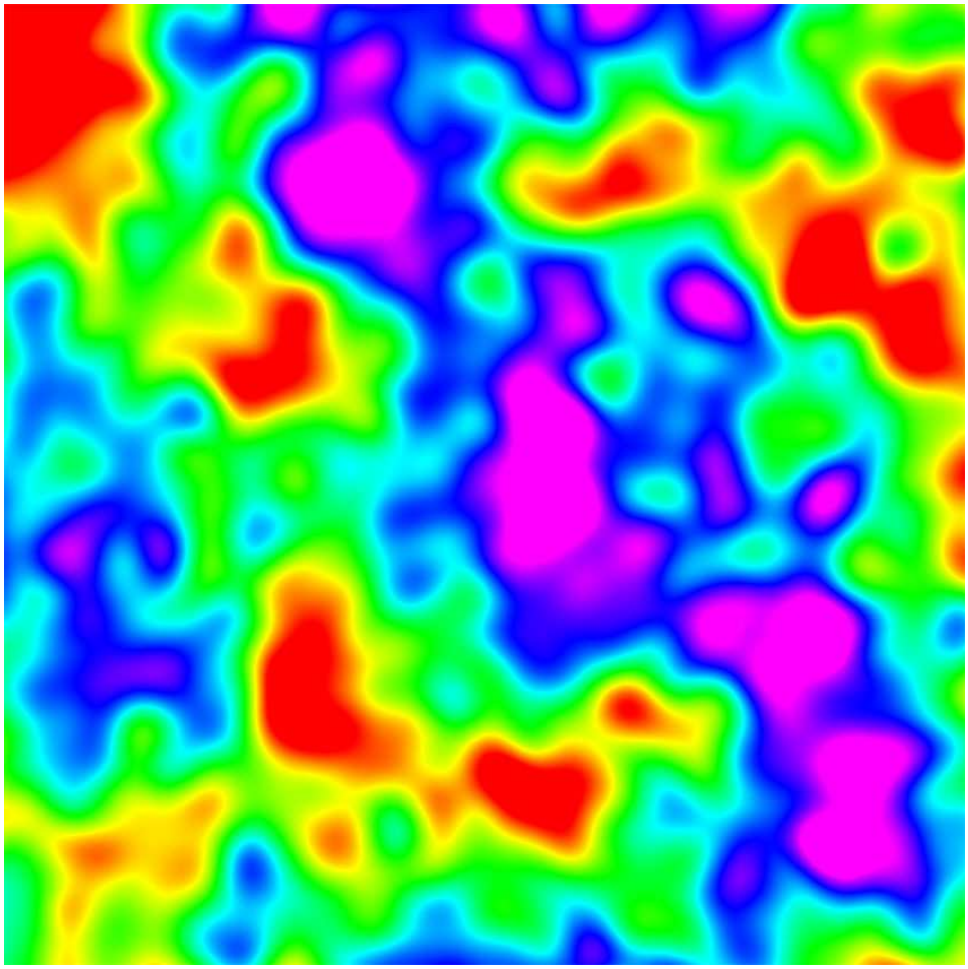

Figure 5.5: A sample permeability field. The base-10 logarithm is plotted; blue areas indicate high permeability, and red low permeability. The permeability was simulated using a correlated Gaussian random field at a resolution of 1680×1680. There were three (stable) semi-variogram structures: one with a correlation length 40% of the field width, one 15% of the field width, and another 5% of the field width; the first two had triple the variance of the last. Overall, the mean permeability is 100 mD, and the standard deviation is 258 mD; the minimum is 0.037 mD, the 5th percentile is 1.08 mD, the median is 24.7 mD, the 95th percentile is 411 mD, and the maximum is 3780 mD. Statistical subsamples were used to generate lower resolution versions for subsequent tests (the field was divided into blocks, and a pixel picked uniformly at random from the block as representative).

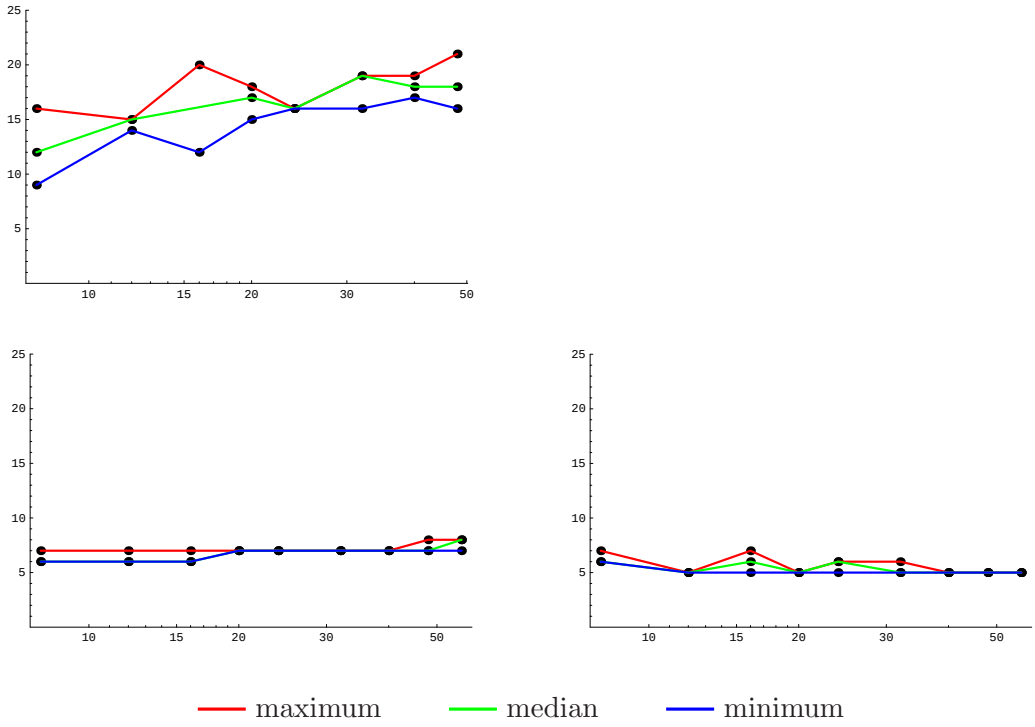

Figure 5.6: As in Figure 5.4, the number of iterations required is plotted as a function of h. However, heterogeneous coefficients subsampled from the field shown in Figure 5.5 are used in place of constant coefficients. As before, the fixed H case is shown at left, and the fixed H/h case at right. The results for the damped naive Newton method are shown in the first row, and the geometric Newton method in the second. At each resolution several subsamples were taken from the permeability field. The minimum number of iterations required for convergence is plotted in blue, the median number in green, and the maximum number in red.

rather unique property. As with the insensitivity to the resolution, we have not been able to prove this, but the computational evidence is compelling (at least for the geometric method — see the figures below). It also comports with the fact that the projection operators $I - Z_\beta$ and Z_β can be bounded independently of A (or the condition number of A) so long as A is symmetric, positive definite, and diagonally dominant [76, 66, 33]. How exactly this relates to the problem at hand remains to be discovered.

As a first test of the sensitivity of the algorithms to the degree of heterogeneity in the coefficients, we took the permeability field of Figure 5.5 and rescaled it to generate a range of fields with differing heterogeneity. That is, we took the field and linearly rescaled its logarithm so that the ratio of maximum to minimum values was a specified value. Figure 5.7 shows the results of these computations. Note the scale on the horizontal axes: the logarithm of the maximum to mimimum values is plotted. Zero on the left corresponds to constant coefficients; to the right, the ratio ranges up to 10^{12}!

For the naive Newton method there may be a trend. It is difficult to tell, though; the data for $h = 1/8$ and $h = 1/12$ seem to flatten out to the right. An additional difficulty was the tuning necessary to get the algorithm to converge; see the following section on initial damping for further details. On the other hand, for the geometric Newton's method it is clear there is no trend. The number of iterations required is fixed regardless of the level of heterogeneity. (Also recall there is no tuning done for this method.)

It is a problem of engineering interest to examine the sensitivity of simulated flows to a changing variogram. This set of tests leads to the educated guess that the performance of the algorithm is insensitive to the correlation length(s) inherent in the permeability field. That is, we varied the resolution while leaving the (physical) field fixed. The lengths that the algorithm saw depended on the resolution but had no effect on convergence. On the other hand, some upscaling and multiscale techniques experience "resonance" effects between the scale of coarsening and the scale of correlation lengths in the permeability field. Some further exploration is warranted.

As a second test of the sensitivity to heterogeneity, we generated an artificial two-level permeability field. It is shown in Figure 5.8; the ratio of the permeability in the two colored regions in the diagram was varied. When the permeability of the red region was less than that of the grey, this is a barrier problem; when the permeability of the red is higher, this is a channel problem. Figure 5.9 shows the results of these computations. Again note the scale on the horizontal axes: the middle of the diagrams (at zero) corresponds to constant coefficients. At the extreme left and right of the diagrams the ratio of maximum to mimimum ranges up to 10^{10}!

Again, for the naive Newton method there may be a trend. It is difficult to tell, though; the data seem to flatten out to the right. Tuning problems may have caused the

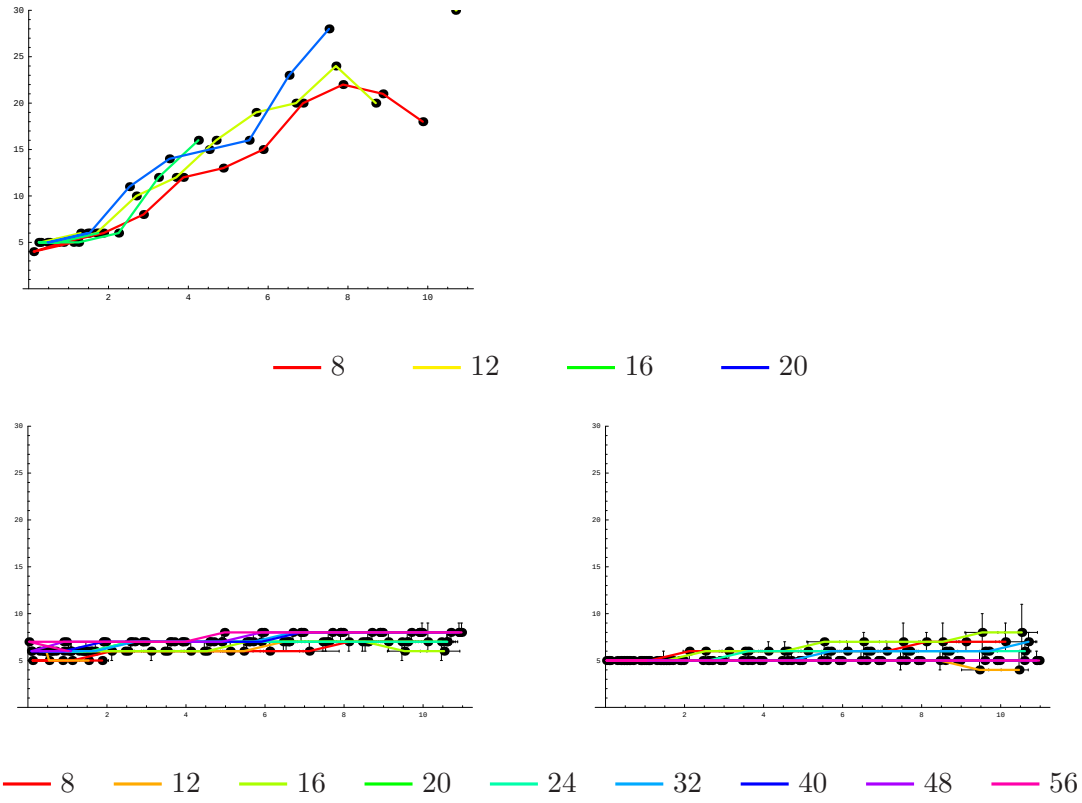

Figure 5.7: The number of iterations required is plotted versus the base-10 logarithm of the ratio of maximum to minimum permeability. At each resolution several subsamples were taken from the permeability field of Figure 5.5; the logarithm of the sample was then rescaled linearly to generate a range of fields with differing ratios of maximum to minimum permeability. In the lower diagrams, error bars on a given data point represent the range of permeabilities of the subsamples and the maximum and minimum number of iterations needed. As before, the fixed H case is shown at left, and the fixed H/h case at right. The results for the damped naive Newton method are shown in the first row, and the geometric Newton method in the second. Color labels the varying resolutions in $1/h$.

two high outliers on the $h = 1/12$ curve. To the left it is difficult to tell because of the incomplete data.

On the other hand, the geometric Newton's method again has no difficulty with the large jumps in the permeability. The number of iterations required is fixed regardless of the level of heterogeneity.

5.4 Sensitivity of initial damping ω_0 for the naive Newton method

As noted, for the damped naive Newton's method it is necessary to specify (an estimate/guess for) the initial damping ω_0 as a parameter to the algorithm. The algorithm can sometimes adapt to a poor choice and estimate a better one; sometimes it cannot, and the algorithm fails to converge. If the initial estimate works well enough in reducing the objective, the algorithm leaves it alone.

When running the heterogeneity sensitivity experiment from the section above, we recorded the initial damping factors that were ultimately used in the first step the algorithm took. In all cases, the initial damping was initially set at $\omega_0 = 0.2$. Plots of the recorded data are shown in Figure 5.10.

There is a fairly clear trend for increasing heterogeneity in the permeability: the initial damping must be set smaller as the heterogeneity increases. Small damping indicates a highly nonlinear problem. (And note small damping means small steps; we might expect more steps to be taken.) There also appears to be a weak trend downward for decreasing h

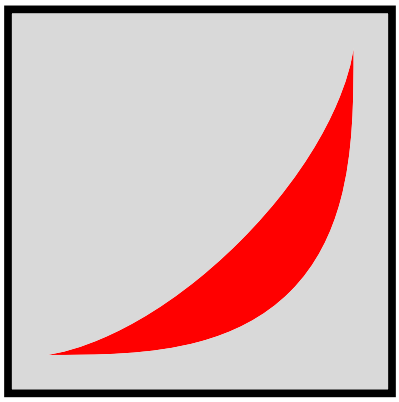

Figure 5.8: A two-level permeability field. The gray represents one permeability, the red another. When the red permeability is higher than the gray, there is a channel. When the gray is higher, there is a barrier. The permeability was integrated exactly in the matrix assembly for the fine-scale problem.

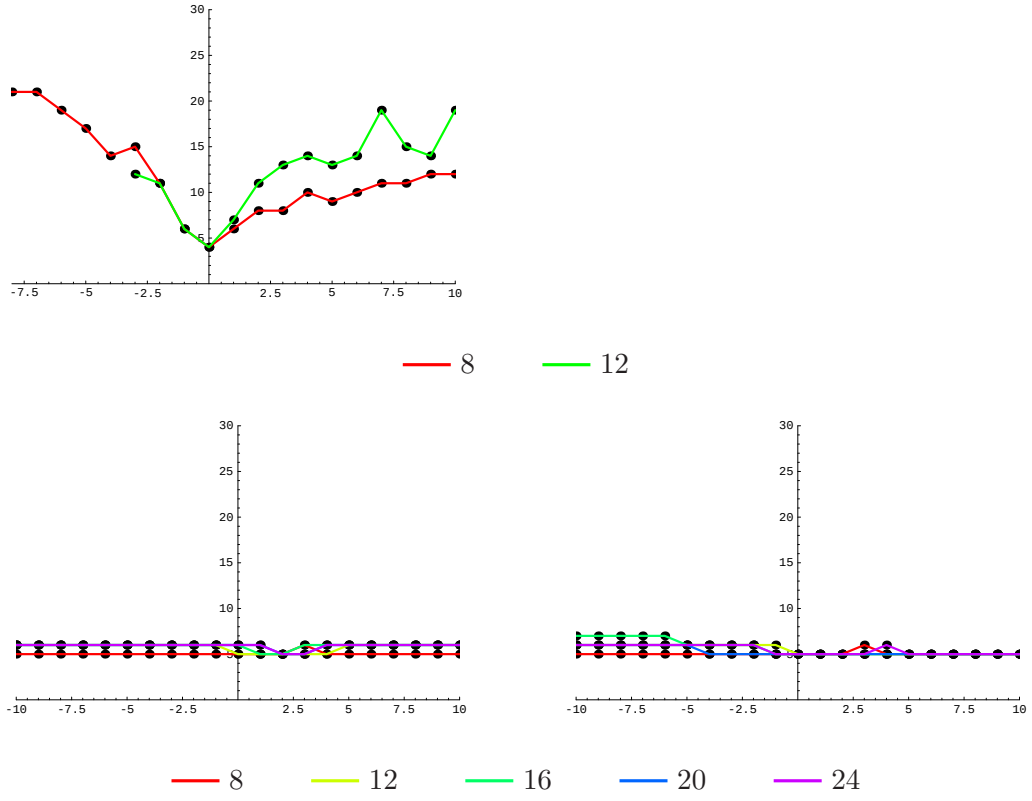

Figure 5.9: The number of iterations required is plotted versus the base-10 logarithm of the ratio of permeabilities from Figure 5.8. As before, the fixed H case is shown at left, and the fixed H/h case at right. The results for the damped naive Newton method are shown in the first row, and the geometric Newton method in the second. Color labels the varying resolutions in $1/h$.

(increasing resolution).

There is a plateau in the data at $\omega_0 = 0.2$ because that is the level of the initial guess for the initial damping. For some levels of heterogeneity this was good enough, and the algorithm left it alone. There is also a plateau at $\omega_0 = 1$; that indicates a full (undamped) Newton step was taken. It is no surprise that this only occurs at or near constant coefficients.

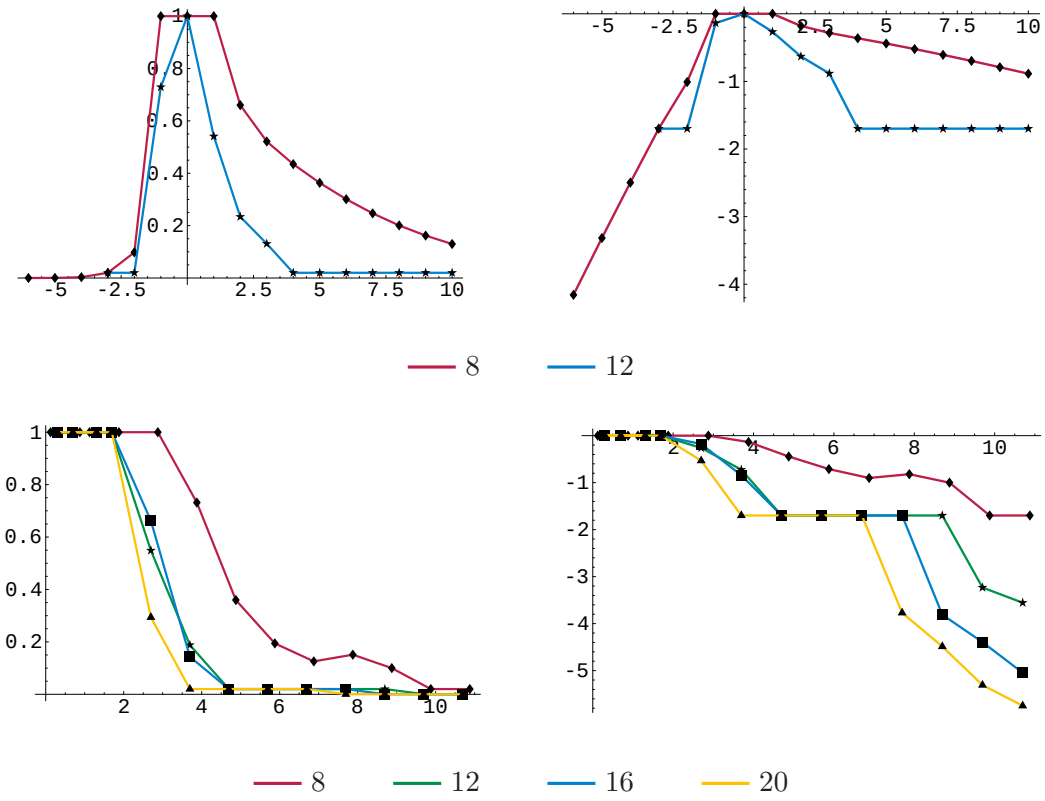

Figure 5.10: The initial damping ω_0 appears to be sensitive to the size of the jumps in the permeability. Plotted above are estimates for the initial damping versus the base-10 logarithm of the maximum to minimum ratio of the permeability. The plots in the top row are for the channel/barrier permeability field of Figure 5.8; those in the second row are for the scaled sample permeability field of Figure 5.5. The plots on the left have ω_0 on the vertical scale; those on the right have $\log_{10} \omega_0$. The labels for the colors are $1/h$. The coarse grid size H was fixed for these experiments.

The above diagrams actually reflect data from a mixed element implementation (with lowest-order Raviart–Thomas elements) of the same problem. Presumably, similar results hold for Galerkin elements.

Chapter 6

Other similarly-featured algorithms for "linear" problems

Many other researchers in multiscale approximation have attempted to modify their coarse basis shapes (form coarse macro elements) to obtain a better multiscale approximation to a fine-scale problem. For instance, see the review in [30], or the papers [44, 45, 17, 31, 84, 7, 5, 6, 14, 15, 16, 48, 1, 2]. However, these are all one-shot calculations: a single collection of basis shapes is calculated for a given right-hand side, the multiscale approximation using these shapes is calculated, and the process stops there. (At least within a single time step in a parabolic problem.) On the other hand, our algorithm continues adjusting (refining) these shapes using feedback via the residual.

Current iterative algorithms for linear problems — such as multigrid and Krylov methods — all have linear convergence. Multigrid generally does perform independent of the resolution, but conjugate gradients with a typical smoother does not. On the other hand, the performance of both tends to degrade in the face of strong heterogeneities (jumps in the coefficients, ill-shaped elements, et cetera). In contrast, as we detail below, there are algorithms that get quadratic convergence on problems that are (ostensibly) linear, and at least one algorithm that gets convergence at a rate independent of jumps of the coefficients in a problem. However, significant differences exist between these problems and algorithms and ours.

6.1 Quadratic convergence on linear problems

Solving linear systems involves nonlinear operations, namely, division. At first blush, one might expect that iterative algorithms for solving linear systems can achieve superlinear convergence since Newton's method does for nonlinear ones. However, only a handful of

algorithms for linear problems have this property. We detail three kinds here: two for solving linear systems of equations (one is specific to Toeplitz systems), and one for finding the minimizer of the norm of the residual of an over-determined linear system.

General linear systems

In 1933, Schulz [75] described the iteration $R \leftarrow R + R(I - AR)$ where R is an approximate inverse of A. This iteration has quadratic convergence; for instance, it is Newton's method applied to the objective $f(R) = R^{-1} - A$. It also has the feature that updates to R only require multiplication and addition. However, this iteration suffers from a number of problems in applying it in practice to the solution of linear systems. One such problem is that of fill-in: if A is sparse, and even if initially R is sparse, then after a few iterations R will not be sparse. Another problem is that the initial approximation R must be sufficiently close (like for Newton's method). In 1996, Brezinski [13] gave some more abstract sufficient conditions for such an matrix update to have superlinear convergence (either the matrix R to A^{-1} or as applied to a right-hand side through the norm of the residual going to zero); however, he gave only the iteration above as an example that satisfied those conditions.

Our method is different in that we do not strive to compute the inverse (which would allow us to solve a linear system with any right-hand side). We focus on changing the matrix A_β so that we get the solution for one right-hand side only (like most iterative methods do). That is, instead of a subspace correction to a solution, we attempt to modify the subspace V_β so the subspace "correction" is perfect. This subspace will almost surely be different for different right-hand side data.

Schulz's iteration can be modified so that one calculates the effect of the (approximate) inverse on a single right-hand side; however, it cannot be computed recursively without computing the update to the matrix R as well (requiring its storage). It could be an efficient procedure if one wanted to compute the solution for many right-hand sides, say, a number proportional to the number of unknowns.

There are other fixed point iterations with quadratic convergence, but they too suffer from the same difficulties as Schulz's method. For instance, Beavers and Denman in [9], and Mastroserio and Montrone in [59] both describe quadratically convergent schemes for the inverse. Mastroserio and Montrone's requires only multiplications and additions. (Both papers solve a more general Lyapunov system, but Beavers and Denman assume the inverse of a linear system is an available operation. With the modification of Hoskins, Meek, and Walton [42, 43] with $B = 0$, their method can be used to solve linear systems.)

Toeplitz systems

A Toeplitz matrix has entries that are constant along diagonals. This special structure can be exploited in developing solvers for Toeplitz systems. For instance, there are iterative solvers for Toeplitz systems that achieve quadratic convergence such as those of Brent, Gustavson, and Yun in [11]; and Linzer and Vetterli in [58]. The method described in these two papers is a little simpler for symmetric systems; we detail it here.

A symmetric Toeplitz matrix A can be specified by just its first column. If A is non-singular, its inverse A^{-1} is also a symmetric Toeplitz matrix. If we wish to solve the $n \times n$ system $Ax = y$, we can calculate $x = A^{-1}y$ using $O(n \log n)$ operations if we know the first column c of A^{-1}. This is possible through the FFT-like property of Toeplitz matrix-vector multiply and the Gohberg–Semencul formula. This formula says:

$$A^{-1} = \frac{1}{b_0}\left(L\begin{pmatrix} b_0 \\ \boldsymbol{b} \end{pmatrix} L^T \begin{pmatrix} b_0 \\ \boldsymbol{b} \end{pmatrix} - L\begin{pmatrix} 0 \\ \tilde{\boldsymbol{b}} \end{pmatrix} L^T \begin{pmatrix} 0 \\ \tilde{\boldsymbol{b}} \end{pmatrix}\right)$$

where $L(\boldsymbol{z})$ is the lower triangular Toeplitz matrix with first column $\boldsymbol{z}$, the vector $\boldsymbol{c} = (b_0, \boldsymbol{b}^T)^T$ is the first column of A^{-1}, the scalar b_0 is the first entry of $\boldsymbol{c}$, the vector $\boldsymbol{b}$ represents the other entries of $\boldsymbol{c}$, and the vector $\tilde{\boldsymbol{b}}$ has the components of $\boldsymbol{b}$ listed in reverse order. Thus, once we know $\boldsymbol{c}$, we can calculate $x = A^{-1}y$ using $O(4n \log n)$ operations.

We can calculate c by solving the system $Ac = e_1$ using an iterative method. In the Gohberg–Semencul formula, if we have an approximation $(\xi_0, \boldsymbol{\xi}^T)^T$ for $\boldsymbol{c}$, we can write an approximate inverse as

$$A^{-1} \approx N\begin{pmatrix} \xi_0 \\ \boldsymbol{\xi} \end{pmatrix} = \frac{1}{\xi_0}\left(L\begin{pmatrix} \xi_0 \\ \boldsymbol{\xi} \end{pmatrix} L^T \begin{pmatrix} \xi_0 \\ \boldsymbol{\xi} \end{pmatrix} - L\begin{pmatrix} 0 \\ \tilde{\boldsymbol{\xi}} \end{pmatrix} L^T \begin{pmatrix} 0 \\ \tilde{\boldsymbol{\xi}} \end{pmatrix}\right).$$

Then a Jacobi-like algorithm is:

1. Pick a guess-timate for c;
2. Calculate the residual $r = e_1 - Ac$;
3. Update $c \leftarrow c + N(c)r$; and
4. Repeat until the residual is small.

The key is that as the solution vector c is improved we can use it to improve our estimate of A^{-1}. This feedback gives quadratic convergence to the solution as opposed the usual linear convergence where our estimate of A^{-1} is fixed. This phenomenon is similar to our algorithm where we use our solution $I_\beta u_\beta$ (via its residual r_β) to update the problem A_β we solve.

Overdetermined linear systems with l_1 or l_∞ minimization

Suppose we wish to solve the minimization problem $\text{argmin}_u \|f - Au\|$ where the linear system $Au = f$ is overdetermined, and where we use either the l_1 norm or the l_∞ norm to measure the residual. There are algorithms that find a sequence of u_k that converge quadratically to the solution u_∞; see, for instance, the results by Coleman and Li in [19, 20]. This is not too surprising since it is known that this type of minimization problem is equivalent to solving a linear program.[1] Since the development of the ellipsoid algorithm by Khachiyan [55], many so-called "interior point" iterative algorithms for solving linear programs have been developed all of which produce a sequence that converges quadratically to the solution. There are, for instance, the Karmarkar [53], Mehrotra [61], and Mizuno–Todd–Ye algorithms [88, 89, 86, 87]. See the textbooks [70, 83, 85] or the reviews [36, 69, 68] for more information.

These algorithms get quadratic convergence and are insensitive to the size of the system (number of unknowns), but require a linear solve at each step of a size usually near or at the number of unknowns in size (the number of "active" constraints). That is, applied to solving $Au = f$ they would terminate in one step by computing $u = A^{-1}f$ directly.

Also, these algorithms may be sensitive to the condition number of the system solved during the step. However, the work of Vavasis on equilibrium systems in [79] (related to the work by Stewart, O'Leary, and Forsgren referenced above in Section 5.3) can be applied to eliminate some of this sensitivity (the sensitivity, at least, to the duality gap — the proximity of the current iterate to the feasible region boundary).

6.2 Insensitivity to jumps

With regards to insensitivity to jumps in the coefficients of a problem, the author is only aware of one other algorithm that has such a property. Tausch and White in [78] describe an electromagnetics problem where there are dielectric materials next to conductors. This results in high contrasts of the permittivity ratio on short length scales.

Using the equivalent charge formulation, one effectively computes a polarization charge across the dielectric and a conductor charge. Because of the high ratio in permittivity, these two can differ by many orders of magnitude; the simultaneous calculation of the differently scaled quantities is numerically challenging. The algorithm proceeds by separating the problem of interest in two stages. In the first stage the dielectric is replaced with one of infinite permittivity; this allows for a fairly accurate computation of charges on

[1] Both papers describe how to re-write a minimization problem as a linear program; both papers also offer the reasonable advice that an algorithm tailored for a minimization problem probably will fare better than a general linear programming algorithm.

the surfaces between the conductors and the dielectrics. The following second stage solves a perturbation problem where the dielectric constant is reduced from infinity to a finite value. The result is an algorithm where work and accuracy are bounded independent of the contrast in permittivity.

There is a difference, however, between this problem and ours. Our problem is challenging because it is ill-conditioned. The above electromagnetics problem suffers from poor scaling. Quoting Tausch and White: "This ... makes clear that the accuracy problem of the equivalent charge formulation is not ill-conditioning but a scaling problem."

Chapter 7

Conclusions and future directions

We have introduced an iterative algorithm for solving large, sparse, ill-conditioned linear systems. Implemented exactly, the algorithm has monotone, global convergence with a quadratic asymptotic rate. Several of the steps in the algorithm are computationally infeasible; first among them is the need for an error estimate at each iteration. Assuming such an estimate is provided, some computational experience indicates that the other approximations we have introduced do not affect the convergence properties of the exact algorithm. As implemented, each iteration is computationally cheap, and convergence is fast.

The very next step in our work will be to test our algorithm as an accelerator. As mentioned before, it seems reasonable to pair our algorithm with a smoother. Established results on inexact Newton methods tell us the error tolerance on the inner iteration needs to get tighter as outer iterations progress in order to maintain superlinear or quadratic convergence. The important practical question, though, is: how many inner iterations (smoothings) does it take to achieve that tolerance? It would need to be a fixed or slowly increasing number for our algorithm to be viable. In tandem with computational experiments would be theoretical work to establish the properties of our algorithm as an accelerator.

Our computational experience also indicated that, as a stand-alone method, the algorithm is insensitive to the resolution and heterogeneity of the problem to be solved. Some theoretical notions also support this idea, but we have yet to produce a proof. The insensitity to heterogeneity apparently is a peerless property of our algorithm. Computational experience has also shown that the other approximations introduced to made the algorithm feasible do not impact performance. Some theoretical footing for this observation would be welcome. Further study is also needed to determine the dependence on the size of the coarse problem (or the level of upscaling).

Even though we solve a simple linear system, our research benefits Darcy flow modeling because, in problems of practical interest, permeability often is given at high-resolution

and is heterogeneous. The insensitivity of our algorithm to this ill-conditioning means that a wider variety of problems is computationally feasible. Further benefits accrue when considering more sophisticated models than single-phase, steady-state flow. In a time dependent problem with an implicit time stepping scheme, one solves a linear system at each time step. In optimizing the production from a reservoir, one wants to pick the best well placement, injection and pumping rates, and so on; this is an inverse problem which requires solving a number of forward problems (ours). The permeability fields used in flow simulations are often simulated themselves; to get a sense for the statistical propertities of quantities predicted from the flow field, one often computes flows for an ensemble of many realizations of permeability fields (each of which requires its own flow simulation). Nonlinear flow problems must be linearized at each solution iteration; problems like ours result. In each context above, we want to solve many, many linear systems just like ours as quickly as possible. Shortening the time it takes to do so would be an important advance (even if we contribute nothing directly to each of these interesting and important problems).

7.1 Possible improvements

Recursion (a multilevel method) is a natural generalization. This can be accomplished by aggregating coarse basis shapes into a super-coarsened system the same way that fine-scale shapes were. There are at least two ways to incorporate this into the algorithm. In the first case, we simply incorporate more degrees of freedom into the nonlinear side of the problem. This expense is offset somewhat by reducing the size of the coarse linear system to be solved at each iteration. In another strategy, one could apply our algorithm as an iterative solver to the coarse problem solve required in each iteration. This way the extra degrees of freedom on the nonlinear side are solved for in stages instead of all at once. Which strategy reduces the overall solution time while keeping robustness is an interesting question to investigate. As an aside, the coarse system resulting from our two-level procedure is denser (more off-diagonal terms) than the original fine system; however, using recursion to solve this system results in no further increases in density.

The coarse system that we solve at each iteration has more degrees of freedom than the usual multiscale's coarse system (about three times as many in 2-D and seven times as many in 3-D for rectangular grids). We can combine the three spaces V_{corner}, $V_{\beta,\text{edge}}$, and $V_{\beta,\text{face}}$ for computational efficiency in the coarse solves. That is, the shape parameters can be used to determine corner shapes alone. However, this reduction of computation comes at the expense of analytical problems. For instance, the overlapping shape parameters mean the ranges of the V_{e_c} are no longer independent. Additionally, if the shapes along every coarse edge of a coarse patch are set so that there is no net flux through each edge, then the

coarse system will be singular — there is a Neumann-like problem implied on said coarse patch.

In the mixed case, this is comparable to having shape parameters all along a coarse edge. (Whereas, in a parallel to the continuous Galerkin elements, there would be a unit flow all along an edge combined with edge "bubbles" with no net flow.) Computationally, we can leave out the additional elements and adjust the full edge so long as the normalized shape has a net flow bigger than a fixed tolerance (and switch back when the edge-normalized coarse solution is bigger than the tolerance).[1] There are no problems with overlapping with mixed elements.

In the matter of initialization, we noted theoretically the almost sure global convergence and suggested practically the naive initialization of uniform shapes ($\beta = 1$). We cannot use the zero shapes ($\beta = 0$) as we know this is a fixed point of our Newton iteration. Although it is possible that uniform shapes will result in a poor start, as a practical matter multiscale algorithms have already shown themselves to produce a very good approximation to the fine-scale solution. Save for an odd source term, indeed, we are likely to already start very close to the solution. As was noted in chapter 6, though, other researchers have tried to improve on their multiscale solution procedures. One improvement that could be readily adapted to give us a better initialization would be to use the implied edge shapes of the coarse macro-elements of [1].

7.2 Extensions to other problems

As a trivial extension, we note that symmetry of the system is not needed; a positive definite symmetric part will do.

Substantial work has been done to establish these methods for mixed elements, in particular, lowest-order Raviart–Thomas elements. Indeed, the mixed element formulation was our original inspiration. Like with the continuous Galerkin elements, this method can be expressed in an algebraic fashion; we believe it can be applied to any equilibrium system.

We conjecture that our method can be applied to other discretizations, too, such as discontinuous Galerkin elements, cell-centered finite differences, and finite volume methods. Higher order finite element methods seem ripe as well by optimizing shapes that have control points on the boundary of elements. We expect that our techique can be applied to other partial differential equations such as for Stokes flow or linear elasticity.

[1] This, of course, raises the question of how to set the tolerance appropriately. However, for normalized shapes, it is not some nebulous quantity but a simple percentage of the magnitude of the shape.

Bibliography

[1] J. E. Aarnes, *On the use of a mixed multiscale finite element method for greater flexibility and increased speed or improved accuracy in reservoir simulation*, Multiscale Modeling and Simulation, 2 (2004), pp. 421–439.

[2] J. E. Aarnes, S. Krogstad, and K.-A. Lie, *A hierarchical multiscale method for two-phase flow based upon mixed finite elements and nonuniform coarse grids*, Multiscale Model. Simul., (2006, to appear).

[3] R. L. Adler, J.-P. Dedieu, J. Y. Margulies, M. Martens, and M. Shub, *Newton's method on Riemannian manifolds and a geometric model for the human spine*, IMA Journal of Numerical Analysis, 22 (2002), pp. 359–390.

[4] B. Aksoylu, H. Klie, and M. Wheeler, *Physics-based preconditioners for porous media flow applications*, In preparation.

[5] T. Arbogast, *Analysis of a two-scale, locally conservative subgrid upscaling for elliptic problems*, SIAM J. Numer. Anal., 42 (2004), pp. 576–598.

[6] T. Arbogast and K. J. Boyd, *Subgrid upscaling and mixed multiscale finite elements*, SIAM J. Numer. Anal., 44 (2006), pp. 1150–1171.

[7] T. Arbogast, S. E. Minkoff, and P. T. Keenan, *An operator-based approach to upscaling the pressure equation*, in Computational Methods in Water Resources XII, Vol. 1: Computational Methods in Contamination and Remediation of Water Resources, V. N. Burganos et al., eds., Southampton, U.K., 1998, Computational Mechanics Publications, pp. 405–412.

[8] J. Bear, *Dynamics of Fluids in Porous Media*, Dover, New York, 1972.

[9] A. N. Beavers, Jr. and E. D. Denman, *A new solution method for the Lyapunov matrix equation*, SIAM Journal on Applied Mathematics, 29 (1975), pp. 416–421.

[10] A. Ben-Israel and T. N. E. Greville, *Generalized Inverses: Theory and applications*, Springer-Verlag, New York, second ed., 2003.

[11] R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun, *Fast solution of Toeplitz systems of equations and computation of Padé approximations*, Journal of Algorithms, 1 (1980), pp. 259–295.

[12] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation (α-SA) multigrid*, SIAM Review, 47 (2005), pp. 317–346.

[13] C. Brezinski, *Variations on Richardon's method and acceleration*, in Numerical analysis: a numerical analysis conference in honour of Jean Meinguet, 1996, pp. 33–44.

[14] F. Brezzi, *Interacting with the subgrid world*, in Numerical Analysis, 1999, Chapman and Hall, 2000, pp. 69–82.

[15] F. Brezzi, T. Hughes, L. Marini, A. Russo, and E. Süli, *A priori error analysis of residual-free bubbles for advection-diffusion problems*, SIAM J. Numer. Anal., 36 (1999), pp. 1933–1948.

[16] F. Brezzi, L. Marini, and E. Süli, *Residual-free bubbles for advection-diffusion problems: the general error analysis*, Numer. Math., 85 (2000), pp. 31–47.

[17] Z. Chen and T. Y. Hou, *A mixed multiscale finite element method for elliptic problems with oscillating coefficients*, Math. Comp., 72 (2003), pp. 541–576.

[18] S. Chippada, C. Dawson, M. L. Martínez-Canales, and M. Wheeler, *A projection method for constructing a mass conservative velocity field*, Computer Methods in Applied Mechanics and Engineering, 157 (1998), pp. 1–10.

[19] T. Coleman and Y. Li, *A global and quadratically convergent method for linear l_∞ problems*, SIAM Journal on Numerical Analysis, 29 (1992), pp. 1166–1186.

[20] ——, *A globally and quadratically convergent affine scaling method for linear l_1 problems*, Mathematical Programming, 56 (1992), pp. 189–222.

[21] H. Darcy, *The Public Fountains of the City of Dijon*, Victor Dalmont, Paris, 1856, ch. Appendix D.

[22] B. L. Darlow, R. E. Ewing, and M. F. Wheeler, *Mixed finite element methods for miscible displacement problems in porous media, SPE 10501*, Soc. Petrol. Engr. J., 24 (1984), pp. 391–398.

[23] J.-P. Dedieu and M. Shub, *Newton's method for overdetermined systems of equations*, Mathematics of Computation, 69 (2000), pp. 1099–1115.

[24] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, no. 16 in Classics in Applied Mathematics, SIAM, Philadelphia, 1996.

[25] P. Deuflhard, *A modified Newton method for the solution of ill-conditioned systems of nonlinear equations with application to multiple shooting*, Numerische Mathematik, 22 (1974), pp. 289–315.

[26] ——, *A relaxation strategy for the modified Newton method*, in Optimization and optimal control, R. Bulirsch, W. Oettli, and J. Stoer, eds., vol. 477 of Lecture Notes in Mathematics, 1975, pp. 59–73.

[27] ——, *Newton Methods for Nonlinear Problems: affine invariance and adaptive algorithms*, Springer, 2004.

[28] C. V. Deutsch and A. G. Journel, *GSLIB geostatistical software library and user's guide*, Oxford University Press, New York, second ed., 1997.

[29] J. Douglas, Jr., R. E. Ewing, and M. F. Wheeler, *Approximation of the pressure by a mixed method in the simulation of miscible displacement*, R.A.I.R.O. Modél. Math. Anal. Numér., 17 (1983), pp. 17–33.

[30] L. J. Durlofsky, *Upscaling of geological models for reservoir simulation: Issues and approaches*, Computational Geosciences, 6 (2002), pp. 1–4.

[31] Y. R. Efendiev, T. Y. Hou, and X.-H. Wu, *Convergence of a nonconforming multiscale finite element method*, SIAM J. Numer. Anal., 37 (2000), pp. 888–910.

[32] R. E. Ewing, T. F. Russell, and M. F. Wheeler, *Convergence analysis of an approximation of miscible displacement in porous media by mixed finite elements and a modified method of characteristics*, Comput. Methods Appl. Mech. Engrg., 47 (1984), pp. 73–92.

[33] A. Forsgren, *On linear least-squares problems with diagonally dominant weight matrices*, SIAM Journal on Matrix Analysis and Applications, 17 (1996), pp. 763–788.

[34] J. Frank and C. Vuik, *On the construction of deflation-based preconditioners*, SIAM Journal on Scientific Computing, 23 (2001), pp. 442–462.

[35] R. A. Freeze and J. A. Cherry, *Groundwater*, Prentice-Hall, Englewood Cliffs, New Jersey, 1979.

[36] R. M. Freund and S. Mizuno, *Interior point methods: Current status and future directions*, Optima, 51 (1996), pp. 1–9.

[37] J. Glimm, B. Lindquist, O. McBryan, and G. Tryggvason, *Sharp and diffuse fronts in oil reservoirs: Front tracking and capillarity*, in Mathematical and Computational Methods in Seismic Exploration and Reservoir Modeling, W. E. Fitzgibbon, ed., Philadelphia, 1985, SIAM, pp. 54–76.

[38] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, third ed., 1996.

[39] W. Hahn, *Stability of Motion*, Springer-Verlag, 1967.

[40] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, Springer, second ed., 1996.

[41] P. E. Hart, D. R. Hutchinson, and M. Lee, *High-resolution multichannel seismic-reflection data acquired in the northern Gulf of Mexico*. U.S. Geological Survey Open-File Report 2005-1411, `http://pubs.usgs.gov/of/2005/1411/`, 2005.

[42] W. D. Hoskins, D. S. Meek, and D. J. Walton, *The numerical solution of the matrix equation $XA + AY = F$*, BIT, 17 (1977), pp. 184–190.

[43] ——, *High-order iterative methods for the solution of the matrix equation $XA + AY = F$*, Linear Algebra and its Applications, 23 (1979), pp. 121–139.

[44] T. Y. Hou and X. H. Wu, *A multiscale finite element method for elliptic problems in composite materials and porous media*, J. Comput. Phys., 134 (1997), pp. 169–189.

[45] T. Y. Hou, X.-H. Wu, and Z. Cai, *Convergence of a multiscale finite element method for elliptic problems with rapidly oscillating coefficients*, Math. Comp., 68 (1999), pp. 913–943.

[46] T. J. R. Hughes, G. R. Feijóo, L. Mazzei, and J.-B. Quincy, *The variational multiscale method—a paradigm for computational mechanics*, Comput. Methods Appl. Mech. Engrg., 166 (1998), pp. 3–24.

[47] D. R. Hutchinson and P. E. Hart, *Cruise report for G1-03-GM, USGS Gas Hydrates Cruise, R/V Gyre, 1–14 May 2003, northern Gulf of Mexico.* U.S. Geological Survey Open-File Report 03-474, `http://pubs.usgs.gov/of/2003/of03-474/`, 2004.

[48] P. Jenny, S. H. Lee, and H. A. Tchelepi, *Multi-scale finite-volume method for elliptic problems in subsurface flow simulation*, J. Comp. Phys., 187 (2003), pp. 47–67.

[49] L. V. Kantorovich, *The method of successive approximations for functional equations*, Acta Mathematica, 71 (1939), pp. 63–97.

[50] ——, *On Newton's method for functional equations*, Doklady Akademii Nauk SSSR, 59 (1948), pp. 1237–1240.

[51] ——, *On Newton's method*, Akademiya Nauk SSSR. Trudy Matematicheskogo Instituta imeni V. A. Steklova, 28 (1949), pp. 104–144.

[52] L. V. Kantorovich and G. P. Akilov, *Functional Analysis in Normed Spaces*, Fizmatgiz, Moscow, 1959.

[53] N. Karmarkar, *A new polynomial time algorithm for linear programming*, Combinatorica, 4 (1984), pp. 373–395.

[54] C. T. Kelley, *Iterative methods for linear and nonlinear equations*, SIAM, Philadelphia, 1995.

[55] L. Khachiyan, *A polynomial algorithm in linear programming*, Doklady Akademii Nauk SSSR, 244 (1979), pp. 1093–1096.

[56] A.-M. Li, U. Simon, and G. Zhao, *Global affine differential geometry of hypersurfaces*, De Gruyter, New York, 1993.

[57] B. Lindquist, *Geostatistically generated permeability field data.* Personal correspondence, October 1992.

[58] E. Linzer and M. Vetterli, *Iterative Toeplitz solvers with local quadratic convergence*, Computing, 49 (1993), pp. 339–347.

[59] C. Mastroserio and M. Montrone, *Un metodo a convergenza quadratica per equazioni matriciali lineari*, Calcolo, 18 (1981), pp. 87–96.

[60] S. McCormick, *A revised mesh refinement strategy for Newton's method applied to nonlinear two-point boundary value problems*, in Numerical treatment of differential equations in applications, R. Ansorge and W. Törnig, eds., vol. 679 of Lecture Notes in Mathematics, 1978, pp. 15–23.

[61] S. Mehrotra, *On the implementation of a primal-dual interior point method*, SIAM Journal on Optimization, 2 (1992), pp. 575–601.

[62] I. P. Mysovskikh, *On convergence of Newton's method*, Akademiya Nauk SSSR. Trudy Matematicheskogo Instituta imeni V. A. Steklova, 28 (1949), pp. 145–147.

[63] ——, *On the convergence of L. V. Kantorovich's method of solution of functional equations and its applications*, Doklady Akademii Nauk SSSR, 70 (1950), pp. 565–568.

[64] K. Nomizu and T. Sasaki, *Affine differential geometry: geometry of affine immersions*, Cambridge University Press, 1994.

[65] U. Nowak and L. Weimann, *A family of Newton codes for systems of highly nonlinear equations*, Tech. Rep. TR 91–10, Zuse Institute Berlin, 1991.

[66] D. P. O'Leary, *On bounds for scaled projections and pseudoinverses*, Linear Algebra and its Applications, 132 (1990), pp. 115–117.

[67] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[68] F. A. Potra, *Interior point methods: twenty years after.* `http://www.math.umbc.edu/~potra/talk0930.pdf`, September 2003.

[69] F. A. Potra and S. J. Wright, *Interior point methods*, Journal of Computational and Applied Mathematics, 124 (2000), pp. 281–302.

[70] C. Roos, J.-P. Vial, and T. Terlaky, *Theory and Algorithms for Linear Optimization: An interior point approach*, John Wiley and sons, 1997.

[71] T. F. Russell and M. F. Wheeler, *Finite element and finite difference methods for continuous flows in porous media*, in The Mathematics of Reservoir Simulation, R. E. Ewing, ed., no. 1 in Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, 1983, pp. 35–106, Chapter II.

[72] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, second ed., 2003.

[73] R. Scheichl and E. Vainikko, *Additive Schwarz and aggregation-based coarsening for elliptic problems with highly variable coefficients*, BICS Preprint, 09/06 (2006).

[74] A. E. Scheidegger, *The Physics of Flow Through Porous Media, 3*rd *ed.*, University of Toronto Press, Toronto, 1974.

[75] G. Schulz, *Iterative Berechnung der reziproken Matrix*, Zeitschrift für Angewandte Mathematik und Mechanik, 13 (1933), pp. 57–59.

[76] G. W. Stewart, *On scaled projections and pseudoinverses*, Linear Algebra and its Applications, 112 (1989), pp. 189–193.

[77] S. Sun and M. Wheeler, *Projections of velocity data for compatibility with transport*, Computer Methods in Applied Mechanics and Engineering, 195 (2006), pp. 653–673.

[78] J. Tausch and J. White, *Capacitance extraction of 3-D conductor systems in dielectric media with high permittivity ratios*, IEEE Transactions on Microwave Theory and Techniques, 47 (1999), pp. 18–26.

[79] S. A. Vavasis, *Stable numerical algorithms for equilibrium systems*, SIAM Journal on Matrix Analysis and Applications, 15 (1994), pp. 1108–1131.

[80] F. Verhulst, *Nonlinear Differential Equations and Dynamical Systems*, Springer, second ed., 1996.

[81] C. Vuik, A. Segal, and J. A. Meijerink, *An efficient preconditioned CG method for the solution of a class of layered problems with extreme contrasts in the coefficients*, Journal of Computational Physics, 152 (1999), pp. 385–403.

[82] M. Weiser, A. Schiela, and P. Deuflhard, *Asymptotic mesh independence of Newton's method revisited*, SIAM Journal on Numerical Analysis, 42 (2005), pp. 1830–1845.

[83] S. J. Wright, *Primal-dual Interior Point Methods*, SIAM, 1997.

[84] X. H. Wu, Y. Efendiev, and T. Y. Hou, *Analysis of upscaling absolute permeability*, Discrete Contin. Dyn. Syst. Ser. B, 2 (2002), pp. 185–204.

[85] Y. Ye, *Interior Point Algorithms: Theory and analysis*, John Wiley and sons, 1997.

[86] Y. Ye and K. Anstreicher, *On quadratic and $O(\sqrt{n}L)$ convergence of predictor-corrector algorithm for LCP*, Mathematical Programming, 62 (1993), pp. 537–551.

[87] Y. Ye, O. Güler, R. Tapia, and Y. Zhang, *A quadratically convergent $O(\sqrt{n}L)$-iteration algorithm for linear programming*, Mathematical programming, 59 (1993), pp. 151–162.

[88] Y. Ye, M. J. Todd, and S. Mizuno, *On adaptive-step primal-dual interior-point algorithms for linear programming*, Mathematics of Operations Research, 18 (1993), pp. 964–981.

[89] ——, *An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm*, Mathematics of Operations Research, 19 (1994), pp. 53–67.

Vita

James Rath was born in the sultry summer of '75 and has detested referring to himself in the third person ever since. He attended Jennings Elementary School beginning in the fall of 1980 and graduated in the spring of 1986. Everything since has become a bit jumbled and fuzzy; all that is known is that he emerged at the end of 2006 wielding this unwieldy dissertation.

Permanent Address:

Postal mail:
PO Box 7923
Austin, TX 78713-7923

Email:
ratjamm@alum.mit.edu

Web:
http://alum.mit.edu/www/ratjamm

This dissertation was typeset with LaTeX 2_ε[2] by the author.

[2] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. Some macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab (with corrections by John Baird).