

UNIVERSITY OF NEW MEXICO

UNDERGRADUATE HONORS THESIS

Exploration of Black Box Multigrid for Linear Systems Modeling Resistive MHD

Author:

Ari RAPPAPORT

Advisors:

Prof. Jehanzeb CHAUDHRY

Dr. John SHADID

Dr. Ray TUMINARO

April 30, 2018

“Young man, in mathematics you don’t understand things. You just get used to them.”

John von Neumann

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Numerical Solution	1
1.2.1	Discretization Methods	1
1.2.2	Linear Solvers	2
1.3	Preliminaries and Notation	2
1.3.1	Vector Operators	3
1.4	Acknowledgements	3
2	The Finite Element Method	5
2.1	Poisson's Equation	5
2.2	Analytic vs Approximate Solution	5
2.2.1	Variational Formulation	5
2.2.2	Deriving a Linear System from The Weak Form	6
2.2.3	FEM in 1D	7
2.2.4	Simplicial 2D FEM	8
2.2.5	Quadrilateral 2D FEM	9
2.2.6	Numerical Experiment	9
2.2.7	Error Analysis	11
3	Multigrid Methods	13
3.1	Fixed Point Iterations and Their Shortcomings	13
3.2	Constructing The Two Grid Scheme	16
3.3	The V-Cycle	18
3.4	Operator Induced Interpolation	19
3.4.1	Schur Complements	20
3.4.2	Building R and P	20
3.5	Black Box Multigrid	21
3.5.1	Collapsing the 2D Stencil	22
3.6	Return to Poisson's Equation	23
3.7	BoxMG for Vector Problems	24
3.8	The Scalar and Vector Variants of BoxMG	24
3.8.1	Scalar BoxMG	24
	Vector BoxMG	25
4	BoxMG for MHD	27
4.1	Exact Penalty Method	27
4.1.1	Numerical Experiments	28
	Experiment 1	28
	Experiment 2	29

A	BoxMG Detailed Example	33
A.1	BoxMG Example	33
B	Stokes Equations	37
B.0.1	BoxMG for Stokes Equations	37

Chapter 1

Introduction

1.1 Motivation

The motivation for this project is to develop numerical methods for the simulation of plasma physics. Plasma is roughly speaking the fourth state of matter. A working definition of plasma, as found in [3], is simply an ionized gas. The temperature required for ionization is calculated through the Saha equation [1]. In general, the temperature for ionization is much higher than the average temperature experienced on earth ($288K$), so creating a plasma state must be done in a laboratory with sophisticated equipment. For example, the Z-Machine at Sandia National Laboratories has achieved plasma states[15].

Plasma also occurs naturally in stars. In fact, astronomers agree that 90% of matter in the universe is in a plasma state. This makes sense since stars are the largest stellar bodies, and much of star matter is in a plasma state.

Thus, the two main application areas for studying plasma are

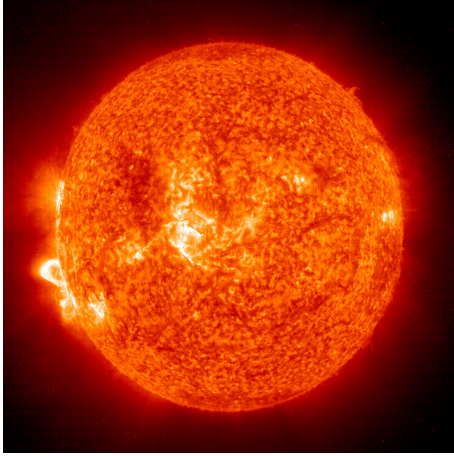
1. How a plasma can be sustained (through magnetic and inertial confinement) on earth for the purpose of controlled nuclear fusion
2. Interstellar plasma, in particular magnetized astrophysical plasmas

These two fields share the common theme that what is studied in particular is a plasma interacting with a magnetic field. The continuum approximation we will be using to model plasma is known as magnetohydrodynamics (MHD).

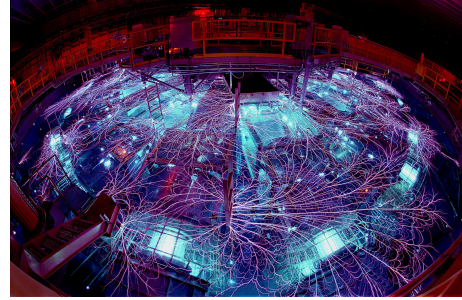
1.2 Numerical Solution

1.2.1 Discretization Methods

Mathematically, at the continuum level MHD is expressed as simultaneous partial differential equations (PDEs). These PDEs are in general non-linear and highly coupled, making solving them analytically extremely difficult. For this reason, we seek to discretize the equations so that they can be solved numerically by computers. There are several existing methods for discretizing PDEs, namely finite difference, finite element, and finite volume. There are pros and cons to each of these methods. However in this exposition we will focus on the finite element method (FEM). Some advantages of FEM are that it easily accommodates complicated geometries, boundary conditions are usually trivial to implement, and there is an elegant mathematical foundation in functional analysis (this is a mathematical thesis after all). Some cons of FEM are the added complexity of enforcing mass conservation, and difficulty when discretizing highly convective operators (leading to techniques like Streamlined



(A) The Core of the Sun



(B) Z-Machine at Sandia Labs

FIGURE 1.1: Examples of Plasma States

Updwind Petrov Galerkin and Discontinuous Galerkin). Despite their similarities and differences, all the methods described here produce a large sparse linear system of equations. The algebraic solution of this system is representative of the analytic solution in a way that can be made precise.

1.2.2 Linear Solvers

Solving linear systems is an old problem in mathematics. There is a great deal of theory concerning the exact solution to linear systems. These so-called direct methods are mostly based on the fundamental approach of Gaussian elimination. Solving a linear system using direct methods, e.g. Gaussian elimination, take $\mathcal{O}(n^3)$, or in special cases $\mathcal{O}(n^2)$ iterations. If n is very large (millions or billions), obtaining an exact solution using modern computers is almost always infeasible. Instead we seek numerical methods for approximating the solution \mathbf{v} , to within a desired tolerance. Due to the sparsity of the matrices produced by FEM, this can be done numerically with a lower asymptotic order. In particular, some methods of interest are so-called Krylov methods and multilevel (multigrid) methods. In this project, a multigrid method will be developed and tested on a variety of linear systems representative of resistive MHD. Results are presented in [chapter 4](#).

1.3 Preliminaries and Notation

The Lebesgue integral of a function f over a Lipschitz domain Ω will be denoted $\int_{\Omega} f$. Furthermore, the L^2 norm of a function $f \in L^2(\Omega)$ as $\|f\| = (\int_{\Omega} |f|^2)^{1/2}$. Similarly, the ℓ^2 norm of a vector $v \in \mathbb{R}^n$ is defined as $\|v\| = (\sum_{i=1}^n v_i^2)^{1/2}$. Also for a vector $\mathbf{v} \in \mathbb{R}^n$, the infinity norm of v , is

$$\|v\|_{\infty} = \max_i |v_i|.$$

We adopt the convention that, for a vector valued function, $\vec{u} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ where $n = 2, 3$ the norm of \vec{u} is defined as $\|\vec{u}\| = \|\|u_1\| + \|u_2\| + \|u_3\|\|$.

1.3.1 Vector Operators

Let $\vec{u}, \vec{v} : \mathbb{R}^n \rightarrow \mathbb{R}^n$, where $n = 2, 3$ be vector valued functions and let \mathbf{S}, \mathbf{T} be rank 2 tensors. We use the BSL conventions so that

$$\nabla \times \vec{v} = \epsilon_{ijk} \frac{\partial v_k}{\partial x_j} \hat{e}_i,$$

$$\nabla \vec{v} = \frac{\partial v_j}{\partial x_i} \hat{e}_i \times \hat{e}_j,$$

$$\nabla \cdot \mathbf{T} = \frac{\partial T_{ij}}{\partial x_i} \hat{e}_j,$$

$$\mathbf{S} : \mathbf{T} = S_{ij} T_{ji}.$$

1.4 Acknowledgements

I would like to thank my advisors, Professor Jehanzeb Chaudhry, Doctor John Shadid, and Doctor Ray Tuminaro. In addition I received help from Sandia Post-docs and PhD students, Doctor Luc Berger-Vergiat, Doctor Matthias Mayr, and Doctor Massimalino Pasini. I have learned a great deal about scientific computing and numerical analysis over the course of this project. I have been very lucky to have had so many skilled mentors. I would also like to thank the Department of Mathematics and Statistics at UNM for making this research project possible.

Chapter 2

The Finite Element Method

As stated in the introduction, the goal of this work is to numerically model the governing equations of MHD. In order to numerically model equations, a discretization process must take place so that the equations can be represented on a computer. The choice is discretization for this project is the Finite Element Method (FEM)[5]. In this Chapter, we present FEM for a simpler but still important PDE, Poisson's equation, in detail.

2.1 Poisson's Equation

Poisson's equation is given by

$$-\nabla^2 u = f. \quad (2.1)$$

where ∇^2 is the Laplace operator, and f, u are functions in some appropriate function space. We suppose f is given data and the goal is to solve for u . This seemingly simple equation appears in many contexts including electrostatics, Newtonian gravity, and diffusion.

2.2 Analytic vs Approximate Solution

While it can be shown that an analytical solution exists for Poisson's equation (through the method of eigenfunction expansion) we will focus on numerical solutions. In particular, we focus on the solution of the system obtained by FEM. The goal of FEM is to construct a linear system that is representative of the continuous problem. In order to do this however, one must first formulate an equivalent problem, namely the variational problem.

2.2.1 Variational Formulation

Consider Poisson's equation with homogeneous Dirichlet boundary conditions:

$$-\nabla^2 u(x) = f(x), \quad x \in \Omega \quad (2.2)$$

$$u(x) = 0, \quad x \in \partial\Omega \quad (2.3)$$

where $\Omega \subset \mathbb{R}^n$, $\partial\Omega$ is its (Lipshitz) boundary. We say that $u \in V$ where $V := \mathcal{H}_0^1(\Omega)$ is the Sobolev space of functions on Ω that vanish on $\partial\Omega$. The first step is to multiply both sides of (2.2) by a "test function", $v \in V$, and integrate over the entire domain Ω . We obtain

$$-\int_{\Omega} v \nabla^2 u = \int_{\Omega} v f. \quad (2.4)$$

It is not clear we have gained anything here. To observe the advantage, recall the generalized product rule:

$$\nabla \cdot (v \nabla u) = v \nabla^2 u + \nabla v \cdot \nabla u.$$

Taking this in combination with the divergence theorem we obtain Green's first identity:

$$\int_{\Omega} \nabla u \cdot \nabla v = \int_{\partial\Omega} u \nabla v \cdot \frac{\partial u}{\partial n} - \int_{\Omega} u \nabla^2 v. \quad (2.5)$$

where $\frac{\partial u}{\partial n}$ is the directional derivative of u in the direction of n . Substituting (2.5) into the left hand side of (2.4) yields

$$\int_{\Omega} \nabla v \cdot \nabla u - \int_{\partial\Omega} \frac{\partial u}{\partial n} v = \int_{\Omega} v f. \quad (2.6)$$

Since v vanishes on $\partial\Omega$, the second term of the left hand side goes to zero so (2.6) reduces to

$$\int_{\Omega} \nabla u \cdot \nabla v = \int_{\Omega} f v. \quad (2.7)$$

We refer to (2.7) as the weak form of Poisson's equation. We now have an equivalent problem to the one defined in (2.2)-(2.3), namely, find $u \in V$ such that (2.7) is satisfied for all $v \in V$. We call this variational problem. To simplify things later we introduce the notation

$$a(u, v) = L(v) \quad \forall v \in V, \quad (2.8)$$

to be equivalent to (2.7) where $a(u, v) := \int_{\Omega} \nabla u \cdot \nabla v$ and $L(v) = \int_{\Omega} f v$.

In order to numerically solve the variational problem, we must first restrict ourselves to searching for a solution in a finite-dimensional space.

2.2.2 Deriving a Linear System from The Weak Form

We now provide a solution for the original problem, that is, how the finite element constructs a representative linear system for a continuous PDE (Poisson in this case). The trick is use (2.8), and a few facts from linear algebra.

It is actually possible to proceed with the derivation at a high level of abstraction. Let V^h be the finite dimensional subspace of V , and let $\dim(V^h) = n$. Next, let $\{\varphi_j\}_{j=1}^n$ be a basis for V^h so we can uniquely decompose a vector $u \in V^h$ as

$$u = \sum_{j=1}^n u_j \varphi_j.$$

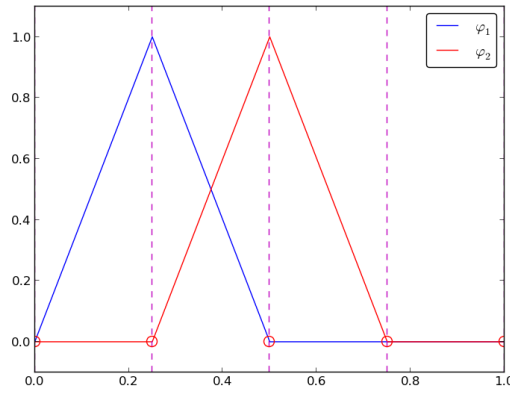


FIGURE 2.1: Example of two hat functions. These types of functions can form a basis of the subspace V^h described in Section 2.2.3

It follows from (2.8)

$$a(u, v) = f(v), \quad \forall v \in V^h \iff (2.9)$$

$$a(u, \varphi_i) = f(\varphi_i), \quad i = 1, \dots, n \iff (2.10)$$

$$a\left(\sum_{j=1}^n u_j \varphi_j, \varphi_i\right) = f(\varphi_i), \quad i = 1, \dots, n \iff (2.11)$$

$$\sum_{j=1}^n a(\varphi_j, \varphi_i) u_j = f(\varphi_i), \quad i = 1, \dots, n (2.12)$$

where the linearity of a justifies the last equivalence. If we now consider the matrix A where $A_{ij} = a(\varphi_j, \varphi_i)$ and $f_i = f(\varphi_i)$, then (2.12) can be written more compactly as

$$A\mathbf{u} = \mathbf{f}. (2.13)$$

2.2.3 FEM in 1D

We now examine some real examples of constructing this finite dimensional space V^h . We will consider only one and two spatial dimensions for simplicity.

For one dimension, consider the case where $\Omega = (0, 1)$. Then let $\mathcal{P} = \{0 = x_0 < \dots < x_n = 1\}$ be a partition of Ω with subintervals $I_j \in \mathcal{P}$, $I_j = [x_j, x_{j+1}]$, and $|I_j| = h_j = h$. Next define V^h as $V^h \subset V = C(0, 1)$ such that $\varphi \in V^h$ is linear on I_j for all $j = 0, \dots, n$. It is a quick matter to show that the set of functions

$$\varphi_j(x_i) = \begin{cases} 1, & i = j \\ 0 & i \neq j \end{cases}, (2.14)$$

form a basis for V^h . See Figure 2.1 for a graph of several of these so-called hat functions. If we proceed in constructing the linear system outlined in the last section, the equation $Au = f$

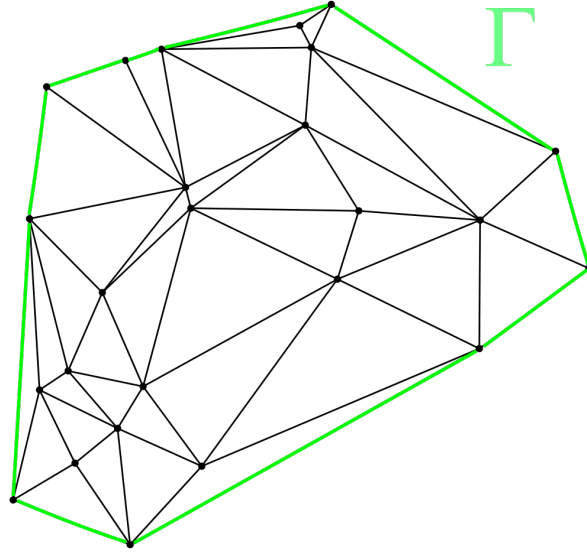


FIGURE 2.2: Simplicial decomposition of Ω defined by a Polygonal Curve $\Gamma = \partial\Omega$

becomes

$$\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-2} \\ f_{n-1} \end{bmatrix}. \quad (2.15)$$

2.2.4 Simplicial 2D FEM

In two dimensions, we assume also that $\partial\Omega$ is a polygonal curve. Let $\mathcal{T} = \{T_1, \dots, T_m\}$ be a set of non-overlapping triangles such that

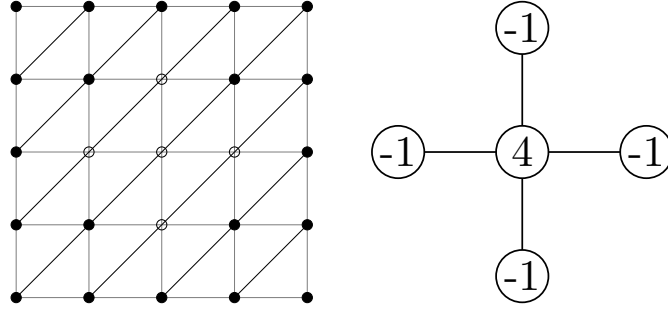
$$\Omega = \bigcup_{i=1}^m T_i,$$

as in Figure 2.2. In the language of homotopy theory, \mathcal{T} is a simplicial complex, but we refer \mathcal{T} as a simplicial decomposition of $\partial\Omega$. Next, define $V^h \subset C(\Omega)$ as

$$V^h = \{v : v \text{ continuous on } \Omega, v|_T \text{ is linear for } T \in \mathcal{T}\}$$

Now to construct the basis $\{\varphi_i\}_{i=1}^n \subset V^h$, we first introduce the nodes N_j , for $j = 1, \dots, n$ running through the nodes of \mathcal{T} viewed as a graph. Then we can define the basis function in a similar fashion as in one dimension:

$$\varphi_j(N_j) = \begin{cases} 1, & i = j \\ 0 & i \neq j \end{cases}. \quad (2.16)$$

FIGURE 2.3: Regular Simplicial Decomposition of $\Omega = (0, 1) \times (0, 1)$

Notice that the support of φ_j is exactly all the simplices T_k such that T_k has N_j as one of its vertices. In the special case of a uniform triangulation for a square domain ($\Omega = (0, 1) \times (0, 1)$), depicted in Figure 2.3, the resulting linear system is given by

$$\begin{bmatrix} 4 & -1 & \dots & 0 & -1 & & & & & & \\ -1 & 4 & -1 & 0 & \dots & -1 & & & & & \\ & & -1 & 4 & -1 & & 0 & -1 & & & \\ & & & & \ddots & & & & \ddots & & \\ -1 & 0 & \dots & 0 & -1 & 4 & -1 & 0 & \dots & 0 & -1 \\ & & & & & & -1 & 0 & -1 & 4 & -1 \\ & & & & & & & -1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-2} \\ f_{n-1} \end{bmatrix} \quad (2.17)$$

2.2.5 Quadrilateral 2D FEM

Alternatively, we can discretize a domain using quadrilateral meshes. For simplicity, we will take a square domain, so let $\Omega = (0, 1) \times (0, 1)$. Then proceeding as before, we decompose Ω into non-overlapping squares, $\mathcal{Q} = \{Q_i\}_{i=1}^m$. Consider again the nodes $\{N_j\}_{j=1}^n$, and the basis functions indexed by nodes, $\{\varphi_j\}_{j=1}^n$ the same as in (2.16). Then our new space is $V^h = \text{span}\{\varphi_j\}_{j=1}^n$. Here we see, as in Figure 2.5, that when comparing to a finite difference stencil, there will be five nonzeros in each row for the discrete Laplacian. The difference between a simplicial and quadrilateral discretization changes the sparsity pattern of the resulting matrix as seen in Figure 2.4.

2.2.6 Numerical Experiment

Once our problem is in the form of (2.8), we are more or less ready to solve using the FEniCS software[8]. First, we define $\Omega = (0, 1) \times (0, 1)$. Then we take Ω^h to be an 40x40 triangulation of our domain, and

$$V^h = \{\text{space of piecewise linear polynomials}\}$$

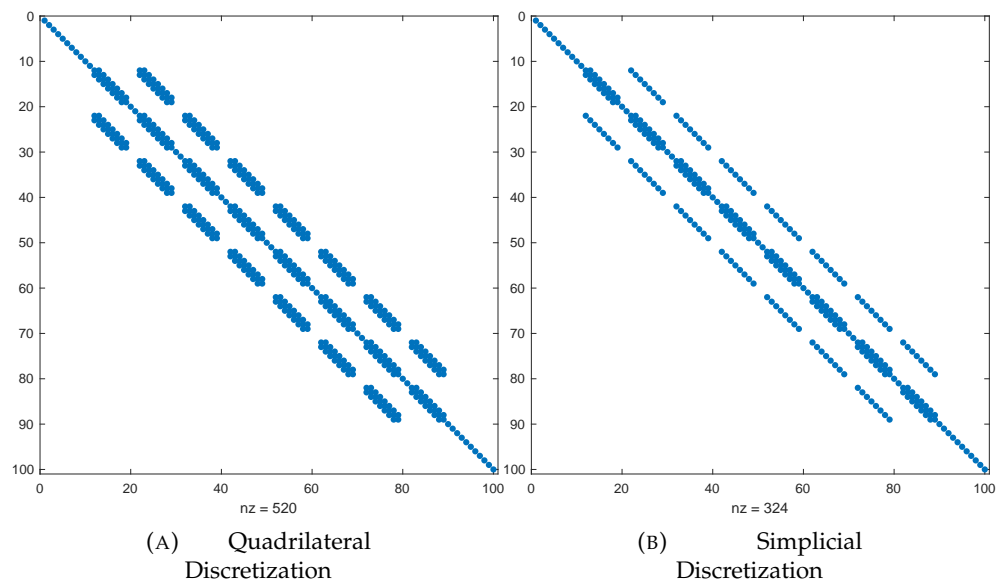


FIGURE 2.4: Demonstrating the different sparsity patterns for discretizing the Laplacian on $(0, 1)$ for solving Poisson's equation

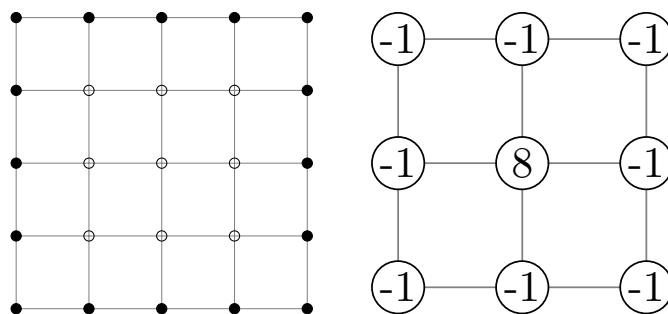


FIGURE 2.5: Regular Quadrilateral Decomposition of $\Omega = (0, 1) \times (0, 1)$

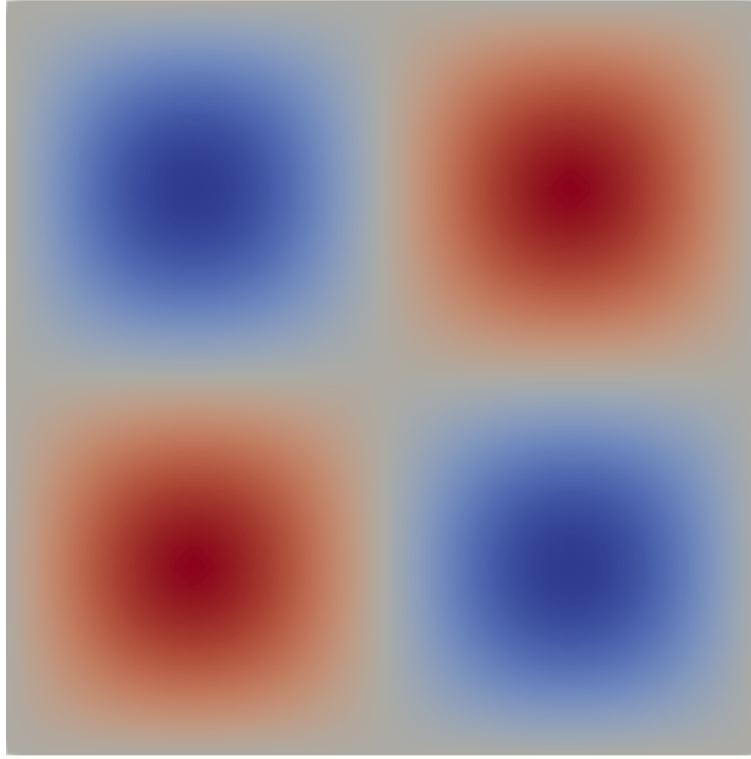


FIGURE 2.6: FEniCS Solution to (2.18)-(2.19)

i.e. if $u^h \in V^h$, $u^h|_T \in \mathbb{P}^1$ for all simplices $T \in \mathcal{T}$. Additionally, we set $u(x, y) = \sin(2\pi x) \sin(2\pi y)$ so that $f(x, y) = 8\pi^2 \sin(\pi x) \sin(\pi y)$. Thus our PDE is given by

$$-\nabla^2 u(x, y) = 8\pi^2 \sin(2\pi x) \sin(2\pi y), \quad x, y \in \Omega \quad (2.18)$$

$$u(x, y) = 0, \quad x, y \in \partial\Omega \quad (2.19)$$

The plot of the solution, $u(x, y)$, is given in 2.6. This result matches the analytical solution well. Indeed, we obtain the theoretical convergence rate to the exact solution as presented in the next section.

2.2.7 Error Analysis

In numerical analysis, one important feature of a numerical method is that of consistency. We define consistency of a numerical method as follows. If we have an approximation of a solution, u_h , and the true solution u , then the consistency error is defined as

$$e_h = \|u - u_h\|. \quad (2.20)$$

We call a numerical method for solving Poisson's equation consistent if

$$\lim_{h \rightarrow 0} e_h = 0. \quad (2.21)$$

We would like to show some numerical evidence that indeed the consistency error is tending to zero as $h \rightarrow 0$. In Table 2.1, values of h are taken along with the L^2 norm as well as the

h	L^2 Error	L^∞ Error
0.1	0.055645	0.045543
0.05	0.014479	0.011269
0.025	0.003656	0.002809
0.0125	0.000916	0.002809

TABLE 2.1: Consistency Test for (2.18)-(2.19). Here h represents the mesh parameter in one spatial dimension, i.e. if there are 10 elements in the x direction then h should be $\frac{1}{10}$. Since we are using a square mesh in 2D, this makes sense as a metric for convergence.

infinity norm for the model Poisson problem, (2.18)-(2.19). In fact, it can be shown that FEM should converge with a rate of h^2 , which agrees with the results of Table 2.1.

Chapter 3

Multigrid Methods

As we saw in the last chapter, solving a linear system can allow us to effectively approximate solutions to PDEs. The matrices we saw in the last section were sparse. In particular, for the matrices of the previous section, the number of non-zeros was $\mathcal{O}(n)$, where n is the number of rows or columns of the matrix. In this chapter we will formulate methods for numerically solving such systems that are also $\mathcal{O}(n)$, namely, multigrid method. Unlike other numerical methods, multigrid takes advantage of the underlying geometric structure of the discretization. Specifically, multigrid transfers the problem attempting to be solved between different grid resolutions. The advantage to this strategy is subtle, but the results are unparalleled. Recall the matrix equation (2.15) for Poisson's equation in 1D from [chapter 2](#). We refer to this matrix as the difference matrix of dimension n and denote it A .

3.1 Fixed Point Iterations and Their Shortcomings

A fixed point iteration is an algorithm for solving a problem where the approximate solution at step i is constructed from the approximate solution at step $i - 1$. In the context of solving linear systems, there are many well known iterative methods. Here, we will examine an iterative method known as the Jacobi Method. Consider (2.2). We can split A up into a sum of matrices $D_A + R_A$ where D_A is the diagonal of A and R_A is the remainder. Consider the equation $A\mathbf{v} = \mathbf{f}$ then

$$(D_A + R_A)\mathbf{v} = \mathbf{f} \implies D_A\mathbf{v} = \mathbf{f} - R_A\mathbf{v} \implies \mathbf{v} = D_A^{-1}(\mathbf{f} - R_A\mathbf{v}).$$

Then if we take an initial guess $\mathbf{v}^{(0)}$ we can compute the solution iteratively as

$$\mathbf{v}^{(i)} = D_A^{-1}(\mathbf{f} - R_A\mathbf{v}^{(i-1)}). \quad (3.1)$$

This reason for the name fixed point iteration is that is that the exact solution satisfies (3.1), i.e. it is a fixed point of (3.1). We can measure the error in (3.1) by $\epsilon_i = \|\mathbf{v}^{(i)} - \mathbf{v}\|$ where \mathbf{v} is the true solution. We call an iterative method convergent if for all initial guesses $\mathbf{v}^{(0)}$ we have that $\lim_{i \rightarrow \infty} \epsilon_i = 0$. Thus convergence implies

$$\mathbf{v} = D_A^{-1}(\mathbf{f} - R_A\mathbf{v}). \quad (3.2)$$

Let $B := -D_A^{-1}R_A$. We call the matrix B the iteration matrix. If we subtract (3.1) from (3.2) and take the norm we obtain $\|\mathbf{v}^{(i)} - \mathbf{v}\| = \epsilon_i = \|B(\mathbf{v}^{(i-1)} - \mathbf{v})\|$. If we substitute in a similar

way for $\mathbf{v}^{(i-1)} - \mathbf{v}$ we see that $\epsilon_i = \|B^2(\mathbf{v}^{(i-2)} - \mathbf{v})\|$, $\epsilon_i = \|B^3(\mathbf{v}^{(i-3)} - \mathbf{v})\|$ and so on until

$$\epsilon_i = \|B^i(\mathbf{v}^{(0)} - \mathbf{v})\|. \quad (3.3)$$

Therefore, we see that in order for the error to go to zero, $\lim_{i \rightarrow \infty} B^i$ must equal the zero matrix.

Since it is difficult to determine the result of repeated matrix multiplications, we turn to the theory of eigenvalues and eigenvectors to analyze convergence. It is known that

$$\lim_{i \rightarrow \infty} B^i = 0 \iff \rho(B) < 1, \quad (3.4)$$

where ρ denotes the spectral radius. So showing the eigenvalues of B are all strictly less than 1 is equivalent to Jacobi's method converging. Our first task will be computing the eigenvectors and eigenvalues of A . We can make use of the trigonometric identity

$$\sin((j+1)\theta) + \sin((j-1)\theta) = 2\sin(j\theta)\cos\theta \quad (3.5)$$

as found in [14]. Define the vector $\mathbf{u} = [\sin\theta, \sin 2\theta, \dots, \sin n\theta]^T$. Then consider the equation

$$(A - 2(1 - \cos\theta)I_n)\mathbf{u} = \sin((n+1)\theta)\mathbf{e}_n, \quad (3.6)$$

where I_n is the identity matrix of dimension n and \mathbf{e}_n is the n th column of this matrix. We can prove (3.6) by considering three cases for j . For $j = 1$,

$$2\cos(\theta)\sin(\theta) - \sin(2\theta) = 0 \quad (3.7)$$

by the double angle identity. For $1 < j < n$ we have

$$-\sin((j-1)\theta) + 2\cos(\theta)\sin(j\theta) - \sin((j+1)\theta) = 0 \quad (3.8)$$

by using (3.5) this time. Finally, if $j = n$,

$$\begin{aligned} & -\sin((n-1)\theta) + 2\sin(n\theta) + [-2\sin(n\theta) + 2\cos\theta\sin(n\theta)] \\ & = -\sin((n-1)\theta) + 2\cos\theta\sin(n\theta). \end{aligned}$$

We can again apply (3.5) with $j = n$ to conclude the (3.6) holds. Then observe the RHS is 0 precisely for $\theta = \theta_k := \frac{k\pi}{n+1}$ for $k \in \mathbb{Z}$. If the RHS vector is 0, this is exactly the eigenvector equation for the matrix A . We call k the wavenumber associated with θ for reasons that will become apparent later. We conclude the eigenvalues of A are

$$\lambda_k = 2(1 - \cos\theta_k) = 4\sin^2\frac{\theta_k}{2} \quad k = 1, \dots, n \quad (3.9)$$

and the associated eigenvectors are given by

$$\mathbf{u}_k = \begin{bmatrix} \sin\theta_k \\ \sin(2\theta_k) \\ \vdots \\ \sin(n\theta_k) \end{bmatrix}, \quad k = 1, \dots, n. \quad (3.10)$$

Recall that we are really concerned with the eigenvalues and eigenvectors of B , the iteration matrix for the Jacobi method. It is easy to verify that $B = I - \frac{1}{2}A$. Then we see the eigenvalues of B are $\lambda(B) = 1 - \frac{1}{2}\lambda(A) = 1 - 2\sin^2(\frac{\theta_k}{2})$. It is clear since $0 < 1 - 2\sin^2(\frac{\theta_k}{2}) < 1, k = 1, \dots, n$, all eigenvalues of B are strictly less than 1 and so Jacobi converges. Unfortunately, for large problems, assurance of convergence is not enough. We must examine the rate at which our method is converging. We can do this here by defining the error vector $\mathbf{e}^{(i)} = \mathbf{v}^{(i)} - \mathbf{v}$ as the un-normed equivalent of ϵ_i .

It is easy to show that B and A have the same eigenvectors. Then since B has n distinct eigenvectors $\mathbf{u}_1, \dots, \mathbf{u}_n$, they form a basis for \mathbb{R}^n so we can write

$$\mathbf{e}^{(0)} = \sum_{k=1}^n c_k \mathbf{u}_k. \quad (3.11)$$

And so using (3.11) we see that

$$\mathbf{e}^{(i)} = B^i \sum_{k=1}^n c_k \mathbf{u}_k = \sum_{k=1}^n c_k B^i \mathbf{u}_k = \sum_{k=1}^n c_k \lambda_k^i \mathbf{u}_k,$$

where λ_k is the eigenvalue of the iteration matrix with wavenumber k . It is then clear that the convergence rate be bounded by the largest such eigenvalue. To gain further insight into these eigenvalues we introduce some definitions. We will call the eigenvalues with wavenumber satisfying $1 \leq k < \frac{n}{2}$, smooth modes and the remaining eigenvalues for which $\frac{n}{2} \leq k \leq n$, oscillatory modes. Then consider the “smoothest” mode that is, $\lambda_1 = 1 - 2\sin^2(\frac{\pi}{2(n+1)}) \approx 1 - \sin^2(\frac{h\pi}{2}) \approx 1 - \frac{\pi^2 h^2}{2}$, which for small h is very close to 1. Therefore, for the smoother modes the Jacobi method will be ineffectual.

The result is that the Jacobi iteration will initially work well, reducing the error in the oscillatory modes. However, it will stall once all the oscillatory components have been reduced. The following figure shows the Jacobi method applied to $A\mathbf{v} = \mathbf{0}$. Here, the initial guess $\mathbf{v}_k^{(i)}$ is k th eigenvector of A . Figure 3.1 shows the result of running a slightly modified version of Jacobi’s algorithm, weighted Jacobi. The iteration for weighted Jacobi takes the form

$$\mathbf{v}^{(i+1)} = \omega D^{-1}(\mathbf{f} - R\mathbf{v}^{(i)}) + (1 - \omega)\mathbf{v}^{(i)}, \quad (3.12)$$

where the parameter ω is usually chosen to be $2/3$. The iteration matrix for weighted Jacobi takes the form

$$B_\omega = I - \omega D^{-1}A.$$

The eigenvalues are then

$$\lambda_j(B_\omega) = 1 - \frac{\omega}{2}(2 - 2\sin \theta_k) = 1 - \omega + \omega \sin \theta_k.$$

The choice $\omega = \frac{2}{3}$ is then optimal for damping high frequencies because for $k \approx n$,

$$1 - \frac{2}{3} + \frac{2}{3} \sin \left(\frac{n}{n+1} \pi \right) \approx \frac{1}{3},$$

in other words high modes get damped by at least a factor of three each iteration. In Figure 3.1, the initial guess \mathbf{u}_k is k th eigenvector of B . We see that indeed, convergence is much

faster for higher values of k , i.e. more oscillatory modes, as initial guesses.

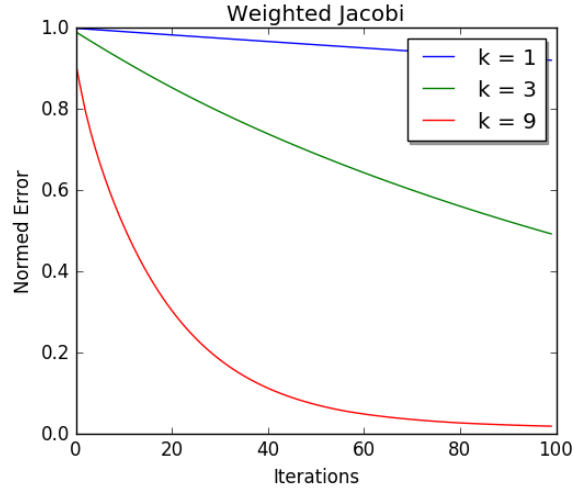


FIGURE 3.1: Using the k th eigenvector as described by (3.10) as an initial guess for weighted Jacobi, (3.12)

3.2 Constructing The Two Grid Scheme

As seen in the last section, Jacobi's method possesses a smoothing property, that is, it is effective in reducing oscillatory modes of error, and ineffective for reducing smooth modes. Thus if we were to take a linear combination of the pure modes of Figure 3.1, the oscillatory modes will quickly be reduced but the method will stall on the smooth modes, e.g. for $k=1$.

Intuitively, if we sample a smooth vector in Ω^h at only odd grid points, the vector will appear more oscillatory as seen in Figure 3.2

This heuristic motivates multigrid methods: *smoother modes appear more oscillatory on a coarser grid*. If we can apply a fixed point method like Jacobi at different grid resolutions, convergence should be obtained in fewer iterations. The first ingredient for a multigrid method are prolongation and restriction operators, P and R . The prolongation operator serves to transfer vectors from a coarser to a finer mesh, and the restriction operator transfers vectors from a finer to a coarser mesh.

Perhaps the most natural choice for the restriction operator is to simply map coarse points on the fine mesh exactly into their corresponding positions on the coarse mesh. In other words, taking the restriction operator to be the identity on coarse points. However, what turns out to be a better choice is to first consider the prolongation operator P . For this operator, it turns out that linear interpolation is a good choice. Thus, we define, for $\mathbf{v}^{2h} \in \Omega^{2h}$, $P^h \mathbf{v}^{2h} = \mathbf{v}^h \in \Omega^h$ as

$$v_{2j}^h = v_j^{2h}, \quad (3.13)$$

$$v_{2j+1}^h = \frac{1}{2}(v_j^{2h} + v_{j+1}^{2h}), \quad 0 \leq j \leq \frac{n}{2} - 1. \quad (3.14)$$

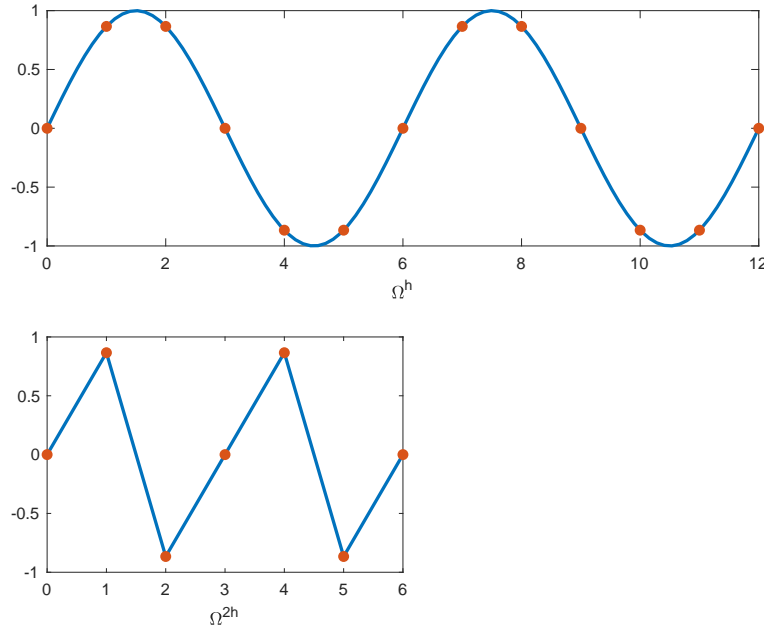


FIGURE 3.2: Comparing Modes on Ω^h and Ω^{2h} . Notice that in the first half of the grid, the second mode appears more oscillatory.

Then we take the restriction operator R to be a constant times P^T :

$$R = cP^T, \quad c \in \mathbb{R}. \quad (3.15)$$

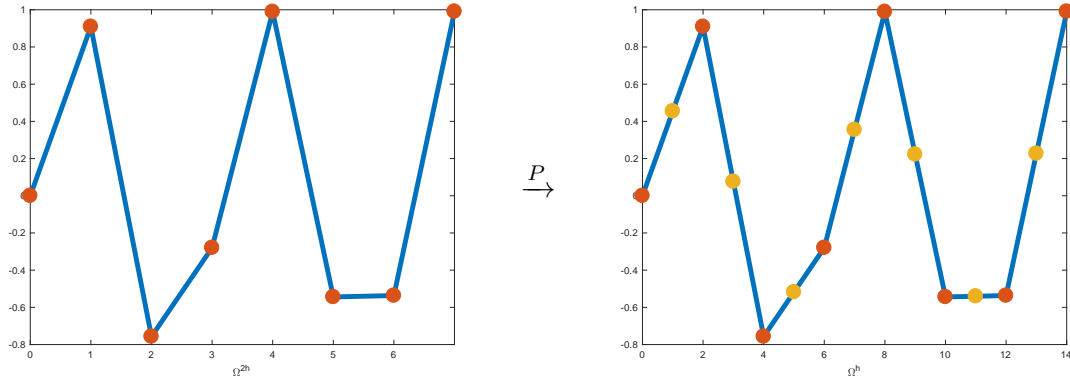


FIGURE 3.3: Showing the action of the prolongation operator P on a coarse grid vector as described in section 3.2.

For a 1D example like [Figure 3.3](#) with 8 nodes on the fine grid instead of 15, we would have the action of P on a coarse grid vector as

$$P\mathbf{v}^{2h} = \frac{1}{2} \begin{bmatrix} 1 & & & & & & & \\ 2 & & & & & & & \\ 1 & 1 & & & & & & \\ & 2 & & & & & & \\ & 1 & 1 & & & & & \\ & & 2 & & & & & \\ & & 1 & 1 & & & & \\ & & & 2 & & & & \\ & & & 1 & & & & \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_{2h} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{bmatrix}_h = \mathbf{v}^h. \quad (3.16)$$

Finally, we note that the operator A^h itself must have some coarser analog. One approach is to simply rediscretize on the coarser mesh using the finite element method. Other approaches are explored later, but for now, assume that grid information is explicitly known, and an FEM discretization can be performed at each grid resolution.

The last piece of the algorithm is making the connection between the original equation, $A\mathbf{v} = \mathbf{f}$ and the so-called residual equation

$$A\mathbf{e} = \mathbf{r}, \quad (3.17)$$

where $\mathbf{r} = \mathbf{f} - A\mathbf{v}$. It is easy to verify solving (2.15) is equivalent to solving (3.17). Additionally, the error is also computed by $\mathbf{e}^h = \mathbf{v} - \mathbf{v}^h$, so we can improve our approximate solution by $\mathbf{v}^h = \mathbf{v}^h + \mathbf{e}^h$.

We are now ready to present the basic two grid scheme in algorithm 1.

Algorithm 1 Two-Grid Cycle

- 1: Relax on $A^h\mathbf{v} = \mathbf{f}$ with initial guess \mathbf{v}_0 to obtain new guess \mathbf{v}^h
 - 2: Compute the residual $\mathbf{r}^h = \mathbf{f} - A^h\mathbf{v}^h$
 - 3: Restrict $\mathbf{r}^{2h} = R^h\mathbf{r}^h$
 - 4: Relax on $A^{2h}\mathbf{e}^{2h} = \mathbf{r}^{2h}$
 - 5: Prolong the coarse grid error vector to the fine grid $\mathbf{e}^h = P^{2h}\mathbf{e}^{2h}$
 - 6: Correct $\mathbf{v} = \mathbf{v} + \mathbf{e}^h$
-

3.3 The V-Cycle

The two-grid-cycle is a good starting point for a multigrid scheme, but in practice it's almost always the case that many levels are necessary to damp all the various frequencies of the error. It is natural to use recursion to generalize 1. This idea has many variants, but we will study only one of these- the V-cycle. As depicted in [Figure 3.7](#), a V-cycle consists of a nested hierarchy of grids, $\Omega^h, \dots, \Omega^{32h}$, in this case with Ω^h being the finest and Ω^{32h} being the coarsest. Initially on the finest grid a smoothing operation like Jacobi's iteration is performed several times. The residual is then calculated, and projected onto the next coarsest grid. Several smoothing sweeps are performed here, but this time on the residual equation, (3.17). This is often called a pre-sweep. After smoothing, the residual is then restricted to the next finest grid. This process is repeated until eventually, on the coarsest grid, the problem is usually small enough to solve directly. Next, the error on the coarsest grid is prolonged

up to the next finer grid. Relaxation is performed with this new initial guess, this is often called a post-sweep. This process is summarized in the recursive algorithm, 2. An example of using a V-cycle as a linear solver is presented in algorithm 3. Essentially, V-cycles are performed until the residual is made as small as a user specified tolerance, tol . Note that in all experiments performed in this work the initial guess \mathbf{v}_0 is chosen to be a constant vector, $[1, \dots, 1]^T$.

Algorithm 2 V-Cycle

Input: \mathbf{v}_0, A, f
Output: \mathbf{v}^h

Relax on $A^h \mathbf{v}^h = \mathbf{f}^h$ with initial guess \mathbf{v}_0 to obtain new guess \mathbf{v}^h

2: Compute the residual $\mathbf{r}^h = \mathbf{f} - A^h \mathbf{v}^h$
 if Ω^h is not coarsest grid then

4: Restrict $\mathbf{r}^{2h} = R^h \mathbf{r}^h$
 Take a new initial guess $\mathbf{v}^{2h} = \mathbf{0}$

6: Set $\mathbf{e}^{2h} = \text{V-Cycle}(A^{2h}, \mathbf{v}^{2h}, \mathbf{r}^{2h})$
 else

8: Set \mathbf{e}^{2h} to Direct-Solve($A^h, R^h \mathbf{r}^{2h}$)
 end if

10: Correct $\mathbf{v} = \mathbf{v} + P^{2h} \mathbf{e}^{2h}$
 Relax on $A^h \mathbf{v}^h = \mathbf{f}^h$ with initial guess \mathbf{v}_0

Algorithm 3 V-Cycle Linear Solver for Numerically Solving $A\mathbf{v} = \mathbf{f}$

Input: \mathbf{v}_0, tol, A, f
Output: \mathbf{v}^h

Set $\mathbf{v}^h = \mathbf{v}_0$
 while $\|\mathbf{f} - A\mathbf{v}^h\| \geq tol$ do

3: $\mathbf{v}^h = \text{V-Cycle}(A, \mathbf{v}^h, f)$
 end while

So far we have been discussing so-called geometric multigrid, where the operators A^{nh} are constructed at each level using FEM on a grid at that level. There is therefore an implicit assumption that such grid information is available. A more general approach is to assume the only information given is the matrix A^h at the finest level. Multigrid methods that only rely on this information are called algebraic multigrid methods. In the next sections we will present a hybrid method that only relies on the operator A^h but also makes certain assumptions about the grid that operator was constructed on.

3.4 Operator Induced Interpolation

In this section we will discuss operator induced interpolation, a hybrid geometric algebraic version of multigrid. In purely geometric multigrid, all operators are constructed based on information about the physical mesh at each level of resolution. However, if only the finest level operator is provided, projections can be used to automate the process of constructing the hierarchy of operators. This is exactly the approach of operator induced interpolation.

One way of automatically generating the hierarchy is to minimize the error in transferring a vector from a coarser to a finer grid. If there is no error at all in this prolongation process, we will have a direct method if we can exactly solve at the coarsest grid. However, first a short digression on Schur complements is in order.

3.4.1 Schur Complements

Let M be an $(p + q) \times (p + q)$ block matrix with blocks

$$M = \begin{bmatrix} A_{p \times p} & B_{p \times q} \\ C_{q \times p} & D_{q \times q} \end{bmatrix}, \quad (3.18)$$

and A invertible. Define the Schur complement of A in M , denoted M/A , as

$$M/A := D - CA^{-1}B. \quad (3.19)$$

The intuition for the Schur complement comes from a kind of block Gaussian elimination. Observe that for the linear system given by

$$Ax + By = u \quad (3.20)$$

$$Cx + Dy = v, \quad (3.21)$$

multiplying (3.20) by CA^{-1} and subtracting it from (3.21), we obtain the so-called Schur system in the second equation:

$$M/Ay = v - CA^{-1}u, \quad (3.22)$$

where the solution the the original system can be obtained by back substitution once (3.22) is solved for y .

3.4.2 Building R and P

Returning to the idea of operator induced interpolation, we again consider the linear system on the fine grid $A^h u = f$. the first step is to construct a new operator \tilde{A}^h given by a symmetric permutation of the original operator A^h :

$$\tilde{A}^h = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix}, \quad (3.23)$$

where each of the submatrices encodes the dependencies of fine and coarse grid points (e.g. A_{fc} encodes how the fine nodes depend on the coarse nodes see Figure 3.4). It should be noted that this permutation is only possible in a straightforward way if the original operator was built on a structured quadrilateral mesh as depicted in 2D by Figure 3.4. As an aside, the coarsening r is defined through the relation

$$n_c = \frac{n_f}{r^D}, \quad (3.24)$$

where n_c is the total number of quadrilateral elements on the fine mesh, n_c is the total number of quadrilateral elements on the coarse mesh, and D is the spatial dimension of the mesh. In

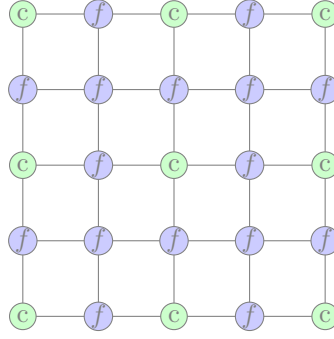
FIGURE 3.4: Diagram showing f and c points

Figure 3.4, $r = 2$ for example. Finally, using the operator (3.23), a linear system of the form $A^h \mathbf{u} = \mathbf{f}$ translates to

$$\begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \begin{bmatrix} \mathbf{u}_f \\ \mathbf{u}_c \end{bmatrix} = \begin{bmatrix} \mathbf{f}_f \\ \mathbf{f}_c \end{bmatrix}. \quad (3.25)$$

The next step is to build restriction and prolongation operators:

$$P = \begin{bmatrix} P_{fc} \\ P_{cc} \end{bmatrix}, \quad P_{fc} = -A_{ff}^{-1} A_{fc}, \quad P_{cc} = I, \quad R = P^T \quad (3.26)$$

P_{cc} is chosen to be the identity to preserve coarse points. The choice of P_{fc} is related to the Schur complement. Observe that for these choice of P and R , we have

$$R \tilde{A}^h P = \quad (3.27)$$

$$\begin{bmatrix} -A_{ff}^{-1} A_{fc} & I \end{bmatrix} \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \begin{bmatrix} -A_{ff}^{-1} A_{fc} \\ I \end{bmatrix} = \quad (3.28)$$

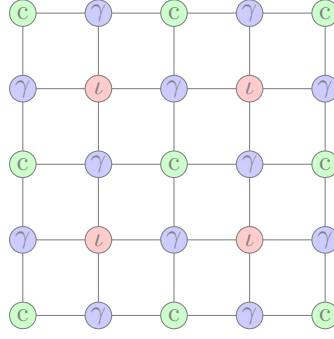
$$\begin{bmatrix} -A_{ff}^{-1} A_{fc} & I \end{bmatrix} \begin{bmatrix} 0 \\ A_{cc} - A_{cf} A_{ff}^{-1} A_{fc} \end{bmatrix} = \quad (3.29)$$

$$A_{cc} - A_{cf} A_{ff}^{-1} A_{fc}, \quad (3.30)$$

which is exactly A_{ff}/\tilde{A} , the Schur complement of A_{ff} in \tilde{A}^h . The matrix $R \tilde{A}^h P$ is called the Galerkin projection of \tilde{A}^h . The Galerkin projection is often a viable choice for the operator on the coarser mesh Ω^{2h} , in this case $\tilde{A}^{2h} = R \tilde{A}^h P$. Furthermore, for this choice of \tilde{A}^{2h} the coarse grid problem $\tilde{A}^{2h} \mathbf{u}^{2h} = \mathbf{f}^{2h}$ is the Schur complement system (3.22) which we have shown is equivalent to solving the original system. The only problem is that inverting A_{ff} is almost as expensive as inverting the original matrix \tilde{A} . To get around this issue, we turn to Black Box Multigrid.

3.5 Black Box Multigrid

Building off of the idea of operator induced interpolation, Black Box multigrid (BoxMG) constructs an approximate Schur complement system to define restriction and interpolation operators. Consider again the matrix \tilde{A} in (3.23). We can further subdivide the fine grid points into fine-points embedded in coarse grid lines (γ) in either x or y , and fine-points in the interior of a coarse grid cell (ι) (see Figure 3.5). Thus the already permuted \tilde{A} from (3.23)

FIGURE 3.5: Diagram Showing ι , γ , and c points

is further divided as

$$\tilde{A} = \left[\begin{array}{cc|c} A_{\iota\iota} & A_{\iota\gamma} & A_{\iota c} \\ A_{\iota\gamma}^T & A_{\gamma\gamma} & A_{\gamma c} \\ \hline A_{\iota c}^T & A_{\gamma c}^T & A_{cc} \end{array} \right]. \quad (3.31)$$

Recall our goal from the last section was to compute the inverse of A_{ff} which in our new notation is

$$A_{ff} = \begin{bmatrix} A_{\iota\iota} & A_{\iota\gamma} \\ A_{\iota\gamma}^T & A_{\gamma\gamma} \end{bmatrix}. \quad (3.32)$$

If $A_{\iota\iota}$ and $A_{\gamma\gamma}$ were diagonal and $A_{\iota\gamma}^T$ was 0, inverting A_{ff} would be much easier as A_{ff} would be upper triangular. Of course to do this we must make assumptions that reduce the connectivity of the graph A_{ff} is representing.

3.5.1 Collapsing the 2D Stencil

To achieve the sparsity pattern outlined in the last section, we must make additional assumptions about the values of the vector that \tilde{A} is operating on. Namely, we must assume that for a γ point embedded in an x or y coarse grid line, the error is constant in the transverse direction. This is a reasonable assumption since after smoothing is performed only smooth vectors remain, which are often nearly constant locally. More concretely, consider a γ point embedded in an x coarse grid line, i.e. the points to the west and east of γ are c points.

$$\begin{bmatrix} -NW & -N & -NE \\ -W & O & -E \\ -SW & -S & -SE \end{bmatrix}. \quad (3.33)$$

If we assume that the error is constant in the y direction, we can “collapse” the stencil along y . Performing this collapse, the 1D averaged stencil is given by,

$$[-(W + SW + NW) \quad (O - S - N) \quad -(E + SE + NE)]. \quad (3.34)$$

This determines the entries of $\hat{A}_{\gamma\gamma}$ and $\hat{A}_{\gamma c}$. Having updated these matrices, our new operator takes the form

$$\tilde{A} = \left[\begin{array}{cc|c} A_{\iota\iota} & A_{\iota\gamma} & A_{\iota c} \\ 0 & \hat{A}_{\gamma\gamma} & \hat{A}_{\gamma c} \\ \hline A_{\iota c}^T & A_{\gamma c}^T & A_{cc} \end{array} \right].$$

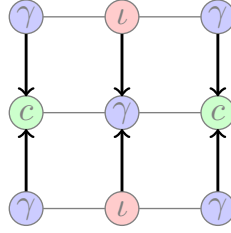


FIGURE 3.6: Demonstrating the BoxMG collapse described in 3.5 for a γ point embedded in an x coarse grid line. The collapse in the y -direction both deletes dependencies from γ to ι points and updates the entries in $A_{\gamma\gamma}$ and $A_{\gamma c}$.

BoxMG for a Quadrilateral Discretization				BoxMG for a Quadrilateral Discretization			
n	V Cycles	Jacobi Sweeps	Levels	n	V Cycles	Jacobi Sweeps	Levels
9	31	(1,1)	2	9	6	(3,3)	2
27	35	(1,1)	3	27	8	(3,3)	3
81	39	(1,1)	4	81	7	(3,3)	4

(A) V-Cycle count using only 1 Jacobi pre and post sweep

(B) V-Cycle count using 3 Jacobi pre and post sweeps

TABLE 3.1: Demonstrating the mesh independence property of BoxMG for solving Poisson's equation on a quadrilateral mesh. The notation (a, b) means a pre-sweeps are performed and b post-sweeps are performed.

Now, by construction,

$$A_{ff} = \begin{bmatrix} A_{uu} & A_{\iota\gamma} \\ 0 & \hat{A}_{\gamma\gamma} \end{bmatrix} \quad (3.35)$$

to build the prolongation operator,

$$\hat{P}_{fc} = \begin{bmatrix} A_{uu} & A_{\iota\gamma} \\ 0 & \hat{A}_{\gamma\gamma} \end{bmatrix}^{-1} \begin{bmatrix} A_{\iota c} \\ \hat{A}_{\gamma c} \end{bmatrix} = -\hat{A}_{ff}^{-1} \hat{A}_{fc} \quad (3.36)$$

gives an approximate to the P from the previous section. We should note that the method presented here corresponds to a coarsening rate of 2, as described earlier. However, in all the numerical studies presented here, a coarsening rate by a factor of 3 is used [6], which operates using the same collapse operation described here. For a detailed example of BoxMG, see [Appendix A](#).

3.6 Return to Poisson's Equation

We now present some results using a BoxMG scheme to iterative solve Poisson's equation in 2D. [Table 3.1](#) shows BoxMG applied to a Quadrilateral Discretization and [Table 3.2](#) shows BoxMG applied to a simplicial discretization. Notice that fewer iterations are required in the case of the quadrilateral discretization because BoxMG is assuming a quadrilateral structure from the mesh, which in turn produces a specific sparsity pattern.

BoxMG for a Simplicial Discretization			
n	V Cycles	Jacobi Sweeps	Number of Levels
9	46	(1,1)	2
27	54	(1,1)	3
81	61	(1,1)	4

TABLE 3.2: Demonstrating the mesh independence property of BoxMG for solving Poisson's equation on a quadrilateral mesh. The notation (a, b) means a pre-sweeps are performed and b post-sweeps are performed

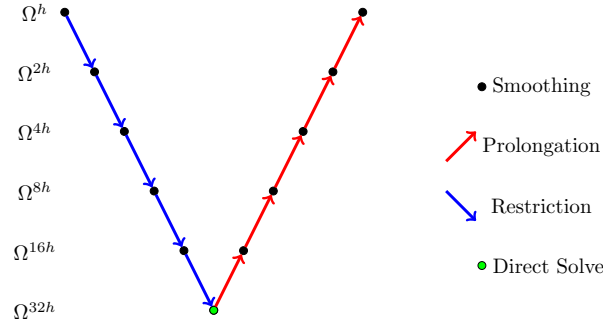


FIGURE 3.7: Graphic illustration of a V-cycle as discussed in section 3.3 with 5 levels of coarsening.

3.7 BoxMG for Vector Problems

The goal of this work is to extend the BoxMG method for building a hierarchy of operators for a vector PDE, i.e. a PDE where the unknown function $\vec{u} = (u_1, u_2, \dots, u_n)^T$ is vector valued with n component functions. These functions u_1, \dots, u_n are often called degrees of freedom of the system, or for shorthand DOFs. In Chapter 4, we will look at vector PDE systems governing magnetohydrodynamics. Also, in Appendix B we present results for a scalar variant of BoxMG for Stokes equations. To clarify, the terminology vector and scalar variants of BoxMG are both applied to vector problems. The scalar variant constructs the BoxMG prolongation operator $P = -\hat{A}_{ff}\hat{A}_{fc}$ for each DOF separately. On the other hand, the vector variant constructs P using information that includes the relationships between DOFs, also known as coupling. This will be elaborated further in the following section.

3.8 The Scalar and Vector Variants of BoxMG

3.8.1 Scalar BoxMG

For simplicity, consider a vector PDE with two unknown functions $u_1(x, y)$ and $u_2(x, y)$. Suppose the discretized system using FEM us given by

$$\begin{bmatrix} A & B^T \\ B & A \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix}. \quad (3.37)$$

The scalar variant of BoxMG constructs a prolongation matrix P_A by first applying BoxMG to the discretized Laplacian block A . Then the full P matrix is constructed simply as

$$P = \begin{bmatrix} P_A & 0 \\ 0 & P_A \end{bmatrix}. \quad (3.38)$$

This method is called the scalar variant because it relies on an application of BoxMG scalar algorithm.

Vector BoxMG

Consider again (3.37). Clearly if the off diagonal block $B = 0$ the scalar BoxMG variant will work perfectly well. However, in general this block will not be 0, and so the scalar variant will be somehow missing coupling information in the operators it constructs. One option to remedy this is to change our point of view from looking at the system in terms of DOFs to looking at the system in terms of nodes. Assuming all the DOFs are collocated, there will be n equations being solved at each node assuming there are n DOFs. The vector variant of BoxMG constructs subsystems so that the compass stencils appear as

$$\begin{bmatrix} -NW_{ij} & -N_{ij} & -NE_{ij} \\ -W_{ij} & O_{ij} & -E_{ij} \\ -SW_{ij} & -S_{ij} & -SE_{ij} \end{bmatrix}, \quad (3.39)$$

where i ranges over the DOFS for the current node, and j ranges over the DOFs for the node to collapse. Then the BoxMG collapse as presented before proceeds in the exact same manner. For a more detailed look at this algorithm, see [7].

Chapter 4

BoxMG for MHD

We now return to the problem of modeling and efficiently solving MHD equations. First we present the full coupled equations. Next we present a stabilized FEM discretization of one of the MHD equations. Finally we make the connection with BoxMG for solving the system and present numerical results. For this project we take a so-called one-fluid visco-resistive MHD system [12][3], where the governing equations are given by

$$\frac{\partial(\rho \vec{u})}{\partial t} + \nabla \cdot [\rho \vec{u} \otimes \vec{u} - \mathbf{T}] - \vec{J} \times \vec{B} - \vec{f}(T) = 0, \quad (4.1)$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot [\rho \vec{u}] = 0, \quad (4.2)$$

$$\frac{\partial(\rho e)}{\partial t} + \nabla \cdot [\rho \vec{u} e + \vec{q}] - \mathbf{T} : \nabla \vec{u} - \eta \|\vec{J}\|^2 = 0, \quad (4.3)$$

$$\frac{\partial \vec{B}}{\partial t} - \nabla \times (\vec{u} \times \vec{B}) + \nabla \times \left(\frac{\eta}{\mu_0} \nabla \times \vec{B} \right) = \vec{0}, \quad \nabla \cdot \vec{B} = 0. \quad (4.4)$$

where \vec{B} is the unknown magnetic field and \vec{u} is the unknown velocity field. The rest of the variables are parameters depending on the system. We will be focusing our attention on (4.4) for this project. This first equation of (4.4) is called the induction equation, and determines the time evolution of the magnetic field \vec{B} , while the second is called Gauss' law for magnetic fields. Physically it corresponds to the lack of magnetic monopoles.

4.1 Exact Penalty Method

The induction equation (4.4) is difficult to solve because the curl operator, $\nabla \times$, has a large nullspace. In particular, all conservative vector fields $\vec{B} = \nabla \varphi$ for a scalar potential φ , are in the nullspace of the curl operator. In a finite dimensional setting this translates to a matrix representation of the discretized operator having diagonal entries with small magnitude, i.e. not strictly diagonally dominant (SDD). Unfortunately, multigrid methods tend to work best for SDD systems. One stabilization approach, as taken in [11], is the so-called Exact Penalty method. This method takes advantage of the vector identity

$$\nabla^2 A = \nabla(\nabla \cdot A) - \nabla \times (\nabla \times A). \quad (4.5)$$

Since analytically, $\nabla \cdot \vec{B} = 0$ we are justified to add the term $-c_p^2 \nabla(\nabla \cdot \vec{B})$ (where different choices of c_p are chosen depending on the situation) into the strong form, thus (4.4) becomes

$$\frac{\partial \vec{B}}{\partial t} - \nabla \times (\vec{u} \times \vec{B}) + \nabla \times \left(\frac{\eta}{\mu_0} \nabla \times \vec{B} \right) - c_p^2 \nabla(\nabla \cdot \vec{B}) = \vec{0}. \quad (4.6)$$

If we take $c_p = \sqrt{\eta/\mu_0}$, we have “completed” the vector Laplacian. Well-posedness of this solution is proven in [4]. In a finite dimensional setting this means adding terms back to the diagonal of the matrix corresponding to the discretized operator. Restricting to an appropriate discrete vector subspace V^h , and integrating against a test function C^h , we obtain the weak formulation

$$\int_{\Omega} \vec{C}^h \cdot \left[-\nabla \times (\vec{u}^h \times \vec{B}^h) + \nabla \times \left(\frac{\eta}{\mu_0} \nabla \times \vec{B}^h \right) \right] + \int_{\Omega} \frac{\eta}{\mu_0} [\vec{C}^h \cdot \nabla(\nabla \cdot \vec{B}^h)] = \vec{0}, \quad (4.7)$$

where the second integral is a stabilization in the weak form corresponding to the $\nabla(\nabla \cdot \vec{B})$ in the strong form.

4.1.1 Numerical Experiments

Experiment 1

For this experiment, we set the velocity field, \vec{u}^h to $\vec{0}$ and assume the ratio $\eta/\mu_0 = 1$. Thus the weak form becomes

$$\int_{\Omega} \vec{C}^h \cdot \left[\nabla \times (\nabla \times \vec{B}^h) \right] + \int_{\Omega} \vec{C}^h \cdot \nabla(\nabla \cdot \vec{B}^h) = \vec{0}. \quad (4.8)$$

We can simplify (4.8) using vector identities. The final weak form is then

$$\int_{\Omega} (\nabla \times \vec{C}^h) \cdot (\nabla \times \vec{B}^h) + \int_{\Omega} (\nabla \cdot \vec{C}^h)(\nabla \cdot \vec{B}^h) = \vec{0}. \quad (4.9)$$

Using both the FEniCS software as well as Sandia’s Drekar [10], we construct a linear system from this weak form for two manufactured problems. We define our problem on the unit square, $\Omega = (0, 1) \times (0, 1)$. For this manufactured problem we solve the modified strong form, (4.6), given by

$$\begin{aligned} \nabla \times (\nabla \times \vec{B}) - \nabla(\nabla \cdot \vec{B}) + g(\vec{x}) &= \vec{0}, & \vec{B} &\in \Omega \\ \vec{B} &= 0, & \vec{B} &\in \partial\Omega. \end{aligned} \quad (4.10)$$

where $g(\vec{x}) : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, $g(\vec{x}) = [0, -2]^T$. The exact solution of (4.10) is $\vec{B} = [0, x(x-1)]^T$. The quadratic profile is plotted in Figure 4.1 and the convergence rates are compared to the theoretical expectation of h^2 in Figure 4.3, where h is the mesh parameter $\frac{1}{n}$ where n is the number of rows or columns of the FEM matrix.

The results of BoxMG for this problem are shown in Table 4.2 and Table 4.1. A tolerance of 10^{-8} on the norm of the residual $\mathbf{r} := A\mathbf{v} - \mathbf{f}$ is used as a convergence criteria. We observe a constant number of V-Cycles, even as we exponentially increase the size of the mesh. Since a V-Cycle takes time preportional to n , we conclude the algorithm is running in $\mathcal{O}(n)$ time.

Vector BoxMG for the Parabolic Problem (FEniCS)			
n	V Cycles	Jacobi Sweeps	Number of Levels
9	13	(3,3)	3
27	14	(3,3)	4
81	14	(3,3)	5

TABLE 4.1: Demonstrating the mesh independence property of the so-called vector BoxMG method proposed in [section 3.8](#) in solving (4.10), where the discretized problem is constructed and verified using the FEniCS software. The notation (a, b) means a pre (respectively b post) smoother sweeps are applied at each level.

This is the best we can possibly do since it takes $\mathcal{O}(n)$ time to process the data: the non-zero entries of the FEM matrix.

We consider the problem (4.10) uncoupled in the sense that the equation for B_x does not depend on B_y and vice versa. The results in [Table 4.1](#) and [Table 4.2](#) show that for this uncoupled problem the difference between the vector BoxMG and scalar BoxMG variants is negligible. In fact, both algorithms have exactly the same number of V-cycles for an increasing mesh size. This is expected theoretically since the scalar BoxMG works well for an uncoupled system like [Equation 4.10](#).

For [Table 4.3](#), there is again negligible difference between V-Cycle counts for a discretization using Drekard, and one using FEniCS. This is a good indication that the problems being built in both software packages are in agreement.

Experiment 2

For this experiment we include a nonzero velocity, which in turn produces a coupling between the B_x and B_y components of magnetic field. For the manufactured solution we take $B_x = \sin(2\pi y)$ and $B_y = \sin(2\pi x)$. Then clearly this solution is divergence free i.e. $\nabla \cdot \vec{B} = 0$. Furthermore, we take a constant velocity field, $u_x = 1, u_y = 0$. The induction equation is then

$$\begin{aligned} \nabla \times (\vec{u} \times \vec{B}) + \nabla \times (\nabla \times \vec{B}) - \nabla(\nabla \cdot \vec{B}) + g(\vec{x}) &= \vec{0}, & \vec{B} &\in \Omega \\ \vec{B} &= 0, & \vec{B} &\in \partial\Omega, \end{aligned} \quad (4.11)$$

where $g(\vec{x}) = [-4\pi^2 \sin(2\pi y), -4\pi^2 \sin(2\pi y) - 2\pi \cos(2\pi y)]^T$. We see a substantial difference between [Table 4.4](#) and [Table 4.5](#) in terms of V-Cycle count. We interpret this to mean that the vector variant of BoxMG is more robust on a coupled problem, which is expected since it preserves coupling in the intergrid transfer operators P and R .

Scalar BoxMG for Parabolic Problem (FEniCS)			
n	V Cycles	Jacobi Sweeps	Number of Levels
9	13	(3,3)	3
27	14	(3,3)	4
81	14	(3,3)	5

TABLE 4.2: Demonstrating the mesh independence property of the so-called scalar BoxMG method proposed in [section 3.8](#) in solving (4.10), where the discretized problem is constructed and verified using the FEniCS software. The notation (a, b) means a pre (respectively b post) smoother sweeps are applied at each level.

Vector BoxMG for the Parabolic Problem (Drekar)			
n	V Cycles	Jacobi Sweeps	Number of Levels
3	8	(3,3)	2
9	11	(3,3)	3
27	12	(3,3)	4
81	13	(3,3)	5

TABLE 4.3: Demonstrating the mesh independence property of the so-called vector BoxMG method proposed in [section 3.8](#) in solving (4.10), where the discretized problem is constructed and verified using the Drekar SNL software. The notation (a, b) means a pre (respectively b post) smoother sweeps are applied at each level.

Vector BoxMG for the Trigonometric Problem (Drekar)			
n	V Cycles	Jacobi Sweeps	Number of Levels
9	13	(3,3)	3
27	13	(3,3)	4
81	13	(3,3)	5

TABLE 4.4: Demonstrating the mesh independence property of the so-called vector BoxMG method proposed in Chapter 3 in solving (4.11), where the discretized problem is constructed and verified using the Drekar SNL software. The notation (a, b) means a pre (respectively b post) smoother sweeps are applied at each level.

Scalar BoxMG for the Trigonometric Problem (Drekar)			
n	V Cycles	Jacobi Sweeps	Number of Levels
9	35	(3,3)	3
27	109	(3,3)	4
81	320	(3,3)	5

TABLE 4.5: The number of v-cycles here suggests the mesh independence is not achieved for the coupled version when the scalar version of BoxMG is used. This is contrasted with a constant number of v-cycles at each level of discretization in [Table 4.4](#). Both the scalar and vector BoxMG methods are discussed in [section 3.8](#). The notation (a, b) means a pre (respectively b post) smoother sweeps are applied at each level.

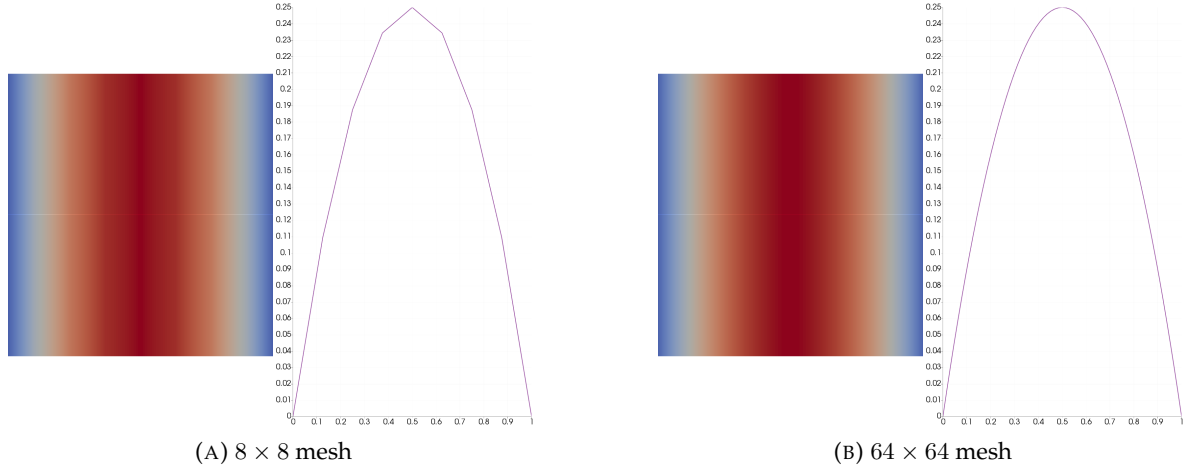


FIGURE 4.1: Plots of the numerical solution of y -component of \vec{B} in (4.10) using FEniCS on varying mesh sizes.

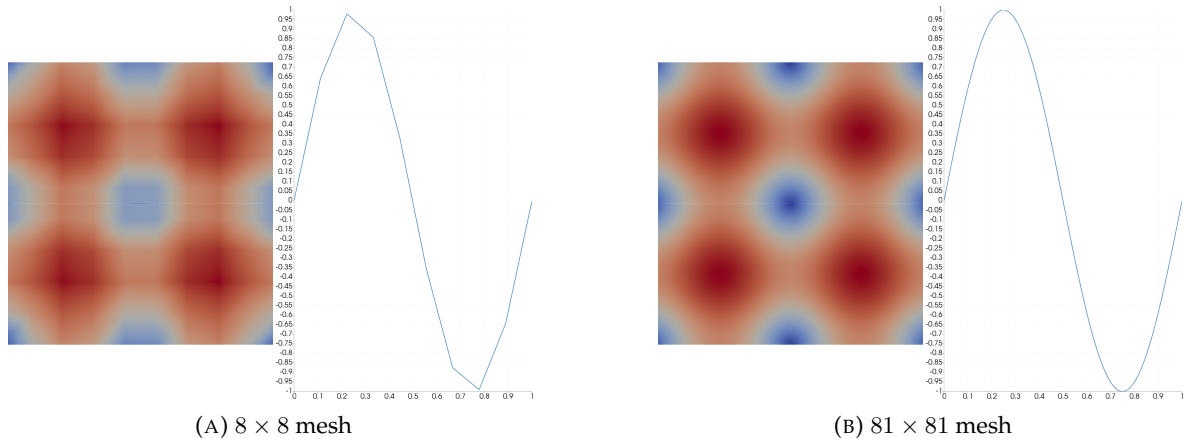


FIGURE 4.2: Plots of the numerical solution of the magnitude of \vec{B} in (4.11) using Drekard on varying mesh sizes. The line plot is taken in the x direction of the B_y component.

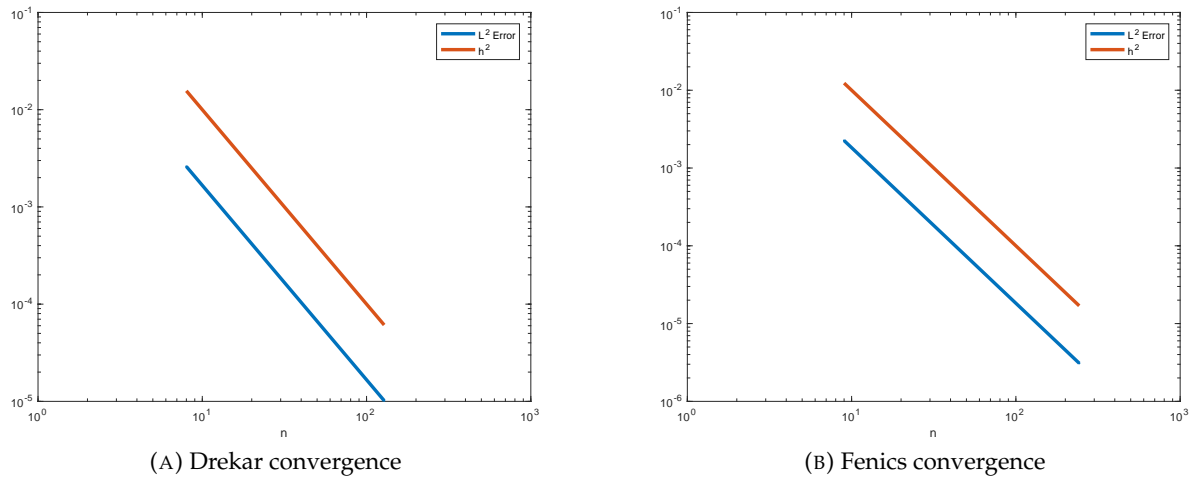


FIGURE 4.3: Rate of convergence Using both FEniCS and for the manufactured problem (4.10) The quadratic convergence is consistent with theory for both software packages. Here h is the the mesh parameter which is the reciprocal of the number of elements on a line.

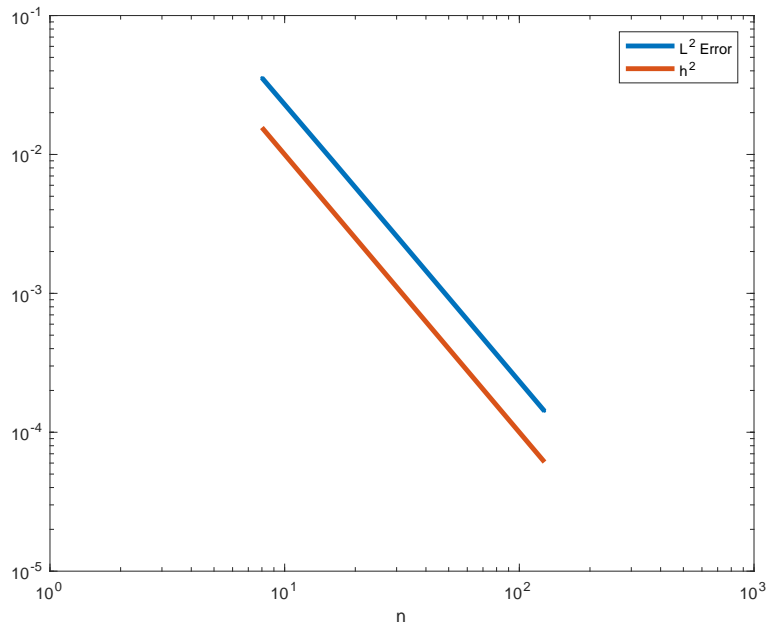


FIGURE 4.4: Rate of convergence for Drekar on the nonzero velocity problem (4.11) The quadratic convergence is consistent with theory. Here h is the the mesh parameter which is the reciprocal of the number of elements on a line.

[illegible]

It follows that the relevant matrices, $A_{\gamma\iota}$, $A_{\gamma\gamma}$, and $A_{\gamma c}$ are given by

$$A_{\gamma\iota} = \begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ -2 & 0 & -2 & 0 \\ 0 & -2 & 0 & -2 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \\ -2 & 0 & 0 & 0 \\ -2 & -2 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & -2 & -2 \\ 0 & 0 & 0 & -2 \end{bmatrix} \quad (\text{A.3})$$

$$A_{\gamma\gamma} = \begin{bmatrix} 8 & 0 & 0 & 0 & 0 & 0 & -2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 & 0 & 0 & -2 & -2 & 0 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 & 0 & -2 & -2 & 0 & -2 & -2 & 0 \\ 0 & 0 & 0 & 16 & 0 & 0 & 0 & -2 & -2 & 0 & -2 & -2 \\ 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 & -2 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 & -2 & -2 \\ -2 & 0 & -2 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 \\ -2 & -2 & -2 & -2 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & -2 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & -2 & 0 & 0 & 0 & 0 & 8 & 0 & 0 \\ 0 & 0 & -2 & -2 & -2 & -2 & 0 & 0 & 0 & 0 & 16 & 0 \\ 0 & 0 & 0 & -2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 8 \end{bmatrix} \quad (\text{A.4})$$

$$A_{\gamma c} = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 \end{bmatrix}. \quad (\text{A.5})$$

After the collapse in the x direction is performed, the matrices appear as

$$(\hat{A}_{\gamma\gamma})_x = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2 & 0 & -2 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 \\ -2 & -2 & -2 & -2 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & -2 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & -2 & 0 & 0 & 0 & 0 & 8 & 0 & 0 \\ 0 & 0 & -2 & -2 & -2 & -2 & 0 & 0 & 0 & 0 & 16 & 0 \\ 0 & 0 & 0 & -2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 8 \end{bmatrix} \quad (\text{A.6})$$

$$(\hat{A}_{\gamma c})_x = \begin{bmatrix} 3 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 \\ -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 \end{bmatrix}. \quad (\text{A.7})$$

Finally, the collapse in the y direction transforms the next rows and the hatted matrices are finally obtained,

$$\hat{A}_{\gamma\gamma} = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix} \quad (\text{A.8})$$

$$\hat{A}_{\gamma c} = \begin{bmatrix} 3 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 0 \\ 3 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 & 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 3 & 0 \end{bmatrix}. \quad (\text{A.9})$$

Appendix B

Stokes Equations

Stokes equations, are a limiting case of the full Navier-Stokes equations for modelling fluid dynamics. In particular, Stokes Equations provide a good model of so-called incompressible flows. A flow is said to be incompressible if its velocity is divergence free, i.e., $\nabla \cdot \vec{u} = 0$ where \vec{u} is the flow velocity. The coupled equations are

$$\begin{aligned} -\Delta \vec{u} + \nabla p &= \vec{f} \\ \nabla \cdot \vec{u} &= 0 \end{aligned} \tag{B.1}$$

where p represents the flow pressure. The second equation is known as the incompressibility constraint. The finite dimensional analog of the continuous system is given by

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \vec{f} \\ g \end{bmatrix} \tag{B.2}$$

The nullspace of B is always non-trivial: $\mathcal{N}(B) = \text{span}\{\mathbf{1}\}$. This corresponds physically to the fact that the solution is non-unique up to a constant pressure. One way to examine the stability in the discretization (B.2) is consider the equivalent pressure Schur complement system

$$BA^{-1}B^T p = BA^{-1}f - g \tag{B.3}$$

It can be shown that (B.2) is numerically (inf-sup) stable if and only if $S := BA^{-1}B^T$ satisfies

$$\mathcal{N}(S) = \mathcal{N}(B) = \text{span}\{\vec{1}\} \tag{B.4}$$

Certain FEM element pairs have been proven to be inf-sup stable, but they introduce other issues. If the element pairs result in an unstable problem, a matrix $-\beta C$, for $\beta \in \mathbb{R}$, is often introduced on the 2x2 block of (B.2) yielding

$$\begin{bmatrix} A & B^T \\ B & -\beta C \end{bmatrix} \begin{bmatrix} \vec{u} \\ p \end{bmatrix} = \begin{bmatrix} \vec{f} \\ g \end{bmatrix} \tag{B.5}$$

This system is a stabilized version of (B.2).

B.0.1 BoxMG for Stokes Equations

In two dimensions, Stokes equations have 3 unknown functions,

$$u_x(x, y), u_y(x, y), p(x, y).$$

n	V-Cycles	Vanka Iterations	Number of Levels
3	9	(3,3)	2
9	14	(3,3)	2
27	17	(3,3)	3
81	19	(3,3)	4

TABLE B.1: Demonstrating mesh independence for solving the 2D lid driven cavity problem with the BoxMG v-cycle described in [subsection B.0.1](#).

We discretize Stokes equations using the IFISS software package[13]. In particular, a $Q1-Q1$ FEM scheme was used, so there will be 3 unknowns or degrees of freedom (DOFs) per node in the FEM mesh. For this reason, it is not clear how to proceed with BoxMG, since it is designed for a single DOF per node.

To overcome this difficulty, we turn our attention to the block matrix given in (B.5). The matrix A turns out to be a block discrete Laplacian operator, with blocks

$$A = \begin{bmatrix} A_x & 0 \\ 0 & A_y \end{bmatrix}, \quad (\text{B.6})$$

where $A_x = A_y = A$ represent the two dimensional discretized Laplacian, as discussed in Chapter 2. Without going into too much detail the matrix $-\beta C$, a so-called mass matrix, also has a similar form to A , at least in terms of sparsity. Thus, one idea to apply BoxMG, is to simply create three prolongation matrices, P_x, P_y, P_p by applying the BoxMG algorithm to A_x, A_y , and $-\beta C$ respectively. The the full P matrix is given by

$$P = \begin{bmatrix} P_x & 0 & 0 \\ 0 & P_y & 0 \\ 0 & 0 & P_p \end{bmatrix} = \begin{bmatrix} P_u & 0 \\ 0 & P_p \end{bmatrix} \quad (\text{B.7})$$

where each P_s is the prolongation operator constructed by BoxMG from the s -DOF matrix. We present the results of solving the classic 2D lid-driven cavity problem [2] using a BoxMG v-cycle in [Table B.1](#). The prolongation matrix P is given by (B.7) and R is simply P^T . Galerkin projections RAP are used to construct the operator at each level. The smoother at each level is a patch-based Vanka smoother [9].

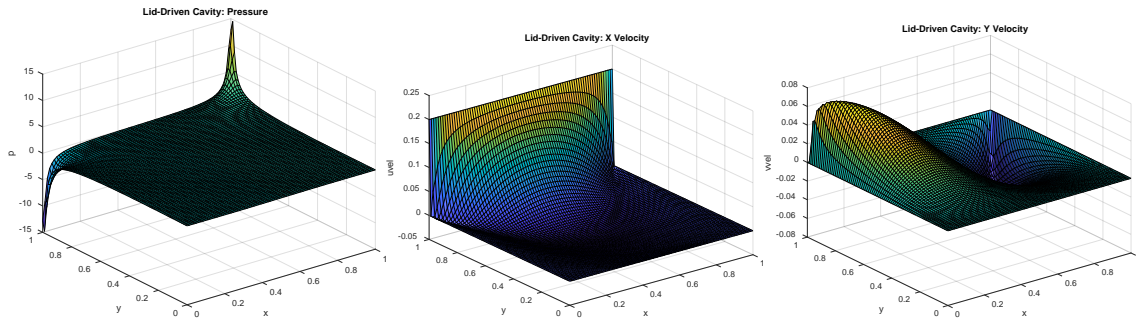


FIGURE B.1: Verificaiton plots for the solution of (B.5) using the BoxMG scheme described in [B.0.1](#)

Bibliography

- [1] Hale Bradt. *Saha Equation*. 2016. URL: <http://homepages.spa.umn.edu/~kd/Ast4001-2015/NOTES/n052-saha-bradt.pdf>.
- [2] U. Ghia, K. N. Ghia, and C. T. Shin. "High-Re Solutions for Incompressible Flow Using the Navier-Stokes." In: *Journal of Computational Physics* 48 (Jan. 1982).
- [3] Hans Goedbloed and Stefaan Poedts. *Principles of Magnetohydrodynamics with Applications to Laboratory and Astrophysical Plasmas*. 2004.
- [4] Max D. Gunzburger, Ammon J. Meir, and Janet S. Peterson. "On the Existence, Uniqueness, and Finite Element Approximation of Solutions of the Equations of Stationary, Incompressible Magnetohydrodynamics." In: *Mathematics of Computation* 56 (1991).
- [5] Claes Johnson. *Numerical Solutions of Partial Differential Equations by the Finite Element Method*. 2009.
- [6] J. E. Dendy Jr. and J.D. Moulton. "Black box multigrid with coarsening by a factor of three." In: *Numerical Linear Algebra with Applications* (2010). URL: <http://onlinelibrary.wiley.com/doi/10.1002/nla.705/pdf>.
- [7] J.E. Dendy Jr. "Black Box Multigrid for Systems." In: *Applied Mathematics and Computation* 19 (1986).
- [8] J. Blechta M. S. Alnaes et al. *The FEniCS Project Version 1.5*. English. Version Archive of Numerical Software, vol. 3, 2015, [DOI]. 2012.
- [9] S. Manservigi. "Numerical Analysis OF Vanka-Type Solvers For Steady Stokes And Navier-Stokes Flows." In: *SIAM Journal of Numerical Analysis* (2006). URL: <http://epubs.siam.org/doi/pdf/10.1137/060655407>.
- [10] Ben Seefeldt et al. *Drekar*. English. Version 2.0. Sandia National Laboratories. 2017.
- [11] J.N. Shadid et al. "Scalable implicit incompressible resistive MHD with stabilized FE and fully-coupled Newton-Krylov-AMG." In: *Computer Methods in Applied Mechanics and Engineering* 304.12 (Dec. 2016). URL: <http://www.sciencedirect.com/science/article/pii/S0045782516300184?via%3Dihub>.
- [12] J.N. Shadid et al. "Towards a scalable fully-implicit fully-coupled resistive MHD formulation with stabilized FE methods." In: *Journal of Computational Physics* 229 (Feb. 2010). URL: <https://www.journals.elsevier.com/journal-of-computational-physics>.
- [13] David Sylvester, Howard Elman, and Alison Ramage. *Incompressible Flow Iterative Solution Software*. English. Version Version 3.0.0. University of Manchester, University of Maryland, and University of Strathclyde. 2009. 32 pp.
- [14] Steve F. McCormick William L. Briggs Van Emden Henson. *A Multigrid Tutorial Second Edition*. Other Titles in Applied Mathematics. 2000.
- [15] Z-Machine. 2017. URL: <http://www.sandia.gov/z-machine>.