Collective Relaxation Techniques for Finite-Element Discretizations of Systems of PDEs

An honors thesis for the Department of Mathematics.

Thomas R. Benson

Tufts University, 2010

Contents

1 Introduction

Airplanes that weigh several tons fly gracefully throught the atmosphere faster than the speed of sound. Ships as big as cities not only float but are able to cruise at speeds exceeding 30 knots. Wires electronically transmit massive amounts of information across the country and around the globe nearly instantaneously. Radio waves allow critical information to reach soldiers even in the most remote places on the planet. These phenomena do not happen in some magical, incomprehensible, disorderly way. They are governed by the natural laws of physics and require precise calculations based on principles that have developed over the past several centuries. It is with these calculations that we herein concern ourselves.

When we observe a natural phenomenon, we look for a way in which we can describe what we observe. For this, we look to the language of mathematics. Beginning from elementary principles we can develop models of varying complexity to attempt to explain and understand that which we observe in nature. The phenomena that we are concerned with here require the use of differential equations, which themselves arise naturally.

We recall Newton's Second Law of Motion,

$$F = ma,$$

where F is the force acting on an object, m is the mass of the object and a is the acceleration of the object. But acceleration is the time derivative of velocity v, which in turn is the time derivative of position x. Thus we have

$$F = ma = m\frac{\mathrm{d}v}{\mathrm{d}t} = m\frac{\mathrm{d}^2x}{\mathrm{d}t^2}.$$

We then see that this differential equation came naturally from the observed laws of nature. While this example may seem to be a relatively "nice" example of a differential equation, there are several more examples of meaningful equations that model physical phenomena that are much more complex.

We now consider fluid dynamics and look to see how models for fluid flow arise. The Stokes equations, which we discuss in more depth later, arise from a simplifying case of the Navier-Stokes equations, which arise from conservation laws. The idea of conservation laws is simple. There is some property L that the fluid possesses, such as mass or momentum, and that quantity is neither created nor destroyed. Any sources or sinks of these quantities must be accounted for in our mathematical model. We recognize this principle with puddles on the street after it rains. After a while, we notice that the puddle is gone. That water was not destroyed. It may have seaped into the ground through cracks, been drunk by animals, or simply evaporated away. All of the water that was in the puddle still exists, but it has now left from the relevant domain. Furthermore, if it had rained again before the puddle disappeared, we would notice more water in the puddle. This was not created in the puddle; it merely flowed into the relevant domain.

We represent this principle mathematically as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega} L d\Omega = -\int_{\partial\Omega} L\vec{u}\cdot\vec{n} + \int_{\Omega} Q d\Omega \tag{1.1}$$

where Ω is the control volume, $\vec{u}$ is the fluid velocity, $\vec{n}$ is the outward unit normal and Q represents sources and sinks of the property L. Using some fancy calculus we can apply the Divergence theorem and the Leibnitz rule and find that

$$\int_{\Omega}\frac{\partial}{\partial t} L d\Omega = -\int_{\Omega}\nabla\cdot(L\vec{u})d\Omega + \int_{\Omega} Q d\Omega,$$

which we can simplify to

$$\int_{\Omega} \left(\frac{\partial L}{\partial t} + \nabla \cdot (L\vec{u}) - Q \right) d\Omega = 0.$$

Finally we observe that this was all done for an arbitrary choice of our domain Ω so it follows that

$$\frac{\partial L}{\partial t} + \nabla \cdot (L\vec{u}) - Q = 0.$$

It is this differential equation, which we see is in fact a partial differential equation, with which we will primarily be concerned.

When we analyze the way this equation looks when we consider conserving two important quantities, mass and momentum, we find the first set of equations we hope to solve. This analysis leads to the Navier-Stokes equations (see Chapter 0 of [5] for more complete details):

$$\begin{aligned} \frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} &= -\frac{1}{\rho}\nabla p + \nu \nabla^2 \vec{u} + \vec{\xi} \\ \nabla \cdot \vec{u} &= 0, \end{aligned}$$

where $\vec{u}$ is the fluid velocity, ρ is the fluid density, ν is the kinematic viscosity, p is the pressure and $\vec{\xi}$ is some forcing function. This is the statement of the system for what are called incompressible, Newtonian fluids, examples of which include air and water. These equations are fundamental in the study of fluid dynamics in disciplines ranging from aircraft design to weather prediction.

There are meaningful cases in which it is appropriate to make simplifying assumptions on this system. One such case is that of what is called creeping flow. Not surprisingly, this is simply fluid flow that is very slow or that occurs when the flow is tightly confined, such as molten rock in the Earth's mantle or flow of blood in certain parts of the body [5]. Even Silly Putty exhibits this type of flow if one is patient enough to sit and watch! In this case, we work with the case where $\nu = 1$ and constant density ρ has been absorbed into the pressure term p. Also, the quadratic term, $\vec{u} \cdot \nabla \vec{u}$, as well as the time derivative disappear. The result is the Stokes Equations:

$$\begin{aligned} -\nabla^2 \vec{u} + \nabla p &= \vec{\xi} \qquad \text{in } \Omega \\ \nabla \cdot \vec{u} &= 0 \qquad \text{in } \Omega. \end{aligned}$$

We will study effective ways to solve these equations for the unknown velocity $\vec{u}$ and pressure p a bit later.

We deviate now to discuss another important system of partial differential equations, namely Maxwell's equations. Maxwell's equations are very important in nearly every aspect of electromagnetics. There would be little to be gained by printing them here, but they may be found in Equations (4.1)-(4.4). The application of these equations with which we are concerned is very specific and related to oil exploration done by the company Schlumberger, though there are countless others. Schlumberger an oilfield services company that assists various companies in the petroleum industry. One of the ways they do this is through geophysical surveying, especially subterranean surveying, which is where electrolmagnetics come into play.

It may not seem apparent at first how electromagnetics can be used in geophysical surveying. Basically, the idea is that an electric field will look different in different media. So they send a ship equipped with electromagnetic receivers out into the ocean or Gulf of Mexico and deploy a low-frequency current. The resulting electromagnetic field is measured at the ocean floor. Before doing this, the researchers have made a guess of what the undersea profile is like, which leads to a guess of several parameters in these equations. Specifically, they guess μ, the permeability of the medium, σ, the electrical conductivity of the medium, and ϵ, the permittivity of the medium. We then take their data and use it in our model, which we can then solve for the electric field that Maxwell's Equations tell us we should have. If our solution and the researchers' guess match, then we know that they made an accurate prediction of the geophysical profile. If they are remarkably different, they update their guess and we then solve Maxwell's equations again. Thus we see that we need to solve Maxwell's equations several times relatively quickly.

When these systems of partial differential equations arise, we need to find ways to solve them. Many of these systems do not have analytic solutions that we can find using the tricks of elementary calculus and basic algebra. Instead, we must look for alternative ways to find solutions to these systems. One possible set of methods that we could investigate is using computer technology to attempt to find solutions. One limitation of computer technology is that computers are capable of working with neither the infinite nor with continuous equations. Thus we look to develop methods that accurately represent either these systems or their solutions, or both, discretely in a finite number of dimensions so that we can use the power of computer technology to solve these systems.

2 Background

We have seen that systems of partial differential equations (PDEs) arise naturally to explain various phenomena. These systems are important continuous models of nature, but they are often difficult to solve. There are several reasons for this difficulty. First, analytic solutions are difficult to find for every single possible boundary or initial condition. There exist a few examples of "nice" problems with analytic solutions, such as Poiseuille channel flow, which analytically solves the Navier-Stokes equations as described in [5]. Nevertheless, meaningful examples are difficult to find. Second, many of these PDEs involve higher order derivatives, which requires that solutions satisfy various degrees of continuity. For example, since the Stokes equations have a second derivative term, namely $-\nabla^2\vec{u}$, any analytic solution of these equations would be required to not only be continuous itself, but also have two continuous derivatives. This imposes serious limitations on what sorts of problems we can even hope to solve. If our domain has a very rigid, jagged topography, or our initial or boundary conditions are not continuous, then we may not be able to find the correct solution because it is too discontinous for the model.

We then turn to computational methods to find solutions to these systems of equations. Since computers cannot deal with continuous problems and neither can they compute in an infinite number of dimensions, we look for ways to limit our problems to a finite number of dimensions. We therefore want to discretize the equations and then find methods to solve them. There are several different ways to discretize equations including Finite Volumes, Finite Differences, and Finite Elements. We will employ only these last two here. Finite Differences, in a nutshell, is a way of discretizing the equations by finding discrete approximations to the partial derivatives in the equations using Taylor's Theorem. The Finite Element method, on the other hand, is a little more complicated. First, we find a weak form of the system. Then we limit this to a discrete case. Then we represents the discrete solution with a finite number of basis functions and thus bring the continuous problem down to a finite-dimensional problem. In both cases, we end up with a linear system of equations that we need to solve.

These linear systems that arise from discrete methods often have very nice properties that we can exploit. First, many of these systems are symmetric or Hermitian. This allows for some simplification. Also, most are sparse systems. This alone simplifies things as we are able to create more efficient algorithms for solving them. These systems are often so big that direct methods become impractical methods for finding solutions. Thus, we look for iterative methods that we can use to effectively and efficiently find solutions. Most of these methods are not exact methods. That is, we do not find *the* solution, but rather we approximate the solution to within some given error tolerance. This also allows us to increase efficiency by giving control over how accurate we need to be. For example, if we are using lazers to destroy tumor cells, we will want to be minimize error as much as possible and thus make our error tolerance very small. This might mean a longer run time, but it also means that we minimize damage to surrounding cells. On the other hand, there may be cases where we need not be so accurate in our solution.

The type of iterative methods we consider here are what are called multigrid methods. The basic idea is this. We begin with a guess at the solution x_0 of the system $\mathcal{A}x = b$. This estimate will have two types of error, namely oscillatory and smooth error. We employ a smoother to reduce the oscillatory error. After this smoother, we are left with smooth error. Now we need a way to get rid of this error. Since it is smooth error, it would make sense that we can consider this error on a coarser grid. We thus incorporate a coarse grid correction step where we restrict the residual to a coarser grid and solve for the error on this grid and interpolate back to the fine grid and update our approximation. We can do this recursively to telescope down to a very coarse grid if we have to. Then we run the smoother again.

Two common examples of smoothers are the (weighted) Jacobi method and the Gauss-

Seidel method. These methods both use the diagonal of the matrix $\mathcal{A}$ as a preconditioner. For systems like Poisson's equation,

$$-\nabla^2 u = f,$$

which yields a diagonally dominant finite-element matrix, these types of smoothers based on the diagonal of the matrix work well. We will demonstrate using finite-element discretization of the the 1D Poisson equation.

Suppose we have an interval with n regularly spaced nodes and let the distance between two nodes be denoted h. Then our finite-element matrix $\mathcal{A}$ has the following entries:

$$a_{ij} = \begin{cases} -\frac{1}{h}, & |i-j| = 1; \\ \frac{2}{h}, & i = j; \\ 0, & \text{otherwise,} \end{cases}$$

for $i, j = 1, 2, \cdots, n$. In our case, we have a Dirichlet boundary condition at the far left boundary and a Neumann condition at the far right. Thus we zero the first row and column leaving only a 1 on the diagonal and we adjust the a_{nn} entry to be $a_{nn} = \frac{1}{h}$. We want to test the smoother on $\mathcal{A}x = 0$ for two reasons. First, we know that the solution for non-singular $\mathcal{A}$ should be $x = 0$. Even if $\mathcal{A}$ is singular, then $x = 0$ is still a solution. Second, the error vector in this case is the solution vector.

Thus we apply to this system the Gauss-Seidel smoother, written component-wise as

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j>i} a_{ij} x_j^{(k)} - \sum_{j<i} a_{ij} x_j^{(k+1)} \right)$$

for $i = 1, 2, \ldots, n$, with an initial guess x_0 of a random vector. Figure 1 shows the solution after 300 iterations. We observe that it is very small in magnitude, on the order of 10^{-4},

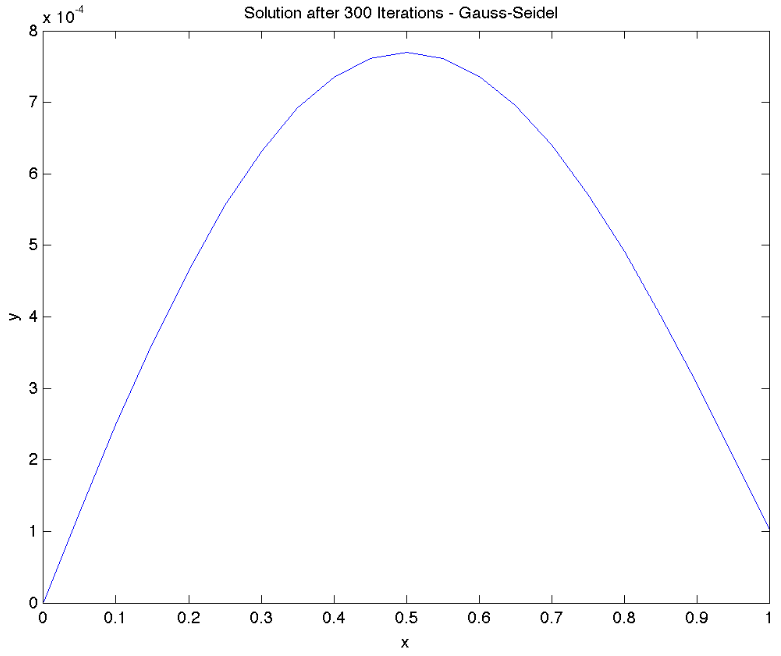


Figure 1: The solution to $\mathcal{A}x = 0$ after 300 iterations of Gauss-Seidel relaxation with a random initial guess. We see that the solution is smooth and small in magnitude. While the oscillatory error has been removed, we need to eliminate smooth error still.

and that it is smooth. This means we have eliminated the oscillatory error that was present in our random initial guess. Figure 2 shows that the scheme has a convergence factor very near to 1. This convergence factor is in fact 0.977789. Nevertheless, the presence of smooth error tells us that we ought to consider a coarse-grid correction scheme.

Note that since these rely explicitly on inverting diagonal elements, they cannot be used if the diagonal of the matrix contains even one zero element. Upon observing a scenario in which Jacobi or Gauss-Seidel smoothing would fail, we might consider what are called block smoothers. For these, our preconditioner is not the diagonal of our matrix but rather a block diagonal matrix where blocks are formed from chosen disjoint blocks of the matrix $\mathcal{A}$. The next option to consider is using what are called block-overlapping smoothers. These are similar to block smoothers except that our blocks are no longer required to be disjoint. The word "overlapping" comes in because we might update the solution at several points

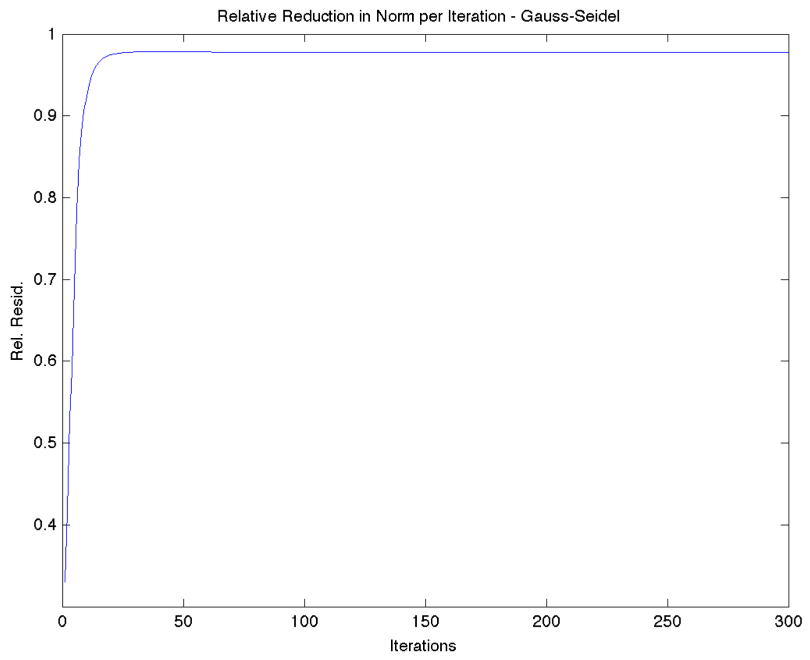


Figure 2: The relative residuals for Gauss-Seidel relaxation on $\mathcal{A}x = 0$ with a random initial guess for 300 iterations. We see that we have a convergence factor of 0.977789, very close to 1.

several times in the course of one sweep of the smoother.

We will look at these block-overlapping smoothers in a little more detail now. Consider the system $\mathcal{A}x = b$. Then let $\mathcal{A}_j$ be the j-th block of the matrix $\mathcal{A}$. The index j indexes some way of tracking which degrees of freedom we want to update with this block of the matrix. There are many possibilities for how we choose j. It might refer to a collection of degrees of freedom surrounding a particular node, those degrees of freedom within a particular area of our domain, or those that share a particular physical significance. The choice then is very dependent on the particular problem. It should be noted that there are choices for j for which we will see convergence and there are choices for which we will not. Now that we have chosen our block, we solve the local system

$$\mathcal{A}_j e_j = r_j$$

where e_j is the error and r_j is the residual associated with those degrees of freedom in block j. Then we add the error correction into our current approximation of the solution as our update. We write this as

$$x_j^{(post)} := x_j^{(pre)} + \mathcal{A}_j^{-1}(b - \mathcal{A}x^{(pre)})_j.$$

It does not seem appropriate to attempt to include a time index here as each element of x will be updated several times in the course of each sweep of the smoother. Thus the superscript "pre" indicates the values of the components of x corresponding to index j before the update at index j and the superscript "post" indicates the values afterward. We now turn to some examples of scenarios in which this type of relaxation scheme is important.

3 Stokes Equations

We begin with the Stokes equations:

$$\begin{aligned} -\nabla^2 \vec{u} + \nabla p &= \vec{\xi} \\ \nabla \cdot \vec{u} &= 0. \end{aligned}$$

We consider this as a boundary value problem with $\Omega = [0,1]^2$ with Dirchlet boundary conditions $\vec{u} = \vec{w}$ on the entire boundary ($\partial\Omega_D = \partial\Omega$).

3.1 Weak Formulation

The subsequent analysis follows the derivation given in [5]. The weak formulation begins with the following identities:

$$\int_\Omega \vec{v} \cdot (-\nabla^2 \vec{u} + \nabla p) = 0 \tag{3.1}$$

$$\int_\Omega q \nabla \cdot \vec{u} = 0 \tag{3.2}$$

for all $\vec{v}$ and q in appropriate spaces of test functions. To relax the continuity requirements on the weak solutions $\vec{u}$ and p, we want to "transfer" derivatives onto the test functions $\vec{v}$ and q in the following way:

$$\begin{aligned} \int_\Omega \vec{v} \cdot \nabla p &= -\int_\Omega p \nabla \cdot \vec{v} + \int_\Omega \nabla \cdot (p\vec{v}) \\ &= -\int_\Omega p \nabla \cdot \vec{v} + \int_{\partial\Omega} p\vec{n} \cdot \vec{v} \\ -\int_\Omega \vec{v} \cdot \nabla^2 \vec{u} &= \int_\Omega \nabla \vec{u} : \nabla \vec{v} - \int_\Omega \nabla \cdot (\nabla \vec{u} \cdot \vec{v}) \\ &= \int_\Omega \nabla \vec{u} : \nabla \vec{v} - \int_{\partial\Omega} (\vec{n} \cdot \nabla \vec{u}) \cdot \vec{v} \end{aligned}$$

using the convention of [5] whereby $\nabla\vec{u} : \nabla\vec{v}$ represents the componentwise scalar product. Combining these gives

$$\int_\Omega \vec{v}\cdot(-\nabla^2\vec{u}+\nabla p) = \int_\Omega \nabla\vec{u} : \nabla\vec{v} - \int_\Omega p\nabla\cdot\vec{v} - \int_{\partial\Omega}\left(\frac{\partial\vec{u}}{\partial n} - p\vec{n}\right)\cdot\vec{v}. \tag{3.3}$$

Before continuing in the analysis, we must introduce appropriate velocity solution and test spaces. We recall the definition of $\mathcal{H}_1(\Omega)$:

$$\mathcal{H}_1(\Omega) := \left\{ u\colon \Omega\to\mathbb{R} \mid u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \in L_2(\Omega) \right\}.$$

From (3.3) and (3.2) we see that there are only first derivatives involved so we have the following solution and test spaces for velocity:

$$\mathbf{H}^1_E := \left\{ \vec{u}\in\mathcal{H}_1(\Omega)^d \mid \vec{u}=\vec{w} \text{ on } \partial\Omega_D \right\},$$
$$\mathbf{H}^1_{E_o} := \left\{ \vec{v}\in\mathcal{H}_1(\Omega)^d \mid \vec{v}=0 \text{ on } \partial\Omega_D \right\}$$

where $d = 2, 3$, the spacial dimension of the problem. Since there are no pressure derivatives, $L_2(\Omega)$ is the appropriate space for p.

We now have the weak formlation of the Stokes problem. We want to find $\vec{u}\in\mathbf{H}^1_E$ and $p\in L_2(\Omega)$ such that

$$\int_\Omega \nabla\vec{u} : \nabla\vec{v} - \int_\Omega p\nabla\vec{v} = \int_{\partial\Omega_N} \vec{s}\cdot\vec{v} + \int_\Omega \vec{\xi}\cdot\vec{v} \quad \text{for all } \vec{v}\in\mathbf{H}^1_{E_o}$$
$$\int_\Omega q\nabla\cdot\vec{u} = 0 \quad \text{for all } q\in L_2(\Omega).$$

Since we have Dirichlet boundaries throughout (and therefore no Neumann boundary condition), the right-hand side of the first equation for our purposes is zero.

Now that we have a weak form, we need to establish that weak solutions are uniquely defined. We should first note that since $\partial\Omega = \partial\Omega_D$, the pressure component of a solution will only be unique up to a constant. Since the bilinear form

$$a(\vec{u},\vec{v}) := \int_\Omega \nabla\vec{u} : \nabla\vec{v} - \int_\Omega p\nabla\cdot\vec{v}$$

is not coercive over $\mathbf{H}^1_{E_o}$, we cannot be guaranteed this right away. But we note that the bilinear form

$$b(\vec{u},\vec{v}) := \int_\Omega \nabla\vec{u} : \nabla\vec{v}$$

is coercive over $\mathbf{H}^1_{E_o}$. We then look to find an *inf-sup*, or LBB, condition to ensure that a pressure solution not only exists, but is unique up to a constant. The inf-sup condition that we must satisfy in this case is

$$\inf_{q\neq\text{constant}} \sup_{\vec{v}\neq\vec{0}} \frac{|(q,\nabla\cdot\vec{v})|}{\|\vec{v}\|_{1,\Omega}\|q\|_{0,\Omega}} \geq \gamma > 0$$

where $\|\vec{v}\|_{1,\Omega} = (\int_\Omega \vec{v}\cdot\vec{v} + \nabla\vec{v} : \nabla\vec{v})^{\frac{1}{2}}$ is a norm for functions in $\mathbf{H}^1_{E_0}$ and $\|q\|_{0,\Omega} = \|q - (1/|\Omega|)\int_\Omega q\|$ is a quotient space norm. This inf-sup condition is then sufficient to show that pressure is uniquely defined up to a constant.

3.2 Mixed Finite Elements

We now turn to defining a discrete weak form using finite-dimensional spaces $\mathbf{X}^h_0 \subset \mathbf{H}^1_{E_o}$ and $M^h \subset L_2(\Omega)$. Since we are approximating two spaces independently, we have what is called "mixed approximation" using "mixed finite elements". The difficulty becomes finding appropriate bases for the chosen spaces. Given $\mathbf{X}^h_E$, a velocity solution space, and recalling that the right-hand side boundary integral in the weak form is zero, we need to find $\vec{u}_h \in \mathbf{X}^h_E$

and $p_h \in M^h$ such that

$$\int_\Omega \nabla \vec{u}_h : \vec{v}_h - \int_\Omega p_h \nabla \cdot \vec{v}_h = \int_\Omega \vec{\xi} \cdot \vec{v}_h \quad \text{for all } \vec{v}_h \in \mathbf{X}_0^h$$
$$\int_\Omega q_h \nabla \cdot \vec{u}_h = 0 \quad \text{for all } q_h \in M^h.$$

Let us define a set of vector-valued velocity basis functions $\{\vec{\phi}_j\}$ such that

$$\vec{u}_h = \sum_{j=1}^{n_u} \mathbf{u}_j \vec{\phi}_j + \sum_{j=n_u+1}^{n_u+n_\partial} \mathbf{u}_j \vec{\phi}_j,$$

with $\sum_{j=1}^{n_u} \mathbf{u}_j \vec{\phi}_j \in \mathbf{X}_0^h$ and the coefficients $\mathbf{u}_j$ for $j = n_u + 1, \ldots, n_u + n_\partial$ fixed so that the second term interpolates the boundary data on $\partial\Omega_d$. Also let us define a set of scalar pressure basis functions $\{\psi_k\}$ such that

$$p_h = \sum_{k=1}^{n_p} \mathbf{p}_k \psi_k.$$

This leads us to the block matrix system

$$\begin{bmatrix} \mathbf{A} & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} \tag{3.4}$$

where $\mathbf{A}$ is the vector-Laplacian matrix with entries

$$\mathbf{a}_{ij} = \int_\Omega \nabla \vec{\phi}_i : \nabla \vec{\phi}_j$$

and B is the divergence matrix with entries

$$b_{kj} = -\int_\Omega \psi_k \nabla \cdot \vec{\phi}_j$$

for $i, j = 1, \ldots, n_u$ and $k = 1, \ldots, n_p$. The right hand side has entries

$$\mathbf{f}_i = \int_\Omega \vec{\xi} \cdot \vec{\phi}_i - \sum_{j=n_u+1}^{n_u+n_\partial} \mathbf{u}_j \int_\Omega \nabla \vec{\phi}_i : \nabla \vec{\phi}_j$$
$$\mathbf{g}_k = \sum_{j=n_u+1}^{n_u+n_\partial} \mathbf{u}_j \int_\Omega \psi_k \nabla \cdot \vec{\phi}_j$$

for the same range of i and k as above. It is important to note that $\mathbf{A}$ is a symmetric, positive-definite matrix. Thus we have our discrete Stokes problem.

In our implementation, we essentially used only one finite-element space. Still following [5], we considered a standard space of scalar finite-element basis functions $\{\phi_j\}_{j=1}^{n}$ and defined our velocity basis set to be

$$\{\vec{\phi}_1, \ldots, \vec{\phi}_{2n}\} := \{(\phi_1, 0)^t, \ldots, (\phi_n, 0)^T, (0, \phi_1)^T, \ldots, (0, \phi_n)^T\}$$

This leads to a natural block splitting of the system (3.4), namely

$$\begin{bmatrix} A & 0 & B_x^T \\ 0 & A & B_y^T \\ B_x & B_y & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_x \\ \mathbf{u}_y \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_x \\ \mathbf{f}_y \\ \mathbf{g} \end{bmatrix}, \tag{3.5}$$

where A is now the $n \times n$ scalar Laplacian matrix. The entries of these matrices are then defined as follows:

$$\begin{aligned} A &= [a_{ij}], & a_{ij} &= \int_\Omega \nabla \phi_i \cdot \nabla \phi_j, \\ B_x &= [b_{x,ki}], & b_{x,ki} &= -\int_\Omega \psi_k \frac{\partial \phi_i}{\partial x}, \\ B_x &= [b_{y,kj}], & b_{y,kj} &= -\int_\Omega \psi_k \frac{\partial \phi_j}{\partial y}. \end{aligned}$$

We now need to consider the solvability of this system. First note that if $n_p > n_u$, the matrix in Equation (3.4) is rank deficient. This means we must choose a higher dimensional space for velocity than for pressure. Now, to determine the solvability of Equation (3.4), we consider the homogeneous system

$$\mathbf{A}\mathbf{u} + B^T\mathbf{p} = 0 \tag{3.6}$$

$$B\mathbf{u} = 0. \tag{3.7}$$

Premultiplying the first equation by $\mathbf{u}^T$ and the second by $\mathbf{p}^T$ implies that $(\mathbf{p}^T B\mathbf{u})^T = \mathbf{u}^T B^T\mathbf{p} = 0$ which then gives that $\mathbf{u}^T\mathbf{A}\mathbf{u} = 0$. Since we know that $\mathbf{A}$ is positive-definite, we then have that $\mathbf{u} = 0$, which implies that the system is uniquely solvable for $\mathbf{u}$. We then substitute $\mathbf{u} = 0$ into the system (3.7)-(3.6), which shows that we must have $B^T\mathbf{p} = 0$. This implies that a pressure solution is only unique up to the null space of B^T. In the case of enclosed flow, then, we must have that $\texttt{null}(B^T) = \{\mathbf{1}\}$.

We need to choose a finite element approximation spaces for this problem that meet all of the requirements above. In addition, we must comply with the discrete form of the inf-sup condtion, namely

$$\min_{q_h \neq \text{constant}} \max_{\vec{v}_h \neq \vec{0}} \frac{|(q_h, \nabla \cdot \vec{v}_h)|}{\|\vec{v}_h\|_{1,\Omega}\|q_h\|_{0,\Omega}} \geq \gamma > 0. \tag{3.8}$$

This leads to approximation using mixed finite elements. Again, of critical importance when choosing finite-element spaces is ensuring that the velocity space is of sufficiently higher dimension than that of pressure.

The way in which we handle the inf-sup condition (3.8) is by using what are called $\mathbf{Q}_2 - \mathbf{Q}_1$, or Taylor Hood, elements. These are rectangular elements in which we use bilinear approximation for the pressure components, but biquadratic approximation for the each velocity components. This requires that we have pressure degrees of freedom at the corners of the elements and velocity degrees of freedom at the center of the elements, at the midpoint

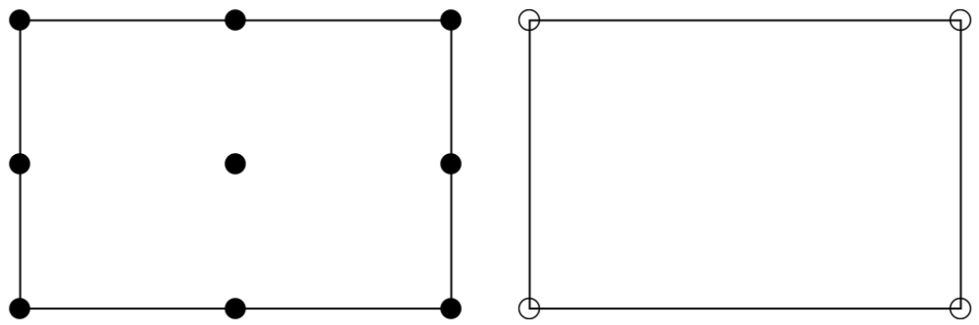


Figure 3: The degrees of freedom on a $Q_2 - Q_1$ element, where a $\bullet$ indicates two velocity degrees of freedom and a $\circ$ indicates a pressure degree of freedom.

of each of the edges of the elements, and at corners of the elements, as shown in Figure 3. Since we are approximating the x-component and the y-component of velocity separately, this means that we have 22 degrees of freedom on each element. Thus if we have an $n \times n$ element grid, we have $n_u = (2n+1)^2$ degrees of freedom for each component of velocity and $n_p = (n+1)^2$ degrees of freedom for the pressure component, or $2n_u + n_p$ total degrees of freedom for the entire problem.

3.3 The Smoother

Now that we have our finite-element discretization and our resulting linear system, we need to solve it. To do this, we will use a multigrid V-cycle routine using the block smoothers described in Section 2 as our relaxation scheme. The first thing we need to do is to decide on our localization index j. Two possibilities for this index are elements and pressure nodes. We used a pressure-centric scheme, so j corresponds to the index of a pressure node. Let

$$\mathcal{A} = \begin{bmatrix} \mathbf{A} & B^T \\ B & 0 \end{bmatrix}.$$

The matrix $\mathcal{A}_j$ then is those entries in $\mathcal{A}$ corresponding to those degrees of freedom that have a connection to pressure node j. Similarly, the components of x_j are those components of x with a connection to pressure node j. For interior pressure nodes, $\mathcal{A}$ is a 25×25 matrix. For edge nodes, it is a 19×19 matrix. For corner nodes it is a 13×13 matrix. Since we are dealing with a uniform grid, we can save considerable time by storing the various $\mathcal{A}_j$'s explicitly within our code. Since they are relatively so small, this does not create a major space inefficiency. Also, because they are both small and dense, we can use the direct methods to solve systems involving them.

Recall that our vector is of the form $\left[\begin{smallmatrix} u \\ p \end{smallmatrix} \right]$ so the local update takes this form:

$$\begin{bmatrix} u \\ \\ p \end{bmatrix}_j := \begin{bmatrix} u \\ \\ p \end{bmatrix}_j + \mathcal{A}_j^{-1} \left(\begin{bmatrix} f \\ \\ g \end{bmatrix} - \mathcal{A} \begin{bmatrix} u \\ \\ p \end{bmatrix} \right)_j$$

For our implementation, we break this step up into six steps, each occuring at each iteration of a loop over all pressure nodes. At each step of the loop we first compute the global residual utilizing the sparse storage format of MATLAB for efficient matrix-vector products. Then, we localize this residual as well as the current solution. Finally we compute the required direct solve using MATLAB's "slash" command (\). Finally, we compute the update on the local level and disperse the result back to the global solution. Looping over all pressure nodes and performing this algorithm at each node then constitutes one sweep of the smoother. We must now investigate how well this smoother performs both independently and when coupled with a coarse-grid correction or multigrid scheme.

3.4 Some Numerical Results

Now that we have completed the finite-element discretization, found a linear system, and developed a smoother that we hope to be effective, we can look at the numerical results for some test problems. First, let us consider the homogeneous equation $\mathcal{A}x = 0$, where again we have

$$\mathcal{A} = \begin{bmatrix} \mathbf{A} & B^T \\ B & 0 \end{bmatrix},$$

the matrix from Equation (3.4) resulting from the finite-element discretization. Obviously the solution to this system is $x = \vec{0}$. We will investigate three different algorithms for solving this system. First, we will look at just the effects of the smoother. Then we will investigate the two-grid algorithm. Finally we will look at the V-cycle multigrid method. In each case, for the $\mathcal{A}x = 0$ problem, we will use a random initial guess. For the velocity degrees of freedom, we use a uniformly distributed initial guess with values in $[0, 1]$. For the pressure degrees of freedom, we again use a uniformly distributed initial guess, but this time with values in $[-\frac{1}{2}, \frac{1}{2}]$. This helps to normalize the pressure solution, which, as we will recall, is only unique up to a constant.

First, we will investigate what effect the smoother has on a random initial guess after several iterations. We ran this experiment on two grid sizes, 16×16 elements and 32×32 elements. After 50 iterations, we clearly see a convergence factor near one in both cases. Figure 4 shows that the relative residuals rapidly approach one. The value of relative residual after 50 steps is in fact near 0.95291 for the 16×16 grid and 0.96969 for the 32×32 grid. Figure 5 shows that after 200 iterations the norm of the residual is approaching zero in both cases. In fact, in the 16×16 case, we see that for 100 iterations, there is a decrease in the norm of the residual by a relative factor of approximately 10^2. In the 32×32 case, 100 iterations only decreases the norm of the residual by about a relative factor of slightly less than 10^1. Finally, Figure 6 and Figure 7 show that the error is very near to zero on the velocity degrees of freedom and is very smooth on the pressure degrees of freedom. Since this is a "smoother", we expect it to smooth the oscillatory compontents of the error, as it has done, but have little effect on that error which is already smooth.

Now we look at the results for the two-grid scheme. We again are looking at solving

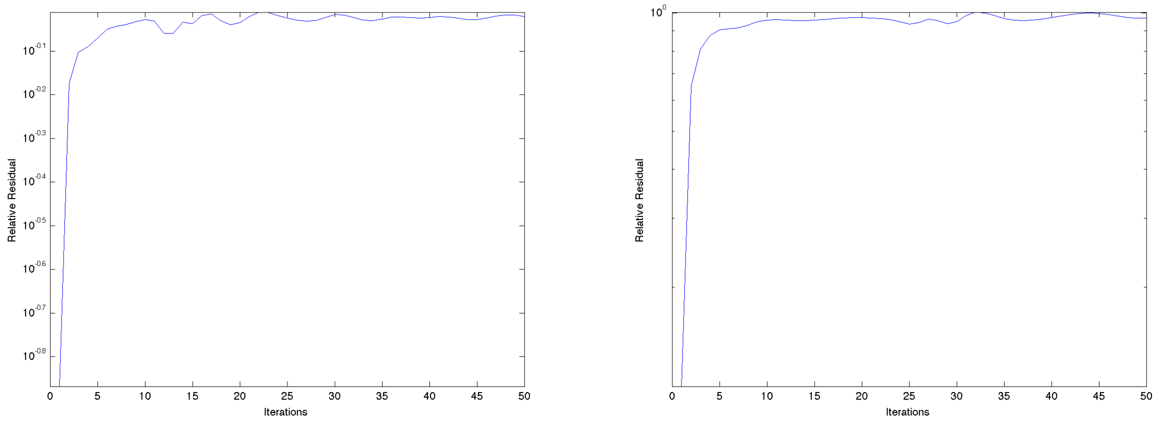


Figure 4: Relaxation of $\mathcal{A}x = 0$ on a 16×16 grid on the left and a 32×32 grid on the right for 50 iterations with a random intial guess. The results show that the relative residuals in both cases very rapidly approach 1.

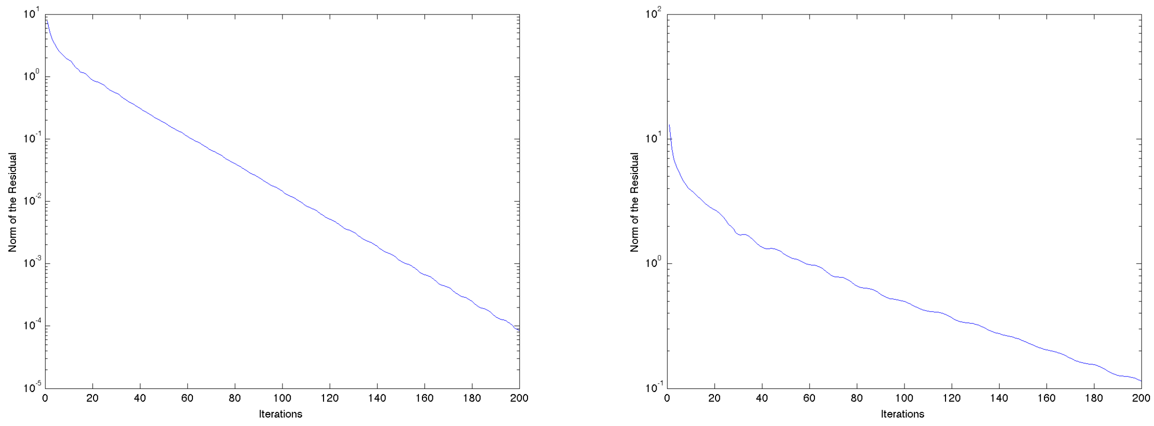


Figure 5: Relaxation on an 16×16 grid on the left and a 32×32 grid on the right of $\mathcal{A}x = 0$ for 200 iterations with a random intial guess. The results show that the norm of the residuals in both cases approaches 0.

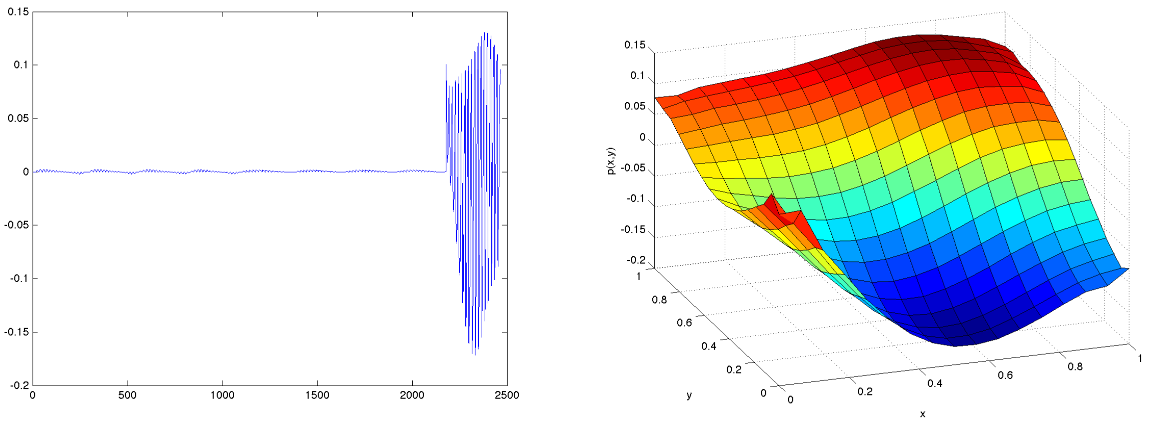


Figure 6: Solution after relaxation on a 16×16 grid of $\mathcal{A}x = 0$ for 100 iterations with a random inital guess. On the left is a plot of the vector representing the solution. On the right is a surface plot of the pressure solution. We see that the velocity degrees of freedom are very uniformly at zero and while the pressure appears oscillatory, it is actually smooth.

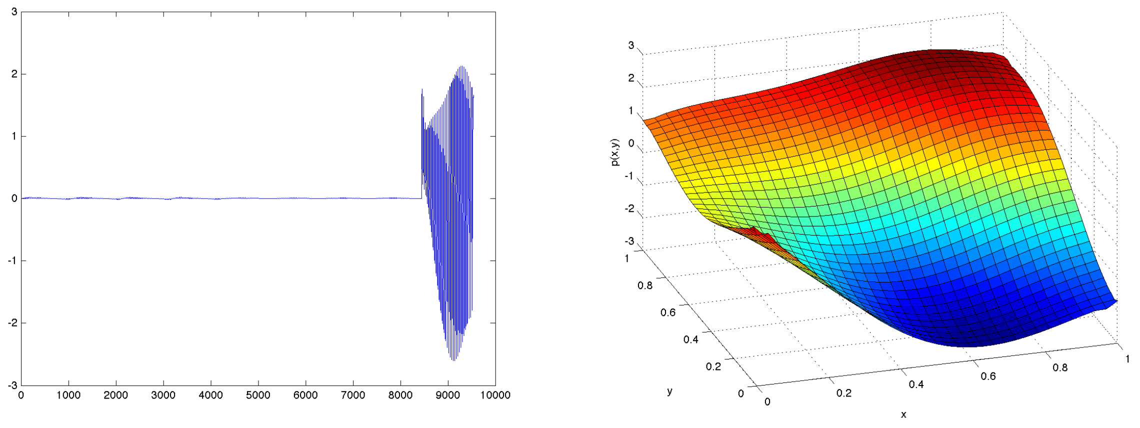
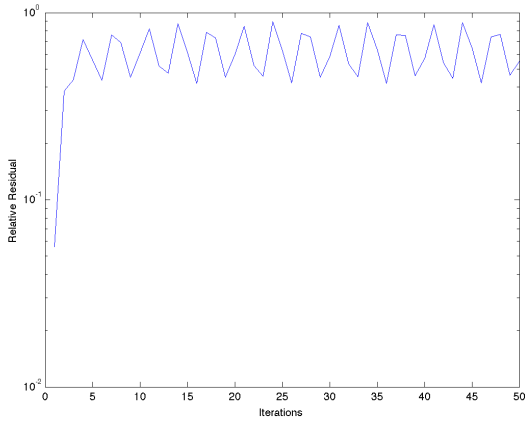


Figure 7: Solution after relaxation on a 32×32 grid of $\mathcal{A}x = 0$ for 100 iterations with a random inital guess. On the left is a plot of the vector representing the solution. On the right is a surface plot of the pressure solution. We see that the velocity degrees of freedom are very uniformly at zero and while the pressure appears oscillatory, it is actually smooth.



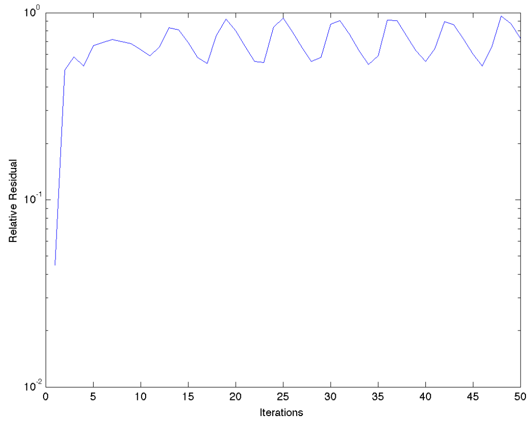
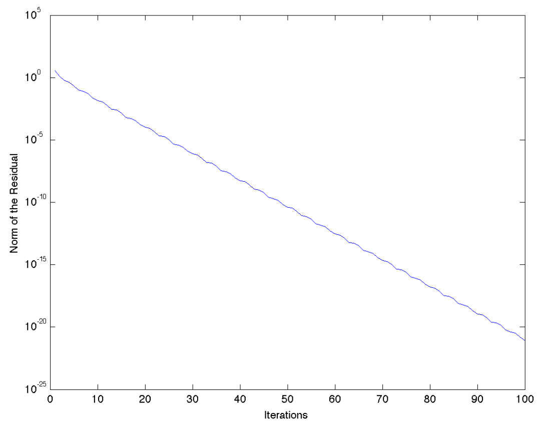


Figure 8: Two-grid solve of $\mathcal{A}x = 0$ on a 16×16 grid on the left and a 32×32 grid on the right for 50 iterations with a random intial guess. The results show that the relative residuals in both cases very rapidly approach 1.

$\mathcal{A}x = 0$ using a random initial guess. We use one forward sweep of relaxation for pre-smoothing and one backward sweep of relaxation for post-smoothing. Figure 8 shows that the relative residuals approach some value fairly near to one again but the convergence factor is slightly lower this time, namely 0.55365 in the 16×16 case and 0.72779 in the 32×32 case. We see in Figure 9 that the norm of the residual in both cases is going to zero more rapidly than when we were not using coarse-grid correction. Over 50 iterations, we see a reduction by a relative factor of 10^{10} in the 16×16 case and 10^{6} in the 32×32 case. Again, we have near-zero values for the velocity and a smooth pressure in our solution, as shown in Figure 10 and Figure 11. Whereas before in Figures 6 and 6 we saw a large range of values, relative to the scale, over which pressure varied, we now see that some of the smooth part of the error has been reduced too, yielding a much flatter solution than we had before. This implies that the coarse-grid correction step is indeed useful for solving this system.

At last we turn to looking at the full recursive v-cycle multigrid algorithm for solving $\mathcal{A}x = 0$ with a random initial guess. The results are much the same as for the two-grid algorithm. Figure 12 shows that the convergence factor is again near 1. After 50 iterations we see a convergence factor near 0.70451 for the 16×16 case and near 0.86184 for the 32×32



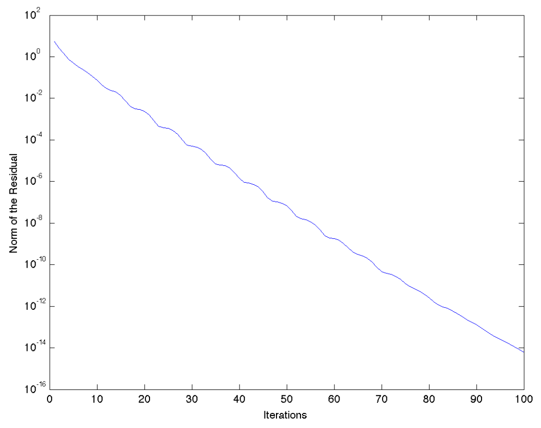


Figure 9: Two-grid solve of $\mathcal{A}x = 0$ on a 16×16 grid on the left and a 32×32 grid on the right for 100 iterations with a random intial guess. The results show that the norm of the residuals in both cases more rapidly approaches 0 compared to just the smoother.

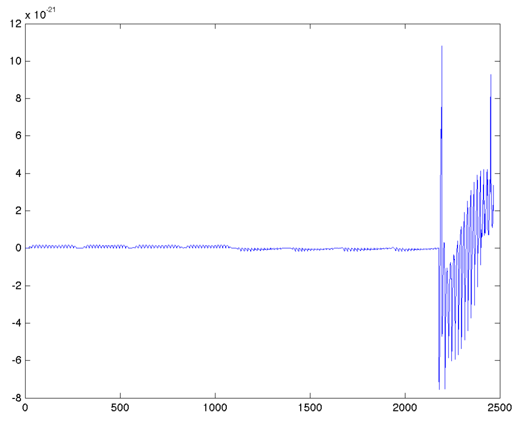
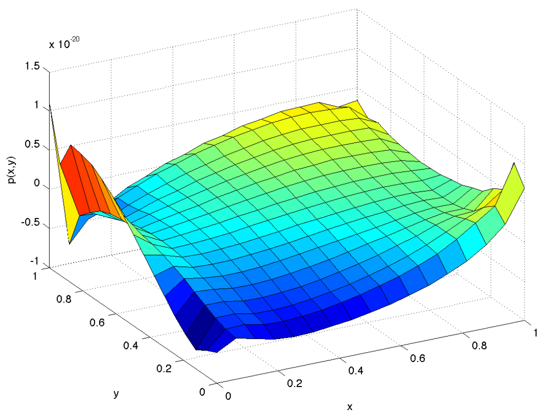



Figure 10: Solution after 100 two-grid iterations on a 16×16 grid of $\mathcal{A}x = 0$ with a random inital guess. On the left is a plot of the vector representing the solution. On the right is a surface plot of the pressure solution. We see that the velocity degrees of freedom are very uniformly at zero and while the pressure appears oscillatory, it is actually smooth.

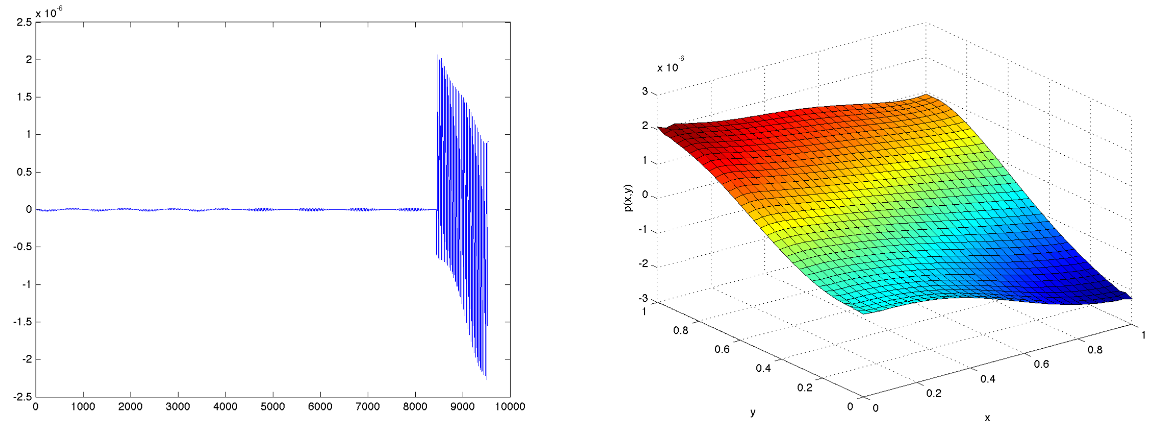


Figure 11: Solution after 100 two-grid iterations on a 32×32 grid of $\mathcal{A}x = 0$ with a random inital guess. On the left is a plot of the vector representing the solution. On the right is a surface plot of the pressure solution. We see that the velocity degrees of freedom are very uniformly at zero and while the pressure appears oscillatory, it is actually smooth.

case. Figure 13 shows that the norm of the residuals is again approaching zero very quickly. After 50 iterations, the norm of the residual has been reduced by a relative factor of 10^{10} in the 16×16 case and 10^4 in the 32×32 case. The solutions are again very near zero for the velocity and smooth and relatively flat for pressure (Figures 14 and 15).

These results show that we have found an effective algorithm for solving this linear system. Now we will run tests on the complete system

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

set up so that we know what solution to expect. We consider the Stokes equation with

$$\vec{u}(x, y) = (-\cos(2\pi x)\sin(2\pi y), \sin(2\pi x)\cos(2\pi y))^T,$$

$$p(x, y) = \pi \cos(2\pi xy)$$

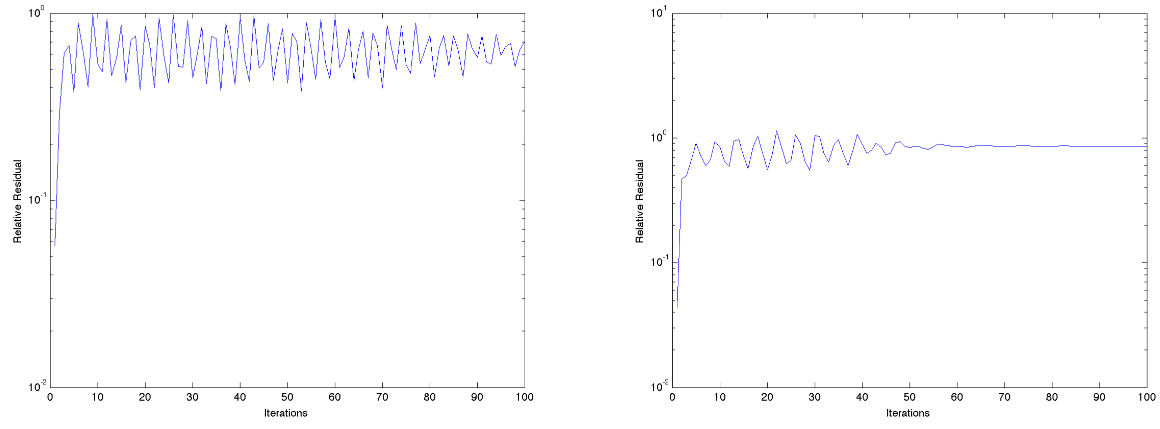


Figure 12: V-cycle solve of $\mathcal{A}x = 0$ on a 16×16 grid on the left and a 32×32 grid on the right for 100 iterations with a random intial guess. The results show that the relative residuals in both cases very rapidly approach 1.

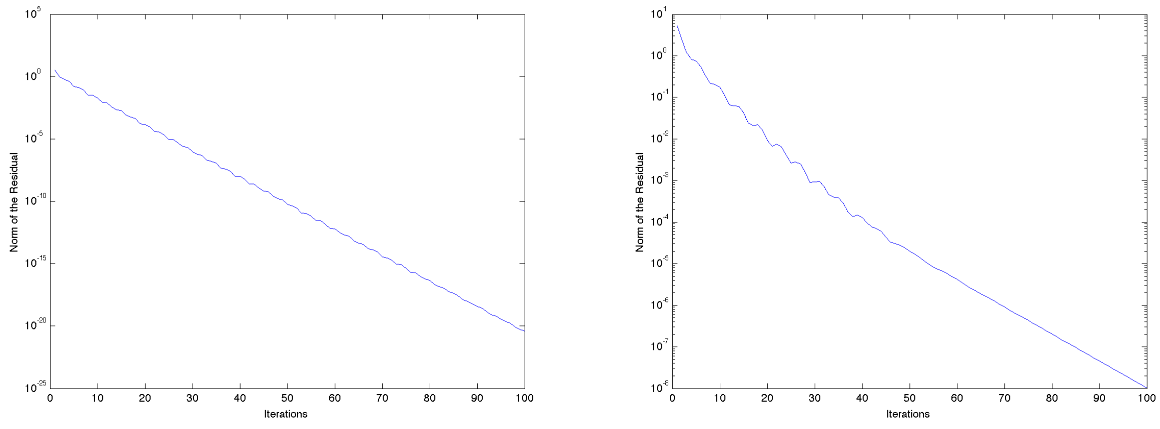


Figure 13: V-cycle solve of $\mathcal{A}x = 0$ on a 16×16 grid on the left and a 32×32 grid on the right for 100 iterations with a random intial guess. The results show that the norm of the residuals in both cases more rapidly approaches 0 compared to just the smoother.

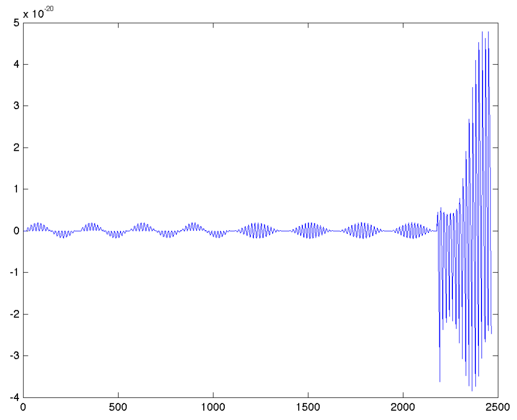
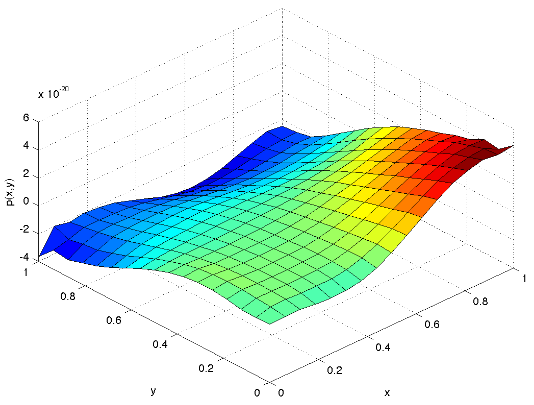



Figure 14: Solution after 100 v-cycle iterations on a 16×16 grid of $\mathcal{A}x = 0$ with a random inital guess. On the left is a plot of the vector representing the solution. On the right is a surface plot of the pressure solution. We see that the velocity degrees of freedom are very uniformly at zero and while the pressure appears oscillatory, it is actually smooth. As we can see, the solution has become flatter than when we ran only the smoother.

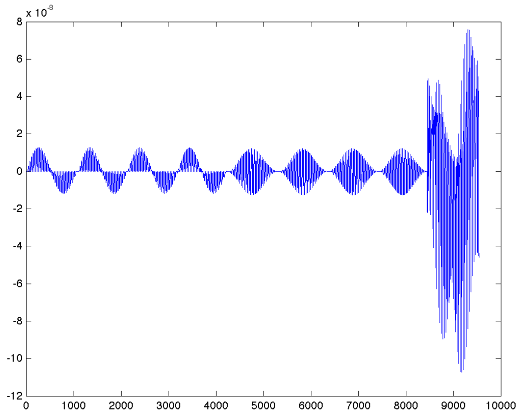
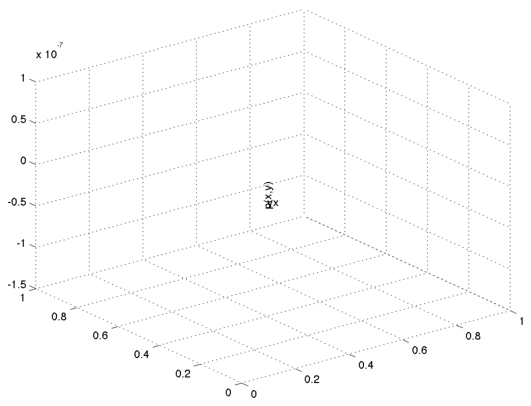



Figure 15: Solution after 100 v-cycle iterations on a 32×32 grid of $\mathcal{A}x = 0$ with a random inital guess. On the left is a plot of the vector representing the solution. On the right is a surface plot of the pressure solution. We see that the velocity degrees of freedom are very uniformly at zero and while the pressure appears oscillatory, it is actually smooth. As we can see, the solution has become flatter than when we ran only the smoother.

and find the appropriate forcing function $\vec{\xi}$. First note that

$$\nabla \cdot \vec{u} = 0.$$

Then we see that

$$-\nabla^2 \vec{u}(x,y) = (-8\pi^2 \cos(2\pi x)\sin(2\pi y), 8\pi^2 \cos(2\pi y)\sin(2\pi x))^T$$

and

$$\nabla p = (-2\pi^2 y \sin(2\pi xy), -2\pi^2 x \sin(2\pi xy)).$$

Summing these gives

$$\vec{\xi} = (-8\pi^2 \cos(2\pi x)\sin(2\pi y) - 2\pi^2 y \sin(2\pi xy), 8\pi^2 \cos(2\pi y)\sin(2\pi x) - 2\pi^2 x \sin(2\pi xy))^T.$$

We then use this $\vec{\xi}$ when computing the right hand side of our matrix equation and solve using the two-grid and multigrid v-cycle schemes.

We begin with the two-grid case. We see from Figure 16 that the relative residuals in this case again are close to one. For the 16×16 case, the convergence factor is approximately 1.0811 and for the 32×32 case, it is approximately 0.7389. This is about what we expect compared to the value of 0.77 given in [10]. Figure 17 shows that the norm of the residuals are very quickly converging to zero. We see final convergence of the 16×16 problem at about 10^{-14}, where as the 32×32 problem is slightly slower to converge. The final solutions here are not visually different than those resulting from the V-cycle, so we will wait to show them until we demonstrate with the v-cycle..

We turn finally to the full v-cycle algorithm. We see similar results to those shown previously. Figure 18 shows the relative residuals behave as they have before, rapidly approaching

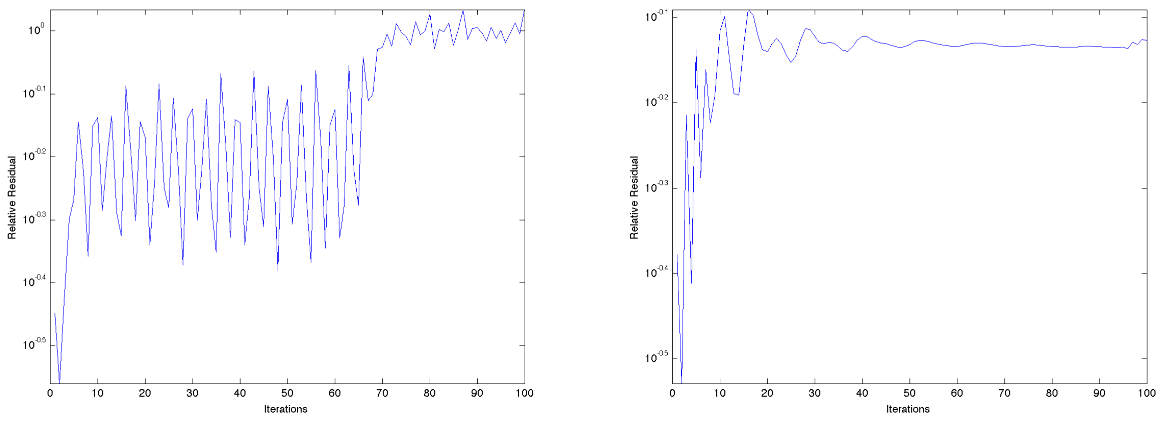


Figure 16: Two-grid solve of $\mathcal{A}x = b$ on a 16×16 grid on the left and a 32×32 grid on the right for 100 iterations with a zero intial guess. The results show that the relative residuals in both cases very rapidly approach a value near 1.

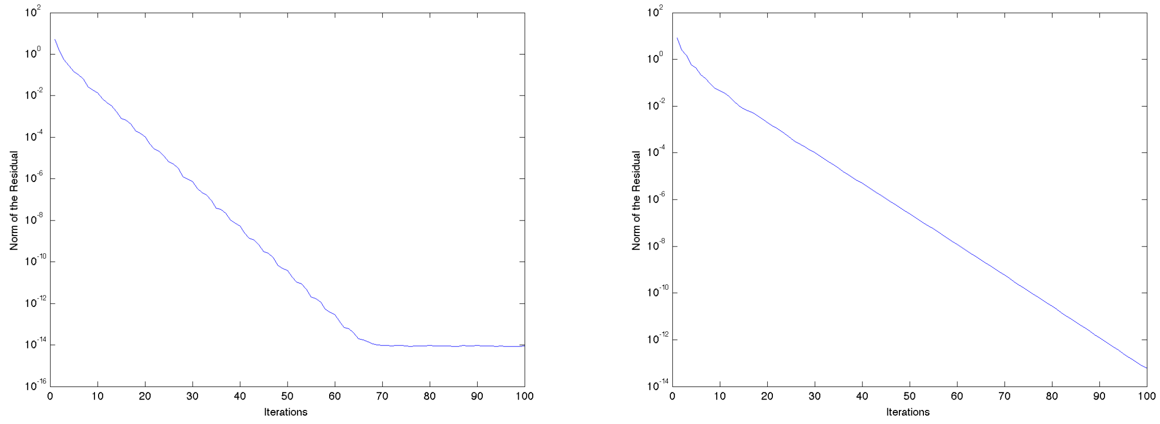


Figure 17: Two-grid solve of $\mathcal{A}x = b$ on a 16×16 grid on the left and a 32×32 grid on the right for 100 iterations with a zero intial guess. The results show that the norm of the residuals in both cases very rapidly approaches a value near 0 and seems to level off at approximately 10^{-14} in the 16×16 case and is still decreasing after 100 iterations in the 32×32 case.

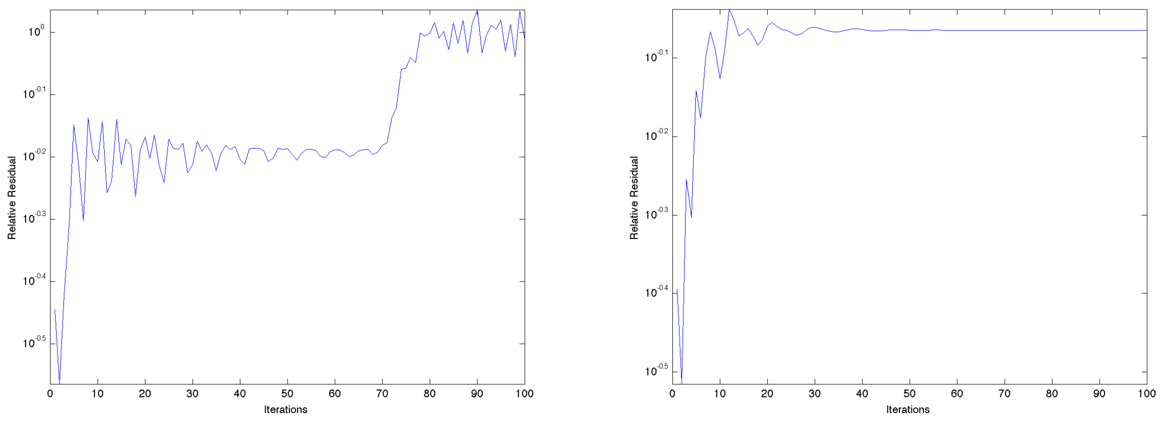


Figure 18: V-cycle solve of $\mathcal{A}x = b$ on a 16×16 grid on the left and a 32×32 grid on the right for 100 iterations with a zero intial guess. The results show that the relative residuals in both cases very rapidly approach a value near 1.

a value near to one, namely 0.97392 in the 16×16 case and 0.86161 in the 32×32 case. The beauty of the results here is in the solution. Recalling that the fact that we have Dirchlet boundary conditions on the entire boundary ensures a unique pressure solution up to a constant, we see from Figure 20 that the result perfectly corresponds to what we knew we ought to get.

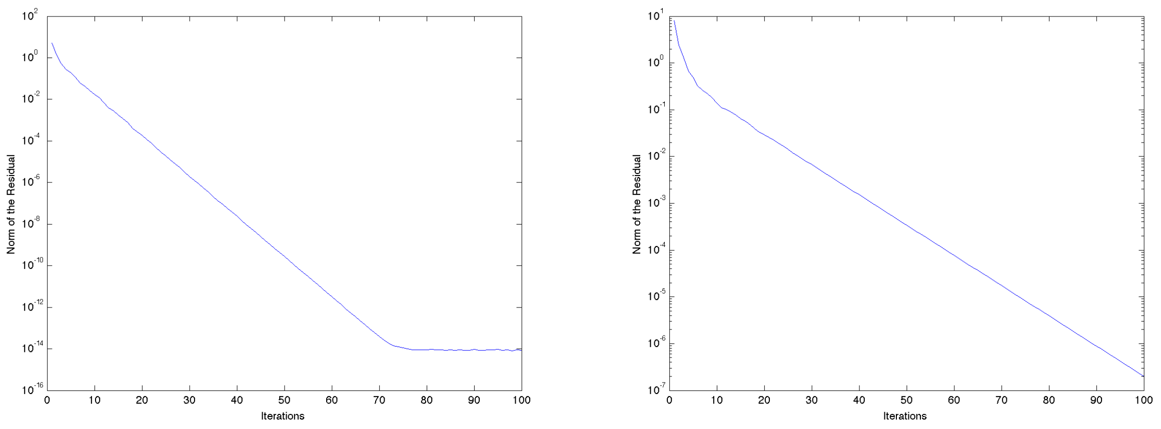


Figure 19: V-cycle solve of $\mathcal{A}x = b$ on a 16×16 grid on the left and a 32×32 grid on the right for 100 iterations with a zero intial guess. The results show that the norm of the residuals in both cases rapidly tends toward a value near 0. After 100 iterations, the residual in the 16×16 case has converged on the order of about 10^{-14}.

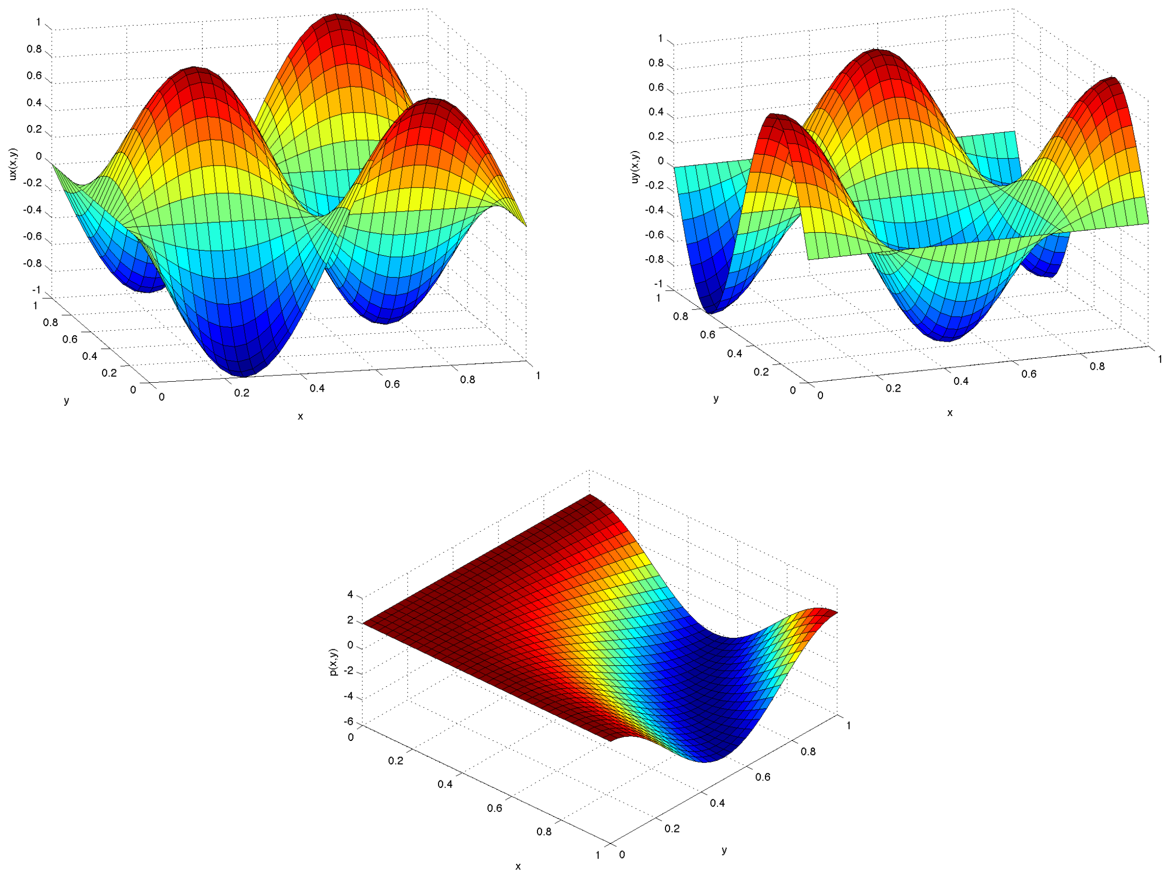


Figure 20: Results of the v-cycle solve of $\mathcal{A}x = b$ on a 32×32 grid. In the top left is the x-component of velocity. In the top right is the y-component of velocity. On the bottom is the pressure. We see we get exactly what we expect.

4 Maxwell's Equations

Now we move on to Maxwell's equations. The background on these equations largely comes from [11]. There are four equations:

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \times \mathbf{E} = 0, \tag{4.1}$$

$$\nabla \cdot \mathbf{D} = \rho, \tag{4.2}$$

$$\frac{\partial \mathbf{D}}{\partial t} - \nabla \times \mathbf{H} = -\mathbf{J}_t, \tag{4.3}$$

$$\nabla \cdot \mathbf{B} = 0. \tag{4.4}$$

Here we have that $\mathbf{E}$ is the electric field intensity and $\mathbf{H}$ is the magnetic field intensity. $\mathbf{B}$ is called the magnetic induction and $\mathbf{D}$ is called the electric displacement. These will soon be eliminated from the characterization of the electromagnetic field. Furthermore, $\mathbf{J}_t$ is the vector total current density function while ρ is a scalar charge density function. We can simplify (4.1)-(4.4) immediately since we are dealing with inhomogeneous, isotropic materials. This means that we have

$$\hat{\mathbf{D}} = \epsilon \hat{\mathbf{E}} \quad \text{and} \quad \hat{\mathbf{B}} = \mu \hat{\mathbf{H}}.$$

We further make the assumption that

$$\hat{\mathbf{J}}_t = \sigma \hat{\mathbf{E}} + \hat{\mathbf{J}}, \tag{4.5}$$

where $\hat{\mathbf{J}}$ is the applied current density and σ is the constant conductivity. Now we can simplify (4.1)-(4.4) by eliminating equations (4.2) and (4.4). We're also going to drop the hats on the variables. This leaves us with two only two equations:

$$\nabla \times \mathbf{E} + \mu \frac{\partial \mathbf{H}}{\partial t} = 0 \tag{4.6}$$

$$\nabla \times \mathbf{H} - \sigma \mathbf{E} - \epsilon \frac{\partial \mathbf{E}}{\partial t} = \mathbf{J}. \tag{4.7}$$

We can simplify further by eliminating one of **E** or **H** to get a single equation. We will eliminate **H** to get

$$\nabla \times \frac{1}{\mu} \nabla \times \mathbf{E} + \sigma \frac{\partial \mathbf{E}}{\partial t} + \epsilon \frac{\partial^2 \mathbf{E}}{\partial t} = -\frac{\partial \mathbf{J}}{\partial t}. \tag{4.8}$$

We further assume that **E** and **J** can be written as $\mathbf{E} = \hat{\mathbf{E}} e^{-i\omega t}$ and $\mathbf{J} = \hat{\mathbf{J}} e^{-i\omega t}$ to get

$$\nabla \times \frac{1}{\mu} \nabla \times \hat{\mathbf{E}} - i\omega\sigma\hat{\mathbf{E}} - \omega^2 \epsilon \hat{\mathbf{E}} = i\omega\hat{\mathbf{J}}. \tag{4.9}$$

Writing $\hat{\sigma} = \sigma - i\omega\epsilon$ this reduces to

$$\nabla \times \frac{1}{\mu} \nabla \times \hat{\mathbf{E}} - i\omega\hat{\sigma}\hat{\mathbf{E}} = i\omega\hat{\mathbf{J}}. \tag{4.10}$$

It is this equation that we will want to solve for the electric field $\hat{\mathbf{E}}$ given a specific current density $\hat{\mathbf{J}}$.

4.1 Discretization using Yee's Scheme

To discretize these equation, Schlumberger requested that we use what is called Yee's scheme. The basis of this scheme is imposing a mesh on a 3 dimensional, rectangular domain and then using a typical finite-difference scheme to discretize Equation (4.10). In our case, we use an irregular grid near to the boundary of the domain with grid spacing gradually decreasing toward the interior where it becomes regular. The final result of our discretization will have electric field degrees of freedom defined on the midpoints of cell edges. The x-component of

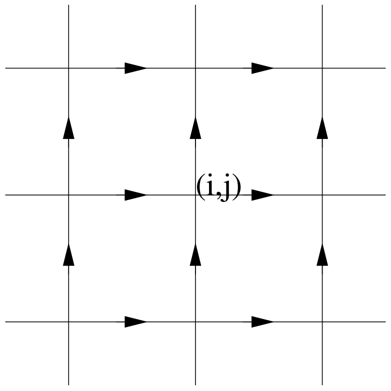


Figure 21: The placement of discrete degrees of freedom for the 2D staggered finite-difference approximation of $\nabla^{\perp}\mathbf{E}$. $E^{(1)}$ is defined at nodes marked with a right-facing arrow, $E^{(2)}$ is defined at nodes with an upward-facing arrow, while $\partial_y E^{(1)}$ and $\partial_x E^{(2)}$ are naturally defined at the cell centers.

the electric field will be defined on the x-edges of cells, the y-component on the y-edges, and the z-component on the z-edges.

4.1.1 Yee's Scheme in 2D

In 2D, Yee's scheme can be seen as a staggered finite-difference discretization of the grad-perp operator (defining $\nabla^{\perp}\mathbf{E} = -\partial_y E^{(1)} + \partial_x E^{(2)}$, where $\mathbf{E} = \left(\begin{smallmatrix} E^{(1)} \\ E^{(2)} \end{smallmatrix}\right)$). The components of the vector field, $\mathbf{E}$, are discretized on a staggered grid, with the x-component of the field, $E^{(1)}$, defined at the midpoints of the x-edges in the mesh, and the y-component, $E^{(2)}$, defined at the midpoints of the y-edges in the mesh. See Figure 21. With this grid placing, the grad-perp of a vector field, $\mathbf{E}$, is naturally defined at the cell centers, with formula

$$\left(\nabla^{\perp}\mathbf{E}\right)_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{E^{(2)}_{i+1,j+\frac{1}{2}} - E^{(2)}_{i,j+\frac{1}{2}}}{h_x} - \frac{E^{(1)}_{i+\frac{1}{2},j+1} - E^{(1)}_{i+\frac{1}{2},j}}{h_y}.$$

The 2D analogue of Equation (4.10) is

$$-\left(\nabla^{\perp}\right)^{T}\frac{1}{\mu}\nabla^{\perp}\mathbf{E} + \imath\omega\sigma\mathbf{E} = \imath\omega\mathbf{J}, \tag{4.11}$$

as the grad-perp operator is skew-self-adjoint in the L^2 inner product (in contrast to the curl, which is self-adjoint). So, to complete the discretization, we need to map $\frac{1}{\mu}\nabla^\perp \mathbf{E}$ from the cell centers back onto the edge-based degrees of freedom. Applying the finite-difference version of $-\left(\nabla^\perp\right)^T = \left(\begin{smallmatrix} \partial_y \\ -\partial_x \end{smallmatrix}\right)$ to this scalar function then naturally defines a vector field with x-component along the x-edges and y-component along the y-edges. For a scalar function, f, defined on a discrete grid at the cell centers,

$$\begin{aligned}
(\partial_y f)_{i+\frac{1}{2},j} &= \frac{f_{i+\frac{1}{2},j+\frac{1}{2}} - f_{i+\frac{1}{2},j-\frac{1}{2}}}{h_y} \\
(-\partial_x f)_{i,j+\frac{1}{2}} &= \frac{f_{i-\frac{1}{2},j+\frac{1}{2}} - f_{i+\frac{1}{2},j+\frac{1}{2}}}{h_x}.
\end{aligned}$$

Substituting the discrete function, $f = \frac{1}{\mu}\nabla^\perp \mathbf{E}$, into these definitions gives

$$\begin{aligned}
\left(-\left(\nabla^\perp\right)^T \frac{1}{\mu}\left(\nabla^\perp \mathbf{E}\right)\right)^{(1)}_{i+\frac{1}{2},j} =& \frac{1}{h_x h_y}\left(\frac{1}{\mu_{i+\frac{1}{2},j+\frac{1}{2}}}\left(E^{(2)}_{i+1,j+\frac{1}{2}} - E^{(2)}_{i,j+\frac{1}{2}}\right) - \frac{1}{\mu_{i+\frac{1}{2},j-\frac{1}{2}}}\left(E^{(2)}_{i+1,j-\frac{1}{2}} - E^{(2)}_{i,j-\frac{1}{2}}\right)\right) \\
&+ \frac{1}{h_y^2}\left(-\frac{1}{\mu_{i+\frac{1}{2},j+\frac{1}{2}}}E^{(1)}_{i+\frac{1}{2},j+1} + \left(\frac{1}{\mu_{i+\frac{1}{2},j+\frac{1}{2}}} + \frac{1}{\mu_{i+\frac{1}{2},j-\frac{1}{2}}}\right)E^{(1)}_{i+\frac{1}{2},j} - \frac{1}{\mu_{i+\frac{1}{2},j-\frac{1}{2}}}E^{(1)}_{i+\frac{1}{2},j-1}\right) \\
\left(-\left(\nabla^\perp\right)^T \frac{1}{\mu}\left(\nabla^\perp \mathbf{E}\right)\right)^{(2)}_{i,j+\frac{1}{2}} =& \frac{1}{h_x h_y}\left(\frac{1}{\mu_{i-\frac{1}{2},j+\frac{1}{2}}}\left(E^{(1)}_{i-\frac{1}{2},j} - E^{(1)}_{i-\frac{1}{2},j+1}\right) 1 \frac{1}{\mu_{i+\frac{1}{2},j+\frac{1}{2}}}\left(E^{(1)}_{i+\frac{1}{2},j+1} - E^{(1)}_{i+\frac{1}{2},j}\right)\right) \\
&+ \frac{1}{h_x^2}\left(-\frac{1}{\mu_{i-\frac{1}{2},j+\frac{1}{2}}}E^{(2)}_{i-1,j+\frac{1}{2}} + \left(\frac{1}{\mu_{i+\frac{1}{2},j+\frac{1}{2}}} + \frac{1}{\mu_{i-\frac{1}{2},j+\frac{1}{2}}}\right)E^{(2)}_{i,j+\frac{1}{2}} - \frac{1}{\mu_{i+\frac{1}{2},j+\frac{1}{2}}}E^{(2)}_{i+1,j+\frac{1}{2}}\right).
\end{aligned}$$

If σ and $\mathbf{J}$ are known at the mid-points of the edges, then the discrete analogue of Equation (4.11) is completely specified by these differences.

4.1.2 Yee's Scheme in 3D

To extend this scheme to 3D, first notice that the calculation in the previous section gives the z-component of the curl of a two-dimensional field, $\mathbf{E} = \left(\begin{smallmatrix} E^{(1)} \\ E^{(2)} \\ 0 \end{smallmatrix}\right)$, and then its contribution to the three-dimensional curl of $\frac{1}{\mu}\nabla \times \mathbf{E}$. We will mimic this approach in 3D, defining the

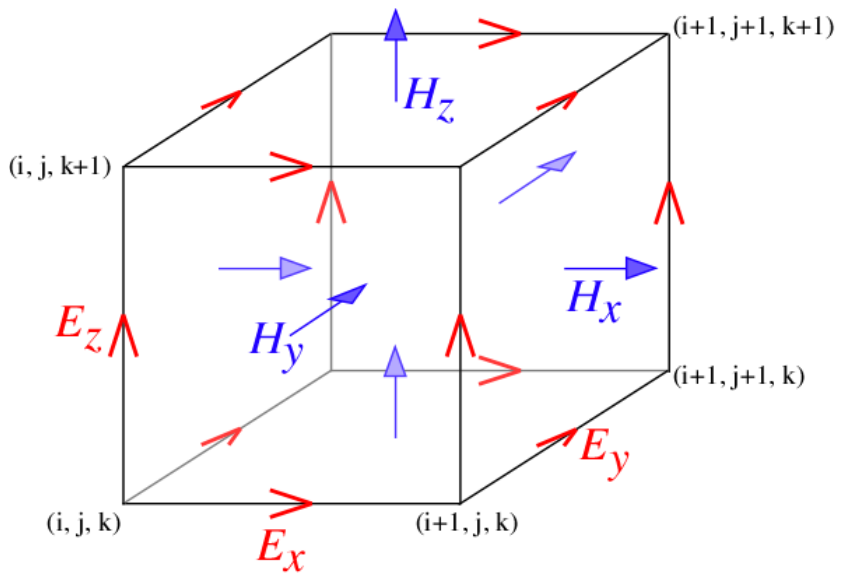


Figure 22: The placement of discrete degrees of freedom for the 3D staggered finite-difference approximation of $\nabla \times \mathbf{E}$. Figure courtesy Steven G. Johnson `http://ab-initio.mit.edu/wiki/index.php/Yee_lattices`. Note difference in notation, $\mathbf{E} = (E_x, E_y, E_z)^T$, $\nabla \times \mathbf{E} = (H_x, H_y, H_z)^T$

discrete values of $\mathbf{E}$ on the edges of a 3D mesh, which will define discrete values of $\nabla \times \mathbf{E}$ at the mid-points of the faces of the mesh (where μ will need to be defined). Then, $\nabla \times \frac{1}{\mu} \nabla \times \mathbf{E}$ will be defined again along the cell edges, where σ and $\hat{\mathbf{J}}$ will be defined.

We start with a field defined on the edges of the mesh, as shown in Figure 22. Using standard finite differences, we derive

$$\begin{aligned}
(\nabla \times \mathbf{E})^{(1)}_{i,j+\frac{1}{2},k+\frac{1}{2}} &= \frac{E^{(3)}_{i,j+1,k+\frac{1}{2}} - E^{(3)}_{i,j,k+\frac{1}{2}}}{h_y} - \frac{E^{(2)}_{i,j+\frac{1}{2},k+1} - E^{(2)}_{i,j+\frac{1}{2},k}}{h_z} \\
(\nabla \times \mathbf{E})^{(2)}_{i+\frac{1}{2},j,k+\frac{1}{2}} &= \frac{E^{(1)}_{i+\frac{1}{2},j,k+1} - E^{(1)}_{i+\frac{1}{2},j,k}}{h_z} - \frac{E^{(3)}_{i+1,j,k+\frac{1}{2}} - E^{(3)}_{i,j,k+\frac{1}{2}}}{h_x} \\
(\nabla \times \mathbf{E})^{(3)}_{i+\frac{1}{2},j+\frac{1}{2},k} &= \frac{E^{(2)}_{i+1,j+\frac{1}{2},k} - E^{(2)}_{i,j+\frac{1}{2},k}}{h_x} - \frac{E^{(1)}_{i+\frac{1}{2},j+1,k} - E^{(1)}_{i+\frac{1}{2},j,k}}{h_y}.
\end{aligned}$$

Similarly, given $\vec{F} = \frac{1}{\mu} \nabla \times \mathbf{E}$, defined at the face centers, the discrete curl of $\vec{F}$ is given by

$$\begin{aligned}
\left(\nabla \times \vec{F}\right)^{(1)}_{i+\frac{1}{2},j,k} &= \frac{F^{(3)}_{i+\frac{1}{2},j+\frac{1}{2},k} - F^{(3)}_{i+\frac{1}{2},j-\frac{1}{2},k}}{h_y} - \frac{F^{(2)}_{i+\frac{1}{2},j,k+\frac{1}{2}} - F^{(2)}_{i+\frac{1}{2},j,k-\frac{1}{2}}}{h_z} \\
\left(\nabla \times \vec{F}\right)^{(2)}_{i,j+\frac{1}{2},k} &= \frac{F^{(1)}_{i,j+\frac{1}{2},k+\frac{1}{2}} - F^{(1)}_{i,j+\frac{1}{2},k-\frac{1}{2}}}{h_z} - \frac{F^{(3)}_{i+\frac{1}{2},j+\frac{1}{2},k} - F^{(3)}_{i-\frac{1}{2},j+\frac{1}{2},k}}{h_x} \\
\left(\nabla \times \vec{F}\right)^{(3)}_{i,j,k+\frac{1}{2}} &= \frac{F^{(2)}_{i+\frac{1}{2},j,k+\frac{1}{2}} - F^{(2)}_{i-\frac{1}{2},j,k+\frac{1}{2}}}{h_x} - \frac{F^{(1)}_{i,j+\frac{1}{2},k+\frac{1}{2}} - F^{(1)}_{i,j-\frac{1}{2},k+\frac{1}{2}}}{h_y}.
\end{aligned}$$

Putting these together gives

$$
\begin{aligned}
&\left(\nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{E}\right)\right)^{(1)}_{i+\frac12,j,k} = \\
&\quad \frac{1}{h_y^j}\left(-\frac{1}{h_y^{j+\frac12}\mu^{(3)}_{i+\frac12,j+\frac12,k}}E^{(1)}_{i+\frac12,j+1,k} + \left(\frac{1}{h_y^{j+\frac12}\mu^{(3)}_{i+\frac12,j+\frac12,k}} + \frac{1}{h_y^{j-\frac12}\mu^{(3)}_{i+\frac12,j-\frac12,k}}\right)E^{(1)}_{i+\frac12,j,k} - \frac{1}{h_y^{j-\frac12}\mu^{(3)}_{i+\frac12,j-\frac12,k}}E^{(1)}_{i+\frac12,j-1,k}\right) \\
&\quad + \frac{1}{h_z^k}\left(-\frac{1}{h_z^{k+\frac12}\mu^{(2)}_{i+\frac12,j,k+\frac12}}E^{(1)}_{i+\frac12,j,k+1} + \left(\frac{1}{h_z^{k+\frac12}\mu^{(3)}_{i+\frac12,j,k+\frac12}} + \frac{1}{h_z^{k-\frac12}\mu^{(3)}_{i+\frac12,j,k-\frac12}}\right)E^{(1)}_{i+\frac12,j,k} - \frac{1}{h_z^{k-\frac12}\mu^{(3)}_{i+\frac12,j,k-\frac12}}E^{(1)}_{i+\frac12,j,k-1}\right) \\
&\quad + \frac{1}{h_x^{i+\frac12}h_y^j}\left(\frac{1}{\mu^{(3)}_{i+\frac12,j+\frac12,k}}\left(E^{(2)}_{i+1,j+\frac12,k} - E^{(2)}_{i,j+\frac12,k}\right) - \frac{1}{\mu^{(3)}_{i+\frac12,j-\frac12,k}}\left(E^{(2)}_{i+1,j-\frac12,k} - E^{(2)}_{i,j-\frac12,k}\right)\right) \\
&\quad + \frac{1}{h_x^{i+\frac12}h_z^k}\left(\frac{1}{\mu^{(2)}_{i+\frac12,j,k+\frac12}}\left(E^{(3)}_{i+1,j,k+\frac12} - E^{(3)}_{i,j,k+\frac12}\right) - \frac{1}{\mu^{(2)}_{i+\frac12,j,k-\frac12}}\left(E^{(3)}_{i+1,j,k-\frac12} - E^{(3)}_{i,j,k-\frac12}\right)\right) \\
&\left(\nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{E}\right)\right)^{(2)}_{i,j+\frac12,k} = \frac{1}{h_x^i h_y^{j+\frac12}}\left(\frac{1}{\mu^{(3)}_{i+\frac12,j+\frac12,k}}\left(E^{(1)}_{i+\frac12,j+1,k} - E^{(1)}_{i+\frac12,j,k}\right) - \frac{1}{\mu^{(3)}_{i-\frac12,j+\frac12,k}}\left(E^{(1)}_{i-\frac12,j+1,k} - E^{(1)}_{i-\frac12,j,k}\right)\right) \\
&\quad + \frac{1}{h_x^i}\left(-\frac{1}{h_x^{i+\frac12}\mu^{(3)}_{i+\frac12,j+\frac12,k}}E^{(2)}_{i+1,j+\frac12,k} + \left(\frac{1}{h_x^{i+\frac12}\mu^{(3)}_{i+\frac12,j+\frac12,k}} + \frac{1}{h_x^{i-\frac12}\mu^{(3)}_{i-\frac12,j+\frac12,k}}\right)E^{(2)}_{i,j+\frac12,k} - \frac{1}{h_x^{i-\frac12}\mu^{(3)}_{i-\frac12,j+\frac12,k}}E^{(1)}_{i-1,j+\frac12,k}\right) \\
&\quad + \frac{1}{h_z^k}\left(-\frac{1}{h_z^{k+\frac12}\mu^{(1)}_{i,j+\frac12,k+\frac12}}E^{(2)}_{i,j+\frac12,k+1} + \left(\frac{1}{h_z^{k+\frac12}\mu^{(1)}_{i,j+\frac12,k+\frac12}} + \frac{1}{h_z^{k-\frac12}\mu^{(1)}_{i,j+\frac12,k-\frac12}}\right)E^{(2)}_{i,j+\frac12,k} - \frac{1}{h_z^{k-\frac12}\mu^{(1)}_{i,j+\frac12,k-\frac12}}E^{(2)}_{i,j+\frac12,k-1}\right) \\
&\quad + \frac{1}{h_y^{j+\frac12}h_z^k}\left(\frac{1}{\mu^{(1)}_{i,j+\frac12,k+\frac12}}\left(E^{(3)}_{i,j+1,k+\frac12} - E^{(3)}_{i,j,k+\frac12}\right) - \frac{1}{\mu^{(1)}_{i,j+\frac12,k-\frac12}}\left(E^{(3)}_{i,j+1,k-\frac12} - E^{(3)}_{i,j,k-\frac12}\right)\right) \\
&\left(\nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{E}\right)\right)^{(3)}_{i,j,k+\frac12} = \frac{1}{h_x^i h_z^{k+\frac12}}\left(\frac{1}{\mu^{(2)}_{i+\frac12,j,k+\frac12}}\left(E^{(1)}_{i+\frac12,j,k+1} - E^{(1)}_{i+\frac12,j,k}\right) - \frac{1}{\mu^{(2)}_{i-\frac12,j,k+\frac12}}\left(E^{(1)}_{i-\frac12,j,k+1} - E^{(1)}_{i-\frac12,j,k}\right)\right) \\
&\quad + \frac{1}{h_y^j h_z^{k+\frac12}}\left(\frac{1}{\mu^{(1)}_{i,j+\frac12,k+\frac12}}\left(E^{(2)}_{i,j+\frac12,k+1} - E^{(2)}_{i,j+\frac12,k}\right) - \frac{1}{\mu^{(1)}_{i,j-\frac12,k+\frac12}}\left(E^{(2)}_{i,j-\frac12,k+1} - E^{(2)}_{i,j-\frac12,k}\right)\right) \\
&\quad + \frac{1}{h_x^i}\left(-\frac{1}{h_x^{i+\frac12}\mu^{(2)}_{i+\frac12,j,k+\frac12}}E^{(3)}_{i+1,j,k+\frac12} + \left(\frac{1}{h_x^{i+\frac12}\mu^{(2)}_{i+\frac12,j,k+\frac12}} + \frac{1}{h_x^{i-\frac12}\mu^{(2)}_{i-\frac12,j,k+\frac12}}\right)E^{(3)}_{i,j,k+\frac12} - \frac{1}{h_x^{i-\frac12}\mu^{(2)}_{i-\frac12,j,k+\frac12}}E^{(3)}_{i-1,j,k+\frac12}\right) \\
&\quad + \frac{1}{h_y^j}\left(-\frac{1}{h_y^{j+\frac12}\mu^{(1)}_{i,j+\frac12,k+\frac12}}E^{(3)}_{i,j+1,k+\frac12} + \left(\frac{1}{h_y^{j+\frac12}\mu^{(1)}_{i,j+\frac12,k+\frac12}} + \frac{1}{h_y^{j-\frac12}\mu^{(1)}_{i,j-\frac12,k+\frac12}}\right)E^{(3)}_{i,j,k+\frac12} - \frac{1}{h_y^{j-\frac12}\mu^{(1)}_{i,j-\frac12,k+\frac12}}E^{(3)}_{i,j-1,k+\frac12}\right).
\end{aligned}
$$

Here, we take $h_x^{i+\frac12}$, $h_y^{j+\frac12}$, and $h_z^{k+\frac12}$ to be the mesh widths of the elements between nodes i and $i+1$ in x, for example, while h_x^i, h_y^j, and h_z^k are the distances from the center (in x)

of element $i-1$ to the center (in x) of element i, across the plane of the i^{th} node in x, for example. These values are related as $h_x^i = \frac{1}{2}\left(h_x^{i-\frac{1}{2}} + h_x^{i+\frac{1}{2}}\right)$.

To symmetrize these equations, we scale each row by a volume factor, giving

$$
\begin{aligned}
&h_x^{i+\frac12} h_y^j h_z^k \left(\nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{E}\right)\right)^{(1)}_{i+\frac12,j,k} = \\
&\quad h_x^{i+\frac12} h_z^k \left(-\frac{1}{h_y^{j+\frac12}\mu^{(3)}_{i+\frac12,j+\frac12,k}} E^{(1)}_{i+\frac12,j+1,k} + \left(\frac{1}{h_y^{j+\frac12}\mu^{(3)}_{i+\frac12,j+\frac12,k}} + \frac{1}{h_y^{j-\frac12}\mu^{(3)}_{i+\frac12,j-\frac12,k}}\right) E^{(1)}_{i+\frac12,j,k} - \frac{1}{h_y^{j-\frac12}\mu^{(3)}_{i+\frac12,j-\frac12,k}} E^{(1)}_{i+\frac12,j-1,k} \right) \\
&+ h_x^{i+\frac12} h_y^j \left(-\frac{1}{h_z^{k+\frac12}\mu^{(2)}_{i+\frac12,j,k+\frac12}} E^{(1)}_{i+\frac12,j,k+1} + \left(\frac{1}{h_z^{k+\frac12}\mu^{(3)}_{i+\frac12,j,k+\frac12}} + \frac{1}{h_z^{k-\frac12}\mu^{(3)}_{i+\frac12,j,k-\frac12}}\right) E^{(1)}_{i+\frac12,j,k} - \frac{1}{h_z^{k-\frac12}\mu^{(3)}_{i+\frac12,j,k-\frac12}} E^{(1)}_{i+\frac12,j,k-1} \right) \\
&+ h_z^k \left(\frac{1}{\mu^{(3)}_{i+\frac12,j+\frac12,k}} \left(E^{(2)}_{i+1,j+\frac12,k} - E^{(2)}_{i,j+\frac12,k}\right) - \frac{1}{\mu^{(3)}_{i+\frac12,j-\frac12,k}} \left(E^{(2)}_{i+1,j-\frac12,k} - E^{(2)}_{i,j-\frac12,k}\right) \right) \\
&+ h_y^j \left(\frac{1}{\mu^{(2)}_{i+\frac12,j,k+\frac12}} \left(E^{(3)}_{i+1,j,k+\frac12} - E^{(3)}_{i,j,k+\frac12}\right) - \frac{1}{\mu^{(2)}_{i+\frac12,j,k-\frac12}} \left(E^{(3)}_{i+1,j,k-\frac12} - E^{(3)}_{i,j,k-\frac12}\right) \right) \\
&h_x^i h_y^{j+\frac12} h_z^k \left(\nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{E}\right)\right)^{(2)}_{i,j+\frac12,k} = h_z^k \left(\frac{1}{\mu^{(3)}_{i+\frac12,j+\frac12,k}} \left(E^{(1)}_{i+\frac12,j+1,k} - E^{(1)}_{i+\frac12,j,k}\right) - \frac{1}{\mu^{(3)}_{i-\frac12,j+\frac12,k}} \left(E^{(1)}_{i-\frac12,j+1,k} - E^{(1)}_{i-\frac12,j,k}\right) \right) \\
&+ h_y^{j+\frac12} h_z^k \left(-\frac{1}{h_x^{i+\frac12}\mu^{(3)}_{i+\frac12,j+\frac12,k}} E^{(2)}_{i+1,j+\frac12,k} + \left(\frac{1}{h_x^{i+\frac12}\mu^{(3)}_{i+\frac12,j+\frac12,k}} + \frac{1}{h_x^{i-\frac12}\mu^{(3)}_{i-\frac12,j+\frac12,k}}\right) E^{(2)}_{i,j+\frac12,k} - \frac{1}{h_x^{i-\frac12}\mu^{(3)}_{i-\frac12,j+\frac12,k}} E^{(1)}_{i-1,j+\frac12,k} \right) \\
&+ h_x^i h_y^{j+\frac12} \left(-\frac{1}{h_z^{k+\frac12}\mu^{(1)}_{i,j+\frac12,k+\frac12}} E^{(2)}_{i,j+\frac12,k+1} + \left(\frac{1}{h_z^{k+\frac12}\mu^{(1)}_{i,j+\frac12,k+\frac12}} + \frac{1}{h_z^{k-\frac12}\mu^{(1)}_{i,j+\frac12,k-\frac12}}\right) E^{(2)}_{i,j+\frac12,k} - \frac{1}{h_z^{k-\frac12}\mu^{(1)}_{i,j+\frac12,k-\frac12}} E^{(2)}_{i,j+\frac12,k-1} \right) \\
&+ h_x^i \left(\frac{1}{\mu^{(1)}_{i,j+\frac12,k+\frac12}} \left(E^{(3)}_{i,j+1,k+\frac12} - E^{(3)}_{i,j,k+\frac12}\right) - \frac{1}{\mu^{(1)}_{i,j+\frac12,k-\frac12}} \left(E^{(3)}_{i,j+1,k-\frac12} - E^{(3)}_{i,j,k-\frac12}\right) \right) \\
&h_x^i h_y^j h_z^{k+\frac12} \left(\nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{E}\right)\right)^{(3)}_{i,j,k+\frac12} = h_y^j \left(\frac{1}{\mu^{(2)}_{i+\frac12,j,k+\frac12}} \left(E^{(1)}_{i+\frac12,j,k+1} - E^{(1)}_{i+\frac12,j,k}\right) - \frac{1}{\mu^{(2)}_{i-\frac12,j,k+\frac12}} \left(E^{(1)}_{i-\frac12,j,k+1} - E^{(1)}_{i-\frac12,j,k}\right) \right) \\
&+ h_x^i \left(\frac{1}{\mu^{(1)}_{i,j+\frac12,k+\frac12}} \left(E^{(2)}_{i,j+\frac12,k+1} - E^{(2)}_{i,j+\frac12,k}\right) - \frac{1}{\mu^{(1)}_{i,j-\frac12,k+\frac12}} \left(E^{(2)}_{i,j-\frac12,k+1} - E^{(2)}_{i,j-\frac12,k}\right) \right) \\
&+ h_y^j h_z^{k+\frac12} \left(-\frac{1}{h_x^{i+\frac12}\mu^{(2)}_{i+\frac12,j,k+\frac12}} E^{(3)}_{i+1,j,k+\frac12} + \left(\frac{1}{h_x^{i+\frac12}\mu^{(2)}_{i+\frac12,j,k+\frac12}} + \frac{1}{h_x^{i-\frac12}\mu^{(2)}_{i-\frac12,j,k+\frac12}}\right) E^{(3)}_{i,j,k+\frac12} - \frac{1}{h_x^{i-\frac12}\mu^{(2)}_{i-\frac12,j,k+\frac12}} E^{(3)}_{i-1,j,k+\frac12} \right) \\
&+ h_x^i h_z^{k+\frac12} \left(-\frac{1}{h_y^{j+\frac12}\mu^{(1)}_{i,j+\frac12,k+\frac12}} E^{(3)}_{i,j+1,k+\frac12} + \left(\frac{1}{h_y^{j+\frac12}\mu^{(1)}_{i,j+\frac12,k+\frac12}} + \frac{1}{h_y^{j-\frac12}\mu^{(1)}_{i,j-\frac12,k+\frac12}}\right) E^{(3)}_{i,j,k+\frac12} - \frac{1}{h_y^{j-\frac12}\mu^{(1)}_{i,j-\frac12,k+\frac12}} E^{(3)}_{i,j-1,k+\frac12} \right).
\end{aligned}
$$

Note that the symmetry of the scaled equations is apparent. The connection from $E^{(1)}_{i+\frac{1}{2},j,k}$ to $E^{(1)}_{i+\frac{1}{2},j+1,k}$ is given by $\frac{-h_x^{i+\frac{1}{2}} h_z^k}{h_y^{j+\frac{1}{2}} \mu^{(3)}_{i+\frac{1}{2},j+\frac{1}{2},k}}$, while that to $E^{(1)}_{i+\frac{1}{2},j-1,k}$ is given by $\frac{-h_x^{i+\frac{1}{2}} h_z^k}{h_y^{j-\frac{1}{2}} \mu^{(3)}_{i+\frac{1}{2},j-\frac{1}{2},k}}$. Clearly a shift from j to $j \pm 1$ will equate these two coefficients. Similarly, the connection from $E^{(1)}_{i+\frac{1}{2},j,k}$ to $E^{(2)}_{i,j+\frac{1}{2},k}$ is $\frac{-h_z^k}{\mu^{(3)}_{i+\frac{1}{2},j+\frac{1}{2},k}}$, the same as the connection from $E^{(2)}_{i,j+\frac{1}{2},k}$ to $E^{(1)}_{i+\frac{1}{2},j,k}$.

4.2 The Smoother

After completing our finite-differences discretization using Yee's Scheme, the result is, as in the Stokes case after the finite-element discretization, a linear system that we need to solve. We will attempt a similar approach, namely an effective smoother coupled with a coarse-grid correction scheme. We will again consider a local, Vanka-type smoother.

The smoother for this equation uses much smaller local matrices than we used in the Stokes problem. In this case, we choose our localization index, j, to be the nodes in the 3D mesh. Therefore, we see that at a maximum, we only have connections with 6 entries in the matrix $\mathcal{A}_j$, namely those corresponding to the coefficients of $E^{(1)}_{i+\frac{1}{2},j,k}$, $E^{(1)}_{i-\frac{1}{2},j,k}$, $E^{(2)}_{i,j+\frac{1}{2},k}$, $E^{(2)}_{i,j-\frac{1}{2},k}$, $E^{(3)}_{i,j,k+\frac{1}{2}}$, and $E^{(3)}_{i,j,k-\frac{1}{2}}$. Thus our local matrices in this case are of dimension 6×6 for interior nodes, 5×5 for nodes on faces of our domain, 4×4 for nodes on edges of the domain, and 3×3 for the corner nodes. Whereas before with Stokes we had a regular grid and constant coefficients, here we have an irregular grid, which leads to variable coefficients. This prevents us from storing these matrices explicitly since they are different at different parts of the mesh.

As this is implemented in FORTRAN 90 code, we do not have the same sort of built-in routines that MATLAB provides. This complicates our implementation a little bit. We follow the same basic steps as with in the Stokes case, loopin this time over the nodes of our grid. First, we need to compute a local residual. We do this by using the finite-difference equations to compute only the relevant components of the residual. Then as before, we localize the current update. We then need to form the local matrix $\mathcal{A}_j$. Since we no longer have a built-in direct solver, we need to assemble this matrix in a way compatible with whatever solver we choose to use. Since we are working in FORTRAN 90, we are able to use a LAPACK solver [1]. Since the local matrix is complex-symmetric, the appropriate storage scheme for the matrix is a packed storage scheme where we only store the upper triangle (or, equivalently, the lower triangle) of the matrix. We then use the LAPACK routine to compute the update to the local solution and then redistribute to the global solution as before.

4.3 Results

At this time, the results for the test problems with this smoother are pending.

5 Conclusion

Partial differential equations (PDEs) play an important role in modelling physical phenomena. When these systems arise in nature, we look for ways to solve them in order to understand and explain our observations. Because these equations do not always have an analytic solution, we look for ways to solve them numerically. Fortunately, in many cases this also allows us to look at solutions that are "more discontinuous" than classical solutions might be. Since we are limited by computer technology, we must find ways to discretize these continuous equations and work in finite dimensions. This prevents us, in general, from finding the exact solution, but rather gives us an approximation to the solution. These discrete methods result in linear systems and we must then apply to tools of Linear Algebra to find solutions. We chose to use the Finite Element method to solve the Stokes equations and the Finite Differences method to solve Maxwell's equations.

In solving the Stokes equations, we saw that there is a block of zeros on the diagonal of the matrix. This prevented us from using traditional smoothers that we might use to solve, for example, Poisson's equation, such as (weighted) Jacobi or Gauss-Seidel methods. Thus we turned to a block-overlapping method with blocks indexed by the pressure nodes. We saw that this led to good, fast convergence both of the systems $\mathcal{A}x = 0$ and $\mathcal{A}x = b$. The next step in experiments with Stokes equations will be considering the case where viscosity may vary across elements and applying these same types of smoothers.

In solving Maxwell's equations, we again hope to use this sort of block-overlapping type smoother, this time indexed by the nodes on the mesh. For the case discussed above, we have not yet seen numerical results. Once we have numerical results for the smoother in this case, we will then implement a coarse-grid correction scheme and then consider a variable coefficient case as well.

Thus we have seen that in cases in which traditional (weighted) Jacobi or Gauss-Seidel might fail or converge poorly or exceedingly slowly, we have other options. Block-overlapping type methods can be shown to be very effective smoothers. Employing them with coarse-grid correction and multigrid schemes can be a very effective way to solve the systems that arise from discretizations of PDEs. Thus we can make progress in understanding and explaining the complexities of the natural world that we observe everyday.

References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide, Third Edition*, Philadelphia, PA, 1999.

[2] D. Arnold, R. Falk, and R. Winther, *Multigrid in $H(\mathrm{div})$ and $H(\mathrm{curl})$*, Numer. Math., 85 (2000), pp. 197–217.

[3] P. B. Bochev, C. J. Garasi, J. J. Hu, A. C. Robinson, and R. S. Tuminaro, *An improved algebraic multigrid method for solving Maxwell's equations*, Siam. J. Sci. Comput., 25 (2003), pp. 623–642.

[4] D. Braess and R. Sarazin, *An efficient smoother for the Stokes problem*, Applied Numerical Mathematics, 23 (1998), pp. 3–19.

[5] H. Elman, D. Silvester, and A. Wathen, *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*, Numerical Mathematics and Scientific Computation, Oxford University Press, New York, 2005.

[6] R. Hiptmair, *Multigrid method for Maxwell's equations*, SIAM J. Numer. Anal., 36 (1999), pp. 204–225.

[7] J. Hu, R. Tuminaro, P. Bochev, C. Garasi, and A. Robinson, *Toward an h-independent algebraic multigrid method for Maxwell's equations*, SIAM J. Sci. Comput., 27 (2006), pp. 1669–1688.

[8] V. John and G. Matthies, *Higher-order finite element discretizations in a benchmark problem for incompressible flows*, International Journal for Numerical Methods in Fluids, 37 (2001), pp. 885–903.

[9] M. Larin and A. Reusken, *A comparative study of efficient iterative solvers for generalized stokes equations*, Numerical Linear Algebra with Applications, 15 (2008), pp. 13–34.

[10] S. P. MacLachlan and C. W. Oosterlee, *A local Fourier analysis framework for finite-element discretizations of systems of PDEs*, SIAM J. Numer. Analysis, (2009). Submitted.

[11] P. Monk, *Finite Element Methods for Maxwell's Equations*, Oxford University Press, Oxford, 2003.

[12] W. Mulder, *Geophysical modelling of 3D electromagnetic diffusion with multigrid*, Comput. Visual. Sci., 11 (2008), pp. 129–138.

[13] J.-C. Nédélec, *Mixed finite elements in* $\mathbf{R}^3$, Numer. Math., 35 (1980), pp. 315–341.

[14] S. Reitzinger and J. Schöberl, *An algebraic multigrid method for finite element discretizations with edge elements*, Numer. Linear Alg. Appl., 9 (2002), pp. 223–238.

[15] U. Trottenberg, C. Oosterlee, and A. Schuller, *Multigrid*, Elsevier Academic Press, London, 2001.

[16] S. P. Vanka, *Block-implicit multigrid solution of Navier-Stokes equations in primitive variables*, J. Comput. Phys., 65 (1986), pp. 138–158.

[17] H. bin Zubair, S. MacLachlan, and C. Oosterlee, *A geometric multigrid method based on L-shaped coarsening for PDEs on stretched grids*, Numer. Linear Alg. Appl., (2009). To Appear.