Block-Structured Algorithms for Domain-Wall Fermions

Michael Perlmutter

May 14, 2010

Contents

0.1 Quantum Chromodynamics

The Standard Model of particle physics is the currently accepted theory for explaining three of the four fundamental forces of nature, the Strong Force, the Weak Force, and the Electro-Magnetic Force. Unifying the Standard Model with quantum models of Gravitation is among the most important areas of research in modern Physics. The Standard Model divides all elementary particles into two categories, fermions and bosons. Fermions that carry color charge are called quarks, and fermions, such as electrons, that do not carry color charge are called leptons. Bosons are the particles which mediate the action of forces. The bosons responsible for electromagnetism are called photons, the bosons which mediate the Strong Force are known as gluons, and the Weak Force is mediated by two sets of bosons known as z^0 and $w^{\pm}$ [3].

Quantum chromodynamics (QCD) is the part of the Standard Model of particle physics which models the Strong Force, which is responsible for binding together protons and neutrons, forming the nucleus of an atom. Moreover, the Strong Force also binds together quarks to form hadrons such as protons and neutrons. A particularly surprising property of QCD is confinement. For the other three forces, the interaction between two particles decreases by the inverse square law as the distance between the particles increases. The Strong Force, however, *increases* in magnitude as particles move farther apart.

QCD is generally believed to be true and is consistent with the results of all experiments so far conducted in particle physics. However, we still aim to use QCD to predict the results of future experiments for verification. Experiments to determine the mass and other physical properties of a large number of subatomic particles are currently underway. Carrying out these experiments requires particle accelerators, which are extremely expensive to build, run, and maintain. The Fermilab Tetravon cost \$120 million to build in 1983. More recently, in 2007, the Large Hadron Collider cost 3 billion Euro to construct. A proposed International Linear Collider is expected to cost roughly \$25 billion. For computational verification, a number of super computers are used such as APEnext and QCDOC. These computers are capable of carrying out roughly 10 trillion floating point operations per second (flops).

The aim of computational QCD is to predict the expected value of some observable quantity, $\mathcal{O}$, such as a mass or a velocity of some particle based off of the established laws of theoretical QCD. This computation is then to be compared to the experimental results as a means of testing the theory. Thus, our goal is to compute

$$\langle \mathcal{O}(A,\psi,\overline{\psi})\rangle = \frac{1}{Z}\int \mathcal{O}(A,\psi,\overline{\psi})e^{-S_{pg}-S_F}d\psi d\overline{\psi}dA,$$

where $\langle \mathcal{O}\rangle$ is the expected value of $\mathcal{O}$, $A = A_\mu(\vec{x}) \in \mathbb{C}^{3\times 3}$ is the gauge potential, $\psi(\vec{x})$ and $\overline{\psi}(\vec{x})$ are Grassmann-valued fermion fields, $Z = \int e^{-S_{pg}-S_F}d\psi d\overline{\psi}dA$, $S_{pg} = S_{pg}(A)$ is the "pure gauge" action, and $S_F = \int_{\vec{x},\vec{y}}\overline{\psi}(\vec{x})M_\rho(A)\psi(\vec{y})$ is the Fermionic action.

The most efficient Monte-Carlo techniques for computing the integral above require inverting $M_\rho(A)$, the Dirac Operator. We write the Dirac operator as a first-order system of 12 coupled PDEs posed on a four-dimensional space. The twelve variables (scalar-functional degrees of freedom), however, appear as a tensor-product of a four-dimensional space (associated with the quantum dynamical spin) with a three-dimensional space (associated with the quantum dynamical color).

The Dirac operator may be written as

$$M_\rho(A) = \sum_{\mu=1}^{4}(\gamma_\mu \otimes (I_3\partial_\mu - \imath A_\mu)) - \rho I_{12},$$

where $\mu = 1, \ldots, 4$ denote the four canonical space-time directions, $\{\gamma_\mu\}_{\mu=1}^4$ are four fixed unitary matrices with all entries 0,±1, and $\pm\imath$, ∂_μ is the standard partial derivative in the μ-direction, I_3 and I_{12} are the 3×3 and 12×12 identity matrices, respectively, the constant, ρ, is a mass parameter, and $\otimes$ denotes a standard tensor product of operators[7]. The field of complex 3×3 matrices, $A_\mu(\vec{x})$, is known as the gauge potential and is chosen through a Monte-Carlo process in the numerical simulation of QCD.

We will be working with a simplified version of this equation, the Schwinger model, in which we take space to be two-dimensional and take the potential, $a(\vec{x}, \mu)$, to be a scalar giving us

$$M_\rho(a) = \sum_{\mu=1}^{2} (\sigma_\mu \otimes (\partial_\mu - \imath a(\vec{x}, \mu)) - \rho I_2,$$

where σ_μ are Pauli matrices;

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \sigma_2 = \begin{pmatrix} 0 & -\imath \\ \imath & 0 \end{pmatrix},$$

and I_2 is the 2×2 identity matrix. Writing this in terms of the covariant partial derivative, $D_\mu = \partial_\mu - \imath a(\vec{x}, \mu)$, we can rewrite $M_\rho(a)$ as

$$M_\rho(a) = \begin{pmatrix} -\rho & D_1 - \imath D_2 \\ D_1 + \imath D_2 & -\rho \end{pmatrix}.$$

We will show later that the covariant partial derivative has a discrete approximation

$$D_\mu u(\vec{x}) \approx \frac{1}{2h} \left(e^{\imath\alpha(\vec{x}_i, \vec{x}_{i+1})} u(\vec{x}_{i+1}) - e^{-\imath\alpha(\vec{x}_{i-1}, \vec{x}_i)} u(\vec{x}_{i-1}) \right).$$

Moreover, we may approximate the second covariant partial derivative, $D_\mu^2 = \frac{\partial}{\partial\mu} D_\mu$, by

$$D_\mu^2 u(\vec{x}_i) \approx -\frac{1}{h^2} \left(-e^{-\imath\alpha(\vec{x}_{i-1}, \vec{x}_i)} u(\vec{x}_{i-1}) + 2u(\vec{x}_i) - e^{\imath\alpha(\vec{x}_i, \vec{x}_{i+1})} u(\vec{x}_{i+1}) \right).$$

Unfortunately, this discrete representation of the Schwinger (and Dirac) operator is unstable. Moreover, the Nielsen-Ninomiya "No Go" theorem shows that we cannot have sparse, stable discretization of the Dirac Operator which satisfies the correct physics. One way of resolving this is overlap, a method which accepts that our discretization will not be sparse. An alternative way to resolve the instability, is to add a scaled second covariant partial derivative to the diagonal of each matrix to get the Dirac-Wilson and Schwinger-Wilson models. The Schwinger-Wilson model is represented by

$$M_\rho(a) = \begin{pmatrix} -\frac{h}{2}(D_1^2 + D_2^2) - \rho & D_1 - \imath D_2 \\ D_1 + \imath D_2 & -\frac{h}{2}(D_1^2 + D_2^2) - \rho \end{pmatrix},$$

and the Dirac-Wilson model is represented by the analogous matrix in 4 dimensions. The scaling factor of $\frac{h}{2}$ is put in so that the first order and second-order covariant partial derivatives have the same scale and so that second-order terms go to zero as h goes to zero.

It is worth noting that the Schwinger model is by itself relevant to other areas of quantum dynamics such as quantum electrodynamics and graphene; however, we will be considering it as the simplification of the Dirac Model for the purposes of QCD. Unfortunately, the discretization of the the Dirac-Wilson operator still distorts certain physical properties of the system. To recreate these properties, we embed either the Dirac-Wilson matrix, or the Schwinger-Wilson matrix, into the domain-wall system,

$$W(m,\rho) = \begin{bmatrix} M_\rho & -P_- & & & +mP_+ \\ -P_+ & M_\rho & -P_- & & \\ & -P_+ & M_\rho & -P_- & \\ & & \ddots & \ddots & \ddots \\ +mP_- & & & -P_+ & M_\rho \end{bmatrix},$$

where

$$P_+ = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}, \qquad P_- = \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix}, \qquad m \ll 1.$$

Solving the domain wall system will yield the same results as solving the system resulting from overlap.

0.2 Finite Differences

We now consider the finite differences method, an important technique for numerically approximating the solution of a differential equation [9]. Consider an ordinary differential equation where a differential operator, L, acts on a real-valued function, $u(x)$, and produces another real-valued function, $f(x)$, $L(u(x)) = f(x)$. For our purposes, we will assume that $u(x)$ is periodic with period 1. However, differential equations with non-periodic solutions can be examined in a similar manner.

Our aim is to approximate this differential equation, $Lu = f$, with a system of linear equations of the form $Av = b$, where A is a square matrix and v and b are vectors. Defining a vector to approximate a function is rather straight forward. Choose an integer n and take v and b to be vectors representing u and f, respectively, defined by:

$$v_i = u\left(\frac{i}{n}\right) \qquad b_i = f\left(\frac{i}{n}\right) \qquad \text{for } 1 \leq i \leq n.$$

Figure 1 provides an example where $\sin(x)$ is discretized on a grid with 16 points.

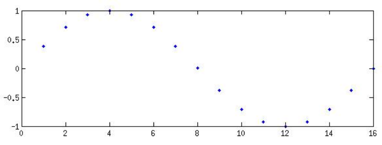


Figure 1: The discretization of $\sin(2\pi x)$, $n = 16$

We define x to be a node point if $x = \frac{i}{n}$ for some i and take h to be defined as $1/n$. It should be clear that v and b become increasingly good approximations as n becomes large. Note that all v_i arise from the value of u at a point in $(0, 1]$; however, this is okay since u is periodic. Thus, for any $x \in \mathbb{R}$, $u(x) = u(y)$ for some $y \in (0, 1]$.

We now have the more challenging task of approximating the differential operator L with a matrix A. The method for doing this will be illustrated by a series of examples starting with $-\frac{d^2u}{dx^2}(x) = f(x)$. It is worth noting that this is the one- dimensional version of Poisson's Equation, $-\nabla^2 u = f$, since the two-dimensional version of this equation, $(-\partial_{xx} - \partial_{yy})u(x, y) = f(x, y)$, will be explored later.

For a function, $u(x)$, with continuous derivatives, Taylor's theorem allows us to compute the values of u at a point near x using the value of u and its derivatives at x. We will assume that all terms with powers of h, 4 or higher, are negligible since large values of n will cause h to be very small and powers of h to be even smaller. Immediate consequences of Taylor's Theorem are

$$u(x+h) = u(x) + h\frac{du}{dx}(x) + \frac{h^2}{2}\frac{d^2u}{dx^2}(x) + \frac{h^3}{3!}\frac{d^3u}{dx^3}(x) + \frac{h^4}{4!}\frac{d^4u}{dx^4}(x) + \ldots,$$

and

$$u(x-h) = u(x) - h\frac{du}{dx}(x) + \frac{h^2}{2}\frac{\partial^2u}{dx^2}(x) - \frac{h^3}{3!}\frac{d^3u}{dx^3}(x) + \frac{h^4}{4!}\frac{d^4u}{dx^4}(x) - \ldots$$

Adding these two equations gives us

$$u(x+h) + u(x-h) = 2u(x) + h^2\frac{d^2u}{dx^2}(x) + O(h^4),$$

$$\Rightarrow f(x) = -\frac{d^2u}{dx^2}(x) = \frac{1}{h^2}[-u(x+h) + 2u(x) - u(x-h)] + O(h^2),$$

where $O(h^k)$ refers to an error the order of h^k. If x is a node point, and we treat the error as negligible, this equation becomes

$$b_i = n^2[-v_{i-1} + 2v_i - v_{i+1}].$$

We are trying to construct a matrix, A, such that $Av = b$. This implies that $A(i,:) \cdot v = b_i$ for all i, where $A(i,:)$ refers to the i^{th} row of A. Evaluating the dot product and plugging in for b_i gives us

$$\sum_{j=1}^{n} A_{i,j}v_j = n^2[-v_{i-1} + 2v_i - v_{i+1}].$$

This, of course, will happen when $A_{i,i} = 2n^2$, $A_{i,i\pm1} = -n^2$, and $A_{i,j} = 0$ otherwise. Thus, A will be a tri-diagonal matrix with $2n^2$ on the diagonal and $-n^2$ on the subdiagonal and superdiagonal. There are, however, two exceptions to this rule, which occur in the 1^{st} and n^{th} rows. When $j = 1$, $v_{j-1} = v_0$ is undefined. However, the periodicity of u allows use v_n in place of v_0 since $v_n = u(1) = u(0)$, which would be the definition of v_0 if we were to extend the definition of v to allow $i = 0$. Thus, we set $A_{1,n} = -n^2$. Similarly, in row n, we set $A_{n,1} = -n^2$. When $n = 16$, this gives us:

$$A = n^2 \begin{bmatrix}
2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
-1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2
\end{bmatrix}.$$

Solving the system $Av = b$ for v will give us a vector which approximates $u(x)$. Later, we will discuss efficient techniques for solving this system of equations.

Now let us consider the two dimensional version of Poisson's equation, $(-\partial_{xx} - \partial_{yy})u(x, y) = f(x, y)$. Our aim is again to represent the differential equation with a system of linear equations of the form $Av = b$. For the one dimensional problems, we let $v_i = u(\frac{i}{n})$. The natural way to adapt this for a two-dimensional problem would be to create a matrix, v, defined by the rule $v_{i,j} = u(\frac{i}{n}, \frac{j}{n})$. However, we need v to be a vector, not a square matrix. We resolve this by reshaping v into an n^2 by 1 vector using lexicographic ordering. For clarity, we will refer to the n by n matrix representing $u(x, y)$ as v^{sq} and the n^2 by 1 vector representing u will simply be referred to as v. Later, if v has superscripts as well, it will be assumed that v is a matrix if it has two subscripts and v is a vector if it has one subscript. The idea behind the reshaping is to have the first n entries of v be from the first column of v^{sq}, the next n entries from the second column of v^{sq} and so forth. More formally, this implies that $v_i = v^{sq}_{j,k}$ where $i = j + (k - 1)n$.

We also need a way to represent the differential equation $(-\partial_{xx} - \partial_{yy})u(x, y) = f(x, y)$ with a matrix. Since v and b will be n^2 by 1, we know that A must be n^2 by n^2. To define A in the same manner as in the one dimensional case, we need a higher dimensional analog of Taylor's theorem. If $u : \mathbb{R}^2 \to \mathbb{R}$ is a continuous periodic function with continuous partial derivatives and period 1, then for any particular $y \in \mathbb{R}$, the function $u^{(y)}(x) : \mathbb{R} \to \mathbb{R}$ defined by $u^{(y)}(x) = u(x, y)$ will be a continuous function with continuous derivatives. Applying Taylor's theorem to $u^{(y)}$ gives us

$$\begin{aligned} u^{(y)}(x+h) &= u^{(y)}(x) + h\frac{du^{(y)}}{dx}(x) + \frac{h^2}{2}\frac{d^2u^{(y)}}{dx^2}(x) + \frac{h^3}{3!}\frac{\partial^3 u^{(y)}}{\partial x^3}(x) + \frac{h^4}{4!}\frac{\partial^4 u^{(y)}}{\partial x^4}(x) + \dots \\ \text{and } u^{(y)}(x-h) &= u^{(y)}(x) - h\frac{du^{(y)}}{dx}(x) + \frac{h^2}{2}\frac{d^2u^{(y)}}{dx^2}(x) - \frac{h^3}{3!}\frac{\partial^3 u^{(y)}}{\partial x^3}(x) + \frac{h^4}{4!}\frac{\partial^4 u^{(y)}}{\partial x^4}(x) - \dots \end{aligned}$$

Solving for $\frac{d^2u^{(y)}}{dx^2}$ gives us

$$\frac{d^2u^{(y)}}{dx^2}(x) = \frac{1}{h^2}[-u^{(y)}(x+h) + 2u^{(y)}(x) - u^{(y)}(x-h)] + O(h^2).$$

By our construction of $u^{(y)}(x)$, the derivatives of $u^{(y)}(x)$ are the partial derivatives of $u(x, y)$ with respect to x. Thus,

$$\begin{aligned} u_{xx} &\cong \frac{1}{h^2}[u(x-h, y) - 2u(x, y) + u(x+h, y)]. \\ \text{Similarly,} \qquad u_{yy} &\cong \frac{1}{h^2}[u(x, y-h) - 2u(x, y) + u(x, y+h)]. \end{aligned}$$

Adding these equations, and multiplying by -1 gives us

$$f(x, y) = -(u_{xx} + u_{yy}) = \frac{-1}{h^2}[u(x-h, y) + u(x, y-h) - 4u(x, y) + u(x+h, y) + u(x, y+h)].$$

Assume that (x, y) is a node point, i.e. $(x, y) = (\frac{j}{n}, \frac{k}{n})$ for some integers j and k. Let $i = j + (k - 1)n$, then for most points,

$$\begin{aligned} &v_i = u(x, y), \qquad v_{i+1} = u(x+h, y), \qquad v_{i-1} = u(x-h, y), \\ &v_{i+n} = u(x, y+h), \qquad v_{i-n} = u(x, y-h), \qquad b_i = f(x, y). \end{aligned}$$

It is important to recognize that moving one column to the right in the matrix v^{sq} corresponds to moving n rows down the vector v. For example, $v^{sq}_{1,2} = v_{n+1}$. As in the one dimensional case, we want a matrix, A, such that $A(i,:) \cdot v = b_i$ for $1 \leq i \leq n^2$. This will occur if

$$\sum_{j=1}^{n^2} A_{i,j} v_j = n^2[-v_{i-1} - v_{i-n} + 4v_i - v_{i+1} - v_{i+n}].$$

Thus, in most rows,

$$A_{i,i} = 4 \qquad A_{i,i\pm 1} = A_{i,i\pm n} = -1, \qquad A_{i,j} = 0 \text{ otherwise.}$$

However, as in the one dimensional case, we must make modifications to account for boundary points. If $i \leq n$, then v_{i-n} is not defined. However, we still need an entry of v which represents $u(x, y-h)$. Since u is periodic,

$$u(x, y-h) = u(x, 1+y-h) = v_{n^2+i-n} \qquad \forall i \leq n.$$

Thus, when $i \leq n$, we set $A_{i,n^2-n+i} = -1$. Similarly, if $i + n > n^2$, we set $A_{i,i+n-n^2} = -1$.

Less obviously, we must also make modifications if i is divisible by n. Consider v_i, some entry of v. $v_i = u(x,y)$ for some $x, y \in (0,1]^2$. If i is divisible by n, then v_i corresponds to $v^{sq}_{n,k}$ for some $k \in \mathbb{Z}$. $v_{i+1} = v^{sq}_{1,k+1} \neq u(x+h, y)$. However, $v_i = v^{sq}_{n,k}$ implies that $x = 1$ by the way in which we constructed v and v_{sq}. Thus,

$$u(x+h, y) = u(1+h, y) = u(h, y) = v^{sq}_{1,k} = v_{i-n+1}.$$

Thus if $i \equiv 0 \mod n$, $A(i,;)$ is defined by the rule that

$$A_{i,i} = 4 \qquad A_{i,i-1} = A_{i,i-n+1} = A_{i,i\pm n} = -1.$$

Similarly, if $i \equiv 1 \mod n$, $A(i,;)$ is defined by

$$A_{i,i} = 4 \qquad A_{i,i+1} = A_{i,i+n-1} = A_{i,i\pm n} = -1.$$

In the case where $n^2 = 16$, this gives us:

$$A = n^2 \left[\begin{array}{cccc|cccc|cccc|cccc}
4 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
-1 & 0 & -1 & 4 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
\hline
-1 & 0 & 0 & 0 & 4 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & -1 & 0 & -1 & 4 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 4 & -1 & 0 & -1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & -1 & 4 & 0 & 0 & 0 & -1 \\
\hline
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 4 & -1 & 0 & -1 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & -1 & 4
\end{array}\right].$$

It is worth noting that A is singular, since all rows sum to zero, which implies that any constant vector in $\mathbb{R}^{16}$ is in the nullspace of A. Also, note that A has block representation,

$$A = n^2 \begin{bmatrix} P'_{1,4} & -I_4 & 0 & -I_4 \\ -I_4 & P'_{1,4} & -I_4 & 0 \\ 0 & -I_4 & P'_{1,4} & -I_4 \\ -I_4 & 0 & -I_4 & P'_{1,4} \end{bmatrix},$$

where $P'_{1,4}$ is the matrix representing the one dimensional Poisson operator for $n = 4$, with the n^2 factored out and 4 replacing 2 on the diagonal, and I_4 is the 4 by 4 indentity matrix.

The Kronecker product is an operation which takes an m by n matrix, A, and a p by q matrix, B, and produces a mp by nq matrix, $A \otimes B$, by the rule that

$$A \otimes B = \begin{bmatrix} a_{1,1}B & a_{1,2}B & \dots & a_{1,n}B \\ a_{2,1}B & \dots & \dots & a_{2,n}B \\ \vdots & & & \vdots \\ a_{m,1}B & \dots & \dots & a_{m,n}B \end{bmatrix},$$

where the $a_{i,j}$ refer the entries of A.

Using the Kronecker product, we may write A as

$$A = P_{1,4} \otimes I_4 + I_4 \otimes P_{1,4},$$

where $P_{1,4}$ is the actual matrix representation of the one dimensional Poisson Operator (not to be confused with $P'_{1,4}$ above).

Poisson's equation, $-\nabla^2 u(x, y) = f(x, y)$, is a special case of the Helmholtz equation, $(-\nabla^2 + k)u(x, y) = f(x, y)$, $k \in \mathbb{R}$, with $k = 0$. Since differential operators are linear, if A is a matrix which represents the Poisson operator, $-\nabla^2$, and B is a matrix which represents the "differential" operator, kI, then $A + B$ will be matrix representing the Helmholtz operator, $-\nabla^2 + kI$. It is clear that kI_{n^2} is the matrix which represents the differential operator kI since

$$(kI_{n^2})v = k(I_{n^2}v) = kv \qquad \forall v \in \mathbb{R}^{n^2},$$

Thus, the matrix, $A = A_{poisson} + kI_{n^2}$ (where $A_{poisson}$ is the matrix which represents the Poisson operator and I_{n^2} is the n^2 by n^2 identity matrix) represents the Helmholtz operator. When $n^2 = 16$,

$$A = n^2 \begin{bmatrix} 4+k & -1 & 0 & -1 & -1 & 0 & 0 & 0 & \dots \\ -1 & 4+k & -1 & 0 & 0 & -1 & 0 & 0 & \ddots \\ 0 & -1 & 4+k & -1 & 0 & 0 & -1 & 0 & \ddots \\ -1 & 0 & -1 & 4+k & 0 & 0 & 0 & -1 & \ddots \\ -1 & 0 & 0 & 0 & 4+k & -1 & 0 & -1 & \ddots \\ 0 & -1 & 0 & 0 & -1 & 4+k & -1 & 0 & \ddots \\ 0 & 0 & -1 & 0 & 0 & -1 & 4+k & -1 & \ddots \\ 0 & 0 & 0 & -1 & -1 & 0 & -1 & 4+k & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}.$$

In block form this becomes,

$$\begin{bmatrix} P'_{1,4}+kI_4 & -I_4 & 0 & -I_4 \\ -I_4 & P'_{1,4}+kI_4 & -I_4 & 0 \\ 0 & -I_4 & P'_{1,4}+kI_4 & -I_4 \\ -I_4 & 0 & -I_4 & P'_{1,4}+kI_4 \end{bmatrix}.$$

Nowhere did we use the fact that k was real valued, thus the same matrix will hold for a Helmholtz equation with a complex value of k.

Similar analysis allows us to represent ∂_x with a matrix. Two equivalent definitions of $\partial_x u(x,y)$ for a differentiable function, $u(x,y)$, are

$$\begin{aligned} \frac{\partial u}{\partial x}(x,y) &= \lim_{h\to 0}\frac{u(x+h,y)-u(x,y)}{h}, \\ \text{and } \frac{\partial u}{\partial x}(x,y) &= \lim_{h\to 0}\frac{u(x,y)-u(x-h,y)}{h}. \end{aligned}$$

Taking the average of these two definitions gives us

$$\frac{\partial u}{\partial x}(x,y) = \lim_{h\to 0}\frac{u(x+h,y)-u(x-h,y)}{2h}.$$

If (x,y) is a node point, this equation becomes

$$b_i = \left(\frac{1}{2h}\right)(v_{i+1}-v_{i-1}),$$

which implies $A_{i,i\pm 1} = \left(\pm\frac{1}{2h}\right)$ and $A_{i,j}=0$ otherwise. As with ∂_{xx}, we must modify this to account for the boundary when either $x \equiv 0 \mod n$ or $x \equiv 1 \mod n$. If $x \equiv 0 \mod n$, we replace $A_{i,i+1}$ with $A_{i,i-n+1}$ and if $x \equiv 1$, we replace $A_{i,i-1}$ with $A_{i,i+n-1}$. For $n=4$, this gives us

$$A = \frac{1}{2h}\left[\begin{array}{cccc|cccc|cccc|cccc}
0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0
\end{array}\right].$$

Now let us consider how to adapt this method for a system of Partial Differential Equations (PDEs). We use as an example the system

$$\begin{bmatrix} -(\partial_{xx}+\partial_{yy}) & -\partial_x \\ \partial_x & -(\partial_{xx}+\partial_{yy}) \end{bmatrix} \begin{bmatrix} u_1(x,y) \\ u_2(x,y) \end{bmatrix} = \begin{bmatrix} f_1(x,y) \\ f_2(x,y) \end{bmatrix},$$

where $u_1, u_2, f_1,$ and f_2 are functions from $\mathbb{R}^2$ to $\mathbb{R}$, and u_1 and u_2 have continuous partial derivatives and are periodic with period 1 in both the x and y directions. Let A and B be matrices representing $-\partial_{xx} - \partial_{yy}$ and ∂_x respectively, and let $v_1, v_2, b_1,$ and b_2 be vectors representing u_1, u_2, f_1 and f_2. We want to approximate the system of PDEs with a linear system of form $Sv = b$, where S is a matrix representing the system of differential operators, v is a vector representing u_1 and u_2, and b is is a vector representing f_1 and f_2. When we multiply out the differential system we get

$$\begin{bmatrix} -(\partial_{xx}+\partial_{yy})u_1(x,y) - \partial_x u_2(x,y) & = f_1(x,y) \\ \partial_x u_1(x,y) - (\partial_{xx}-\partial_{yy})u_2(x,y) & = f_2(x,y) \end{bmatrix}.$$

If we substitute in A, B, v_1, v_2, b_1 and b_2 we get

$$\begin{aligned} Av_1 + Bv_2 &= v_1 \\ -Bv_1 + Av_2 &= v_2. \end{aligned}$$

Thus, we let

$$S = \begin{bmatrix} A & B \\ -B & A \end{bmatrix}, \qquad v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

Then $Sv = b$ implies.

$$\begin{bmatrix} A & B \\ -B & A \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},$$

which is precisely the condition we want.

Now let us consider discretizing the covariant dartial derivative operator, $D_x = \partial_x - \imath A_x(x,y)$, where ∂_x is the standard partial derivative operator, $A_x(x,y)$ is the partial derivative with respect to x of a function, A, from $\mathbb{R}^2$ to $\mathbb{R}$ with continuous partial derivatives, and $\imath$ is the imaginary unit. Let $u(x,y)$ be a periodic function from $\mathbb{R}^2$ to $\mathbb{R}$ with continuous derivatives. For $1 \le j,k \le n$, let $(x_j, y_k) = \left(\frac{j}{n}, \frac{k}{n}\right)$, and let $h = 1/n$. If (x_j, y) is some node point, the Fundamental Theorem of Calculus implies that the average value of $\partial_x u(x,y)$ on the line between (x_j, y) and (x_{j+1}, y) is given by

$$\left(\frac{1}{h}\right) \int_{(x_j,y)}^{(x_{j+1},y)} \partial_x u(x,y) dx = \left(\frac{1}{h}\right) (u(x_{j+1},y) - u(x_j,y)).$$

By the product rule,

$$\begin{aligned} \partial_x(e^{-\imath A(x,y)}u(x,y)) &= \partial_x(u(x,y))e^{-\imath A(x,y)} - iA_x(x,y)e^{-\imath A(x,y)}u(x,y) \\ e^{\imath A(x,y)}\partial_x(e^{-\imath A(x,y)}u(x,y)) &= \partial_x(u(x,y)) - \imath A_x(x,y)u = D_x u(x,y). \end{aligned}$$

Thus, the average value of $D_x u(x,y)$ on the line between (x_j, y) and (x_{j+1}, y) is given by

$$\begin{aligned}\left(\frac{1}{h}\right)\int_{(x_j,y)}^{(x_{j+1},y)}(D_x u(x,y))dx &= \left(\frac{1}{h}\right)\int_{(x_j,y)}^{(x_{j+1},y)} e^{\imath A(x,y)}\partial_x(e^{-\imath A(x,y)}u)dx \\ &\approx \left(\frac{1}{h}\right) e^{\imath A(x',y)}\int_{(x_j,y)}^{(x_{j+1},y)} \partial_x(e^{-\imath A(x,y)u})dx \\ &= \left(\frac{1}{h}\right) e^{\imath A(x',y)}\left[e^{-\imath A(x_{j+1},y)}u(x_{j+1},y) - e^{-\imath A(x_j,y)}u(x_j,y)\right],\end{aligned}$$

where (x',y) is any point on the line between (x_j,y) and (x_{j+1},y). The approximation is justified because $e^{\imath A(x,y)}$ is the composition of continuous functions and is, therefore, itself continuous. Thus, on a sufficiently small interval, $e^{\imath A(x,y)}$ is approximately constant. If we choose n to be very large, the distance between (x_{j+1},y) and (x_j,y) will be very small since $||(x_{j+1},y)-(x_j,y)|| = h = 1/n$. Thus, we may factor our the "constant" to get our approximation. In fact, by the Mean Value Theorem for Integrals, the equality holds exactly for some x' in $[x_j, x_j+1]$.

Since they are the only two node points on the interval, x_j and x_{j+1} are the two natural choices of x'. We refer to the equation resulting from taking $x' = x_j$ as the left-hand approximation, (L), and to the equation resulting form taking $x' = x_{j+1}$ as the right-hand approximation, (R). If $x' = x_j$, we get

$$\left(\frac{1}{h}\right)[e^{\imath(A(x_j,y)-A(x_{j+1},y))}u(x_{j+1},y) - u(x_j,y)], \tag{L}$$

and if $x' = x_{j+1}$,

$$\left(\frac{1}{h}\right)[u(x_{j+1},y) - e^{\imath(A(x_{j+1},y)-A(x_j,y))}u(x_j,y)]. \tag{R}$$

If we take the right hand guess on the interval $[x_{j-1}, x_j]$ we get

$$\left(\frac{1}{h}\right)[u(x_j,y) - e^{\imath(A(x_j,y)-A(x_{j-1},y))}u(x_{j-1},y)]. \tag{R'}$$

L and R are approximations of the average value of $D_x u(x,y)$ on the interval $[x_j, x_{j+1}]$; R' is the analogous approximation of it's average value on $[x_{j-1}, x_j]$. If we average L and R' we get an approximation of the average value of $D_x u(x,y)$ on the interval $[x_{j-1}, x_{j+1}]$, which should approximately equal $D_x u(x_j,y)$. We choose L rather than R because both L and R' assume that $x' = x_j$. We refer to the average of L and R as the central approximation.

$$\left(\frac{1}{2h}\right)[e^{\imath(A(x_j,y)-A(x_{j+1},y))}u(x_{j+1},y) - e^{\imath(A(x_j,y)-A(x_{j-1},y))}u(x_{j-1},y)] \tag{C}$$

Note that if A is constant with respect to x this reduces to our approximation of the standard partial derivative $\partial_x u = \left(\frac{1}{2h}\right)(u(x_{j+1}) - u(x_{j-1}))$. This is to be expected, since if A is constant with respect to x,

$$D_x = \partial_x + \imath A_x(x,y) = \partial_x.$$

The matrix approximating the covariant partial derivative will be similar to the matrix representing the standard partial derivative, except $e^{\imath(A(x_j,y)-A(x_{j+1},y))}$ replaces -1 on the superdiagonal and $-e^{\imath(A(x_j,y)-A(x_{j-1},y))}$ replaces -1 on the subdiagonal. Thus, if we let $\alpha_j = A(x_{j+1},y) - A(x_j,y)$, the matrix representing D_x is given by

$$B_x = \frac{1}{2h}\left[\begin{array}{cccc|cccc|c}
0 & e^{-\imath\alpha_1} & 0 & -e^{\imath\alpha_4} & 0 & 0 & 0 & 0 & \cdots \\
-e^{\imath\alpha_1} & 0 & e^{-\imath\alpha_2} & 0 & 0 & 0 & 0 & 0 & \ddots \\
0 & -e^{\imath\alpha_2} & 0 & e^{-\imath\alpha_3} & 0 & 0 & 0 & 0 & \ddots \\
e^{-\imath\alpha_4} & 0 & -e^{\imath\alpha_3} & 0 & 0 & 0 & 0 & 0 & \ddots \\
\hline
0 & 0 & 0 & 0 & 0 & e^{-\imath\alpha_5} & 0 & -e^{\imath\alpha_8} & \ddots \\
0 & 0 & 0 & 0 & -e^{\imath\alpha_5} & 0 & e^{-\imath\alpha_6} & 0 & \ddots \\
0 & 0 & 0 & 0 & 0 & -e^{\imath\alpha_6} & 0 & e^{-\imath\alpha_7} & \ddots \\
0 & 0 & 0 & 0 & e^{-\imath\alpha_8} & 0 & -e^{\imath\alpha_7} & 0 & \ddots \\
\hline
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots
\end{array}\right].$$

Similarly, if $D_y u(x,y) = \partial_y - \imath A_y(x,y)$, and (x, y_j) is some node point, we can approximate $D_y u(x, y_j)$ by

$$\left(\frac{1}{2h}\right)[e^{\imath(A(x,y_j)-A(x,y_{j+1}))}u(x,y_{j+1}) - e^{\imath(A(x,y_j)-A(x,y_{j-1}))}u(x,y_{j-1})]$$

and represent D_y by $B_y =$

$$\frac{1}{2h}\left[\begin{array}{cccc|cccc|cccc|c}
0 & 0 & 0 & 0 & e^{-\imath\beta_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & 0 & e^{-\imath\beta_2} & 0 & 0 & 0 & 0 & 0 & 0 & \ddots \\
0 & 0 & 0 & 0 & 0 & 0 & e^{-\imath\beta_3} & 0 & 0 & 0 & 0 & 0 & \ddots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{-\imath\beta_4} & 0 & 0 & 0 & 0 & \ddots \\
\hline
-e^{\imath\beta_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{-\imath\beta_5} & 0 & 0 & 0 & \ddots \\
0 & -e^{\imath\beta_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{-\imath\beta_6} & 0 & 0 & \ddots \\
0 & 0 & -e^{\imath\beta_3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{-\imath\beta_7} & 0 & \ddots \\
0 & 0 & 0 & -e^{\imath\beta_4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{-\imath\beta_8} & \ddots \\
\hline
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots
\end{array}\right],$$

where $\beta_j = A(x, y_{j+1}) - A(x, y_j)$.

We define the second covariant partial derivative to be

$$D_x^2 = \frac{\partial D_x u(x,y)}{\partial x} = \lim_{h\to 0} \frac{D_x u(x+.5h,y) - D_x u(x-.5h,y)}{h}.$$

L gives an estimate of of $D_x u(x+.5h,y)$ and R' gives an estimate of $D_x u(x-.5h,y)$. Thus, $\frac{L-R'}{h}$ gives us the following estimate:

$$\begin{aligned}
D_x^2 &\simeq \frac{e^{\imath(A(x_j,y)-A(x_{j+1},y))}u(x_{j+1},y) - 2u(x_j,y) + e^{\imath(A(x_j,y)-A(x_{j-1},y))}u(x_{j-1},y)}{h^2} \\
&= \frac{e^{-\imath\alpha_j}u(x,y_{j+1}) - 2u(x,y_j) + e^{\imath\alpha_{j-1}}u(x,y_{j-1})}{h^2}.
\end{aligned}$$

Similarly,

$$
\begin{aligned}
D_y^2 &\simeq \frac{e^{\imath(A(x,y_j)-A(x,y_{j+1}))}u(x,y_{j+1}) - 2u(x,y_j) + e^{\imath(A(x,y_j)-A(x,y_{j-1}))}u(x,y_{j-1})}{h^2} \\
&= \frac{e^{-\imath\beta_j}u(x,y_{j+1}) - 2u(x,y_j) + e^{\imath\beta_{j-1}}u(x,y_{j-1})}{h^2}.
\end{aligned}
$$

As with the first covariant partial derivative, note that in the case that A is constant, these reduce to approximations of the standard second partial derivatives. Also, the matrices representing D_x^2 and D_y^2 will be similar to those representing ∂_{xx} and ∂_{yy}, respectively, with the off-diagonal entries replaced with the appropriate values of $e^{\imath\alpha}$ or $e^{\imath\beta}$. For example, a 1 in the matrix representation of ∂_{xx} corresponding to $u(x_{j+1}, y)$ is replaced by $e^{-\imath\alpha_j}$.

We define the covariant Helmholtz operator to as $H = -(D_x^2 + D_y^2) - k^2(b_1 + \imath b_2)$ for $k, b_1, b_2 \in \mathbb{R}$, $b_1 \geq 0$. Because both matrices and linear differential operators are linear, if A_x and A_y are the matrices representing D_x^2 and D_y^2, then the covariant Helmholtz operator is represented by $A_H = -(A_x + A_y - k^2(b_1 + \imath b_2)I_{n^2})$. Similar to the ordinary Helmholtz matrix, this has block representation

$$
\frac{1}{h^2}\begin{bmatrix} P_1 + \sigma I_4 & D_1^{(u)} & 0 & D_4^{(l)} \\ D_1^{(l)} & P_2 + \sigma I_4 & D_2^{(u)} & 0 \\ 0 & D_2^{(l)} & P_3 + \sigma I_4 & D_3^{(u)} \\ D_4^{(u)} & 0 & D_3^{(l)} & P_4 + \sigma I_4 \end{bmatrix},
$$

where

$$
P_j = \begin{bmatrix} 4 & -e^{-\imath\alpha_{c_j+1}} & 0 & -e^{\imath\alpha_{c_j+1}} \\ -e^{\imath\alpha_{c_j+1}} & 4 & -e^{-\imath\alpha_{c_j+2}} & 0 \\ 0 & -e^{\imath\alpha_{c_j+1}} & 4 & -e^{-\imath\alpha_{c_j+3}} \\ -e^{-\imath\alpha_{c_j+4}} & 0 & -e^{\imath\alpha_{c_j+1}} & 4 \end{bmatrix},
$$

$$
D_j^{(u)} = \begin{bmatrix} -e^{-\imath\beta_{c_j+1}} & 0 & 0 & 0 \\ 0 & -e^{-\imath\beta_{c_j+2}} & 0 & 0 \\ 0 & 0 & -e^{-\imath\beta_{c_j+3}} & 0 \\ 0 & 0 & 0 & -e^{-\imath\beta_{c_j+4}} \end{bmatrix},
$$

$$
D_j^{(l)} = \begin{bmatrix} -e^{-\imath\beta_{c_j}} & 0 & 0 & 0 \\ 0 & -e^{-\imath\beta_{c_j+1}} & 0 & 0 \\ 0 & 0 & -e^{-\imath\beta_{c_j+2}} & 0 \\ 0 & 0 & 0 & -e^{-\imath\beta_{c_j+3}} \end{bmatrix},
$$

with $c_j = (j-1)n$ and $\sigma = -k^2(b_1 + \imath b_2)$.

We let A_P be the matrix representing the covariant Poisson operator (the special case of A_H with $k = 0$) and let $S_A = \frac{h}{2}A_P$. The operator $D_x + \imath Dy$ is approximated by $S_B = B_x - \imath B_y$. We define the Schwinger matrix to be

$$
S = \begin{bmatrix} S_A & S_B \\ -\overline{S_B} & S_A \end{bmatrix},
$$

where $\overline{S_B} = B_x - \imath B_y$.

The Schwinger matrix is a discrete representative of an important system of partial differential equations in quantum chromodynamics. However, our discretization of the covariant partial derivatives distorts certain physical properties of the model. To recreate these physical properties, we embed the Schwinger matrix in the $2ln^2$ by $2ln^2$ domain wall operator

$$W(m,\rho) = \begin{bmatrix} M_\rho & -P_- & & & +mP_+ \\ -P_+ & M_\rho & -P_- & & \\ & -P_+ & M_\rho & -P_- & \\ & & \ddots & \ddots & \ddots \\ +mP_- & & & -P_+ & M_\rho \end{bmatrix},$$

where

$$M_\rho = S + \rho I_{2n^2}, \qquad P_+ = \begin{bmatrix} I_{n^2} & 0 \\ 0 & 0 \end{bmatrix}, \qquad P_+ = \begin{bmatrix} 0 & 0 \\ 0 & I_{n^2} \end{bmatrix}, \qquad m \ll 1.$$

0.3 Singular Value Decomposition

Now let us discuss the Singular Value Decomposition (SVD), an important technique of matrix factorization. SVD works for any m by n matrix, but, since we aim to apply it to the Domain Wall System, we will consider it for square n by n matrices with full rank. First, we recall some linear algebra. If, for an n by n matrix, A, and a scalar λ, there exists a non-zero vector v such that $Av = \lambda v$, then λ is an eigenvalue of A and v is an eigenvector of A corresponding to λ. For each λ, the associated eigenspace is the minimal vector space which contains all of the eigenvectors of A associated with λ. We say that V is an eigenvector matrix of A if the columns of V are linearly independent, each of the columns of V is an eigenvector of A, and for each eigenvalue, λ, the eigenspace associated with λ is contained in the span of the columns of V. If V is an eigenvector matrix, and the norm of each column of V is 1, then we say that V is a normalized eigenvector matrix. If A is Hermitian-symmetric, the columns of V are orthonormal, so V is unitary, i.e., $V^{-1} = V^T$. The rank of A refers the dimension of the span of the columns of A. If the rank of A is equal to n, we say that A has full rank. In the case that A has full rank, any eigenvector matrix of A will be the same size as A. For further review, see the text by Strang [8].

Let A be a square n by n matrix with full rank. We can factor A as

$$A = U\Sigma V^T,$$

where U is a normalized eigenvector matrix of AA^T, V is a normalized eigenvector matrix of A^TA, and Σ is a diagonal matrix, whose non-zero entries, $\sigma_1, \sigma_2, \ldots \sigma_n$, are the square roots of the eigenvalues of both AA^T and A^TA. We make Σ uniquely defined with the convention that $\sigma_1 \geq \sigma_2 \ldots \geq \sigma_n$.

Note that any factorization of A into $U\Sigma V^T$, with Σ diagonal and the same size as A, must be of this form since

$$AA^T = (U\Sigma V^T)(V\Sigma^T U^T) = U\Sigma^2 U^T \text{ and } A^TA = (V\Sigma^T U^T)(U\Sigma V^T) = V\Sigma^2 V^T$$
$$\Rightarrow AA^TU = U\Sigma^2 \text{ and } A^TAV = V\Sigma^2,$$

which implies that the columns of U and V are eigenvectors of AA^T and A^TA respectively. Moreover, if V is any eigenvector matrix of A^TA, then there will exist a U, such that U is an eigenvector matrix of AA^T and $A = U\Sigma V^T$, since for any column of V, $V^{(k)}$

$$A^TAv^{(k)} = \sigma_k^2 v^{(k)} \Rightarrow AA^TAv^{(k)} = A\sigma_k^2 v^{(k)} = \sigma_k^2 Av^{(k)}.$$

Thus, $Av^{(k)}$ is an eigenvector of AA^T, if we let $u^{(k)} = Av^{(k)}$ and $U = \left(u^{(1)}u^{(2)}\ldots u^{(n)}\right)$, then $AV = U\Sigma$, which implies that $A = U\Sigma V^T$.
Now let us consider a simple example.

$$\text{Let } A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \Rightarrow D = A^TA = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix},$$

λ will be an eigenvalue of A^TA if and only if there exists a non-zero vector, v, such that $Av = \lambda v = \lambda Iv \Leftrightarrow (A - \lambda Iv) = 0$ which will occur if and only if λ is a root of the characteristic polynomial, $|A - xI|$. For this example, the characteristic polynomial is given by:

$$|A^TA - \lambda I| = \begin{vmatrix} 2-\lambda & -1 & 0 & 0 \\ -1 & 2-\lambda & -1 & 0 \\ 0 & -1 & 2-\lambda & -1 \\ 0 & 0 & -1 & 1-\lambda \end{vmatrix} = \lambda^4 - 7\lambda^3 + 15\lambda^2 - 10\lambda + 1$$

and has approximate roots, $\lambda_1 = 3.53209$, $\lambda_2 = 2.473$, $\lambda_3 = 1$, and $\lambda_4 = 0.120615$.

The singular values, σ_1, σ_2, σ_3, and σ_4, are computed by taking the square roots of each λ, giving us $\sigma_1 \approx 1.8794$, $\sigma_2 \approx 1.5321$, $\sigma_3 = 1$, and $\sigma_4 \approx 0.3473$.

For each λ_j, we solve $A^TA - \lambda_j I = 0$ to find the associated eigenvector. Note that since the number of eigenvalues is equal to the number of columns in A^TA, the eigenspace of each eigenvalue must be one dimensional. Plugging in each λ_j and solving directly, we find that the eigenvectors of of A^TA are given by,

$$v_1 \approx \begin{pmatrix} -0.4285 \\ 0.6565 \\ -0.5774 \\ 0.2280 \end{pmatrix} v_2 \approx \begin{pmatrix} 0.6565 \\ -0.2280 \\ -0.5574 \\ 0.4285 \end{pmatrix} v_3 \approx \begin{pmatrix} 0.5774 \\ 0.5774 \\ 0 \\ 0.5774 \end{pmatrix} v_4 \approx \begin{pmatrix} -.2280 \\ -.4285 \\ -.5774 \\ -.6565 \end{pmatrix},$$

where v_j is the normalized eigenvector associated with λ_j.

We set $V = (v_1, v_2, v_3, v_4)$, and compute $U = AV\Sigma^{-1}$,

$$U = AV\Sigma^{-1} = \begin{bmatrix} -.2280 & .42850 & .5574 & -.6565 \\ .5773 & -.5773 & 0 & -.5773 \\ -.6565 & -.2280 & -.5574 & -.4285 \\ .4285 & .6565 & -.5774 & -.2280 \end{bmatrix}.$$

Now let us consider how to apply this to the domain wall matrix,

$$
\begin{aligned}
W(m,\rho) &= \begin{bmatrix} M_\rho & -P_- & & & +mP_+ \\ -P_+ & M_\rho & -P_- & & \\ & -P_+ & M_\rho & -P_- & \\ & & \ddots & \ddots & \ddots \\ +mP_- & & & -P_+ & M_\rho \end{bmatrix} \\
&\approx \begin{bmatrix} M_\rho & -P_- & & & \\ -P_+ & M_\rho & -P_- & & \\ & -P_+ & M_\rho & -P_- & \\ & & \ddots & \ddots & \ddots \\ & & & -P_+ & M_\rho \end{bmatrix} \\
&= \begin{bmatrix} M_{\rho-1} & & & & \\ & M_{\rho-1} & & & \\ & & M_{\rho-1} & & \\ & & & \ddots & \\ & & & & M_{\rho-1} \end{bmatrix} + \left[\begin{array}{cc|cc|cc|c} I & 0 & 0 & 0 & & & \\ 0 & I & 0 & -I & & & \\ \hline -I & 0 & I & 0 & 0 & 0 & \\ 0 & 0 & 0 & I & 0 & -I & \\ \hline & & -I & 0 & I & 0 & \\ & & 0 & 0 & 0 & I & \ddots \\ \hline & & & & & \ddots & \ddots \end{array}\right] \\
&= I_l \otimes M_{\rho-1} + D \otimes I_{n^2},
\end{aligned}
$$

where D is the $2l$ by $2l$ matrix given by

$$
D = \left[\begin{array}{cc|cc|cc|c} 1 & 0 & 0 & 0 & & & \\ 0 & 1 & 0 & -1 & & & \\ \hline -1 & 0 & 1 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 1 & 0 & -1 & \\ \hline & & -1 & 0 & 1 & 0 & \\ & & 0 & 0 & 0 & 1 & \ddots \\ \hline & & & & & \ddots & \ddots \end{array}\right].
$$

The Mixed-Product Property of Kronecker Product states that for any matrices, A, B, C, and D, $AC \otimes BD = (A \otimes B)(C \otimes D)$. In particular, $AB \otimes I = AB \otimes II = (A \otimes I)(B \otimes I)$. Therefore, if $U\Sigma V^T$ is a Singular Value Decomposition of D, then $\tilde{U}\tilde{\Sigma}\tilde{V}^T$ is a Singular Value Decomposition of $D \otimes I_{n^2}$, where $\tilde{U} = U \otimes I_{n^2}$, $\tilde{\Sigma} = \Sigma \otimes I_{n^2}$, and $\tilde{V} = V \otimes I_{n^2}$.

We initially examine the Singular Value Decomposition of D, with the hopes that it will lead us to a useful factorization of the entire domain wall system. Let $R = I_l \otimes M_{\rho-1}$, and suppose $D \otimes I_{n^2}$ has singular value decomposition $\tilde{U}\tilde{\Sigma}\tilde{V}^T$. If $\tilde{U}^T R\tilde{V}$ has a block diagonal structure similar to R, such as if $\tilde{U}^T R\tilde{V}$ is block diagonal, then so will be $\tilde{U}^T R\tilde{V} + \tilde{\Sigma}$, which, would allow us to decouple the domain wall system in the following manner,

$$
\begin{aligned}
W_\rho \mathbf{x} = \mathbf{b} &\Rightarrow (R + \tilde{U}\tilde{\Sigma}\tilde{V}^T)\mathbf{x} = \mathbf{b} \Rightarrow (\tilde{U}^T R + \tilde{U}^T\tilde{U}\tilde{\Sigma}\tilde{V}^T)\mathbf{x} = \tilde{U}^T\mathbf{b} \Rightarrow (\tilde{U}^T R\tilde{V}\tilde{V}^T + \tilde{\Sigma}\tilde{V}^T)\mathbf{x} = \tilde{U}^T\mathbf{b} \\
&\Rightarrow (\tilde{U}^T R\tilde{V} + \tilde{\Sigma})(\tilde{V}^T\mathbf{x}) = \tilde{U}^T\mathbf{b} \Rightarrow (\tilde{U}^T R\tilde{V} + \tilde{\Sigma})\mathbf{y} = \mathbf{c},
\end{aligned}
$$

where $\mathbf{y} = \tilde{V}^T\mathbf{x}$ and $\mathbf{c} = \tilde{U}^T\mathbf{b}$.

As in the example, the key to finding the Singular Value Decomposition of D is in finding the eigenvectors of D^TD. Direct computation shows that

$$D^TD = \begin{bmatrix} 2 & 0 & -1 & & & & & & \\ 0 & 1 & 0 & -1 & & & & & \\ -1 & 0 & 2 & 0 & -1 & & & & \\ & -1 & 0 & 2 & 0 & -1 & & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & -1 & 0 & 2 & 0 & -1 & \\ & & & & -1 & 0 & 2 & 0 & -1 \\ & & & & & -1 & 0 & 1 & 0 \\ & & & & & & -1 & 0 & 2 \end{bmatrix}.$$

Notice that every non-zero entry is an even number of columns off of the diagonal. Therefore, if we were to multiply any vector, v, by D^TD the even entries of $(D'D)v$ would be determined only by the even rows and columns of D^TD, and the odd entries of $(D^TD)v$ would be determined only by the odd rows of D^TD and columns. Thus we may decouple D^TD into two l by l matrices,

$$(D^TD)_o = \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix} \text{ and } (D^TD)_e = \begin{bmatrix} 1 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix}.$$

For any vector v with n entries, define $\tilde{v}$ by the rule that $\tilde{v}_i = v_{n+1-i}$. That is if $v = (v_1, v_2, \ldots, v_n)^T$, then $\tilde{v} = (v_n, v_{n-1}, \ldots, v_1)^T$. Note that if v is an eigenvector of $(D^TD)_e$ with eigenvalue λ, then $\tilde{v}$ is eigenvector of $(D^TD)_o$ with eigenvalue λ. Moreover, $v \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ and $\tilde{v} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ are eigenvectors of D^TD with eigenvalue λ. Therefore, if we know the eigenvalues and eigenvectors of $(D^TD)_e$, we will know the eigenvectors and eigenvalues of D^TD.

Theorem: For all $1 \le k \le l, v^{(k)}$ defined by $v_j^{(k)} = \frac{\sin(j\theta^{(k)}) - \sin((j-1)\theta^{(k)})}{\sin\theta^{(k)}}$ is an eigenvector of D^TD_e with eigenvalue $\lambda^{(k)} = 2 - 2\cos\theta^{(k)}$, where $\theta^{(k)} = \frac{(2k-1)\pi}{2l+1}$.

Proof: We need to show that for all $1 \le k \le l$, for all $1 \le j \le l$, $(D^TD)_e(j,:) \cdot v^{(k)} = \lambda^{(k)} v_j^{(k)}$. (Where $(D^TD)_e(j,:)$ refers to the j^{th} row of $(D^TD)_e$.) This condition is satisfied exactly if the following three conditions are satisfied

$$v_1 - v_2 = (2 - 2\cos\theta^{(k)})v_1 \tag{1}$$

$$-v_{j-1} + 2v_j - v_{j+1} = (2 - 2\cos\theta^{(k)})v_j \qquad \text{for } 2 \le j \le l-1 \tag{2}$$

$$-v_{l-1} + 2v_l = (2 - 2\cos\theta^{(k)})v_l \tag{3}$$

(2): We will first prove the more general case of (2) in which $\theta^{(k)}$ is any angle such that $\sin\theta \ne 0$, and j is any integer. Thus, for the our proof of (2), (and our proof of (1)), we will omit the superscript (k) since the proof holds for any angle with $\sin\theta \ne 0$. The proof of (3) relies on the definition of $\theta^{(k)}$, so we will reintroduce the superscript then.

Let j be an integer and θ an angle such that $\sin\theta \neq 0$. We need that

$$\begin{aligned}
&-v_{j-1} + 2v_j - v_{j+1} = (2 - 2\cos\theta)v_j = 2v_j - 2\cos\theta v_j \\
&\Leftrightarrow v_{j-1} + v_{j+1} = 2\cos\theta v_j \\
&\Leftrightarrow \frac{\sin((j-1)\theta) - \sin((j-2)\theta) + \sin((j+1)\theta) - \sin(j\theta)}{\sin\theta} = \frac{2\cos\theta(\sin(j\theta) - \sin((j-1)\theta))}{\sin\theta} \\
&\Leftrightarrow \sin((j-1)\theta) - \sin((j-2)\theta) + \sin((j+1)\theta) - \sin(j\theta) = 2\cos\theta(\sin(j\theta) - \sin((j-1)\theta)).
\end{aligned}$$

Using the trigonometric identities,

$$\begin{aligned}
\sin(\alpha+\beta) &= \sin\alpha\cos\beta + \cos\alpha\sin\beta & \cos(\alpha+\beta) &= \cos\alpha\cos\beta - \sin\alpha\sin\beta \\
\sin(2\alpha) &= 2\sin\alpha\cos\alpha & \cos(2\alpha) &= 1 - 2\sin^2\alpha,
\end{aligned}$$

this becomes

$$\begin{aligned}
&\sin(j\theta)\cos\theta - \sin(j\theta) + \sin^2\theta\sin(j\theta) + \cos(j\theta)\sin\theta\cos\theta \\
&= \sin(j\theta)\cos\theta - \sin(j\theta)\cos^2\theta + \cos(j\theta)\sin\theta\cos\theta \\
&\Leftrightarrow (\sin^2\theta + \cos^2\theta)\sin(j\theta) = \sin(j\theta) \\
&\Leftrightarrow \sin(j\theta) = \sin(j\theta) \\
&\Leftrightarrow 1 = 1.
\end{aligned}$$

(1): As with (2), this holds for any angle, θ, such that $\sin\theta \neq 0$. Thus we will omit the superscripts.

$$\begin{aligned}
v_1 &= \frac{\sin\theta - \sin 0}{\sin\theta} = 1 \\
v_2 &= \frac{\sin(2\theta) - \sin(\theta)}{\sin\theta} = \frac{2\sin\theta\cos\theta - \sin(\theta)}{\sin\theta} = 2\cos\theta - 1. \\
\text{Thus, } v_1 - v_2 &= 1 - (2\cos\theta - 1)) = 2 - 2\cos\theta = (2 - 2\cos\theta)v_1.
\end{aligned}$$

(3): Here, the particular definition of $\theta^{(k)}$ is important, so we will include the superscripts. First, however, we prove two quick trigonometric identities.

$$\begin{aligned}
&\sin(\pi - \alpha) = \sin\pi\cos(\alpha) - \cos\pi\sin\alpha = 0 - (-1)\sin\alpha = \sin\alpha \\
&\sin((2k-1)\pi - \alpha) = \sin((2k-1)\pi - \alpha + 2k\pi) = \sin(\pi - \alpha) = \sin\alpha \ \forall k \in \mathbb{Z}.
\end{aligned}$$

We need to show that $-v_{l-1}^{(k)} + 2v_l^{(k)} = \lambda^{(k)}v_l^{(k)}$. If we extend the definition of $v^{(k)}$ to include $v_{l+1}^{(k)}$, then we know from our proof of (2) that

$$-v_{l-1}^{(k)} + 2v_l^{(k)} - v_{l+1}^{(k)} = \lambda^{(k)}v_l^{(k)}.$$

Thus, it suffices to prove that $v_{l+1}^{(k)} = 0$.

$$\begin{aligned}
&v_{l+1}^{(k)} = \frac{\sin((l+1)\theta^{(k)}) - \sin(l\theta^{(k)})}{\sin\theta^{(k)}}) = 0 \iff \sin((l+1))\theta^{(k)} = \sin(l\theta^{(k)}) \\
&(l+1)\theta^{(k)} + l\theta^{(k)} = (2l+1)\theta^{(k)} = (2l+1)\frac{(2k-1)\pi}{2l+1} = (2k-1)\pi \\
&\Rightarrow (l+1)\theta^{(k)} = (2k-1)\pi - l\theta^{(k)} \Rightarrow \sin((l+1)\theta^{(k)}) = \sin(l\theta^{(k)}) \Rightarrow v_{l+1}^{(k)} = 0 \\
&\Rightarrow -v_{l-1}^{(k)} + 2v_l^{(k)} = \lambda^{(k)}v_l^{(k)}.
\end{aligned}$$

Therefore, each $v^{(k)}$ is an eigenvector of $D^T D_e$ with eigenvalue $\lambda^{(k)}$.

We now have everything we need to get a Singular Value Decomposition of D. As we noted earlier, $\tilde{\Phi}_1^{(k)} = v^{(k)} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\tilde{\Phi}_2^{(k)} = \tilde{v}^{(k)} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ are eigenvectors of $D^T D$ for $1 \leq k \leq l$. Therefore, $\Phi_1^{(k)} = \frac{\tilde{\Phi}_1^{(k)}}{||\tilde{\Phi}_1^{(k)}||}$ and $\Phi_2^{(k)} = \frac{\tilde{\Phi}_2^{(k)}}{||\tilde{\Phi}_2^{(k)}||}$ are normalized eigenvectors of $D^T D$. Thus, a singular value decomposition of D is given by $D = U \Sigma V^T$, where $V = (\Phi_1^{(1)} \Phi_2^{(1)} \Phi_1^{(2)} \dots \Phi_2^{(l)})$,

$$\Sigma = \begin{bmatrix} \sigma_1 & & & & & & \\ & \sigma_1 & & & & & \\ & & \sigma_2 & & & & \\ & & & \sigma_2 & & & \\ & & & & \ddots & & \\ & & & & & \sigma_l & \\ & & & & & & \sigma_l \end{bmatrix},$$

and $U = D V \Sigma^{-1}$.

We now try to apply this result to the $2ln^2$ by $2ln^2$ Domain Wall matrix, $R + D \otimes I_{n^2}$, with $R = I \otimes M_{\rho-1}$. As mentioned earlier, we know that a Singular Value Decomposition of $D \otimes I_{n^2}$ is given by $\tilde{U} \Sigma \tilde{V}^T$, with $\tilde{U} = U \otimes I_{n^2}$, $\tilde{\Sigma} = \Sigma \otimes I_{n^2}$, and $\tilde{V} = V \otimes I_{n^2}$. We will be able to decouple the Domain Wall system if $\tilde{U}^T R \tilde{V}$ is block diagonal.

For this result to hold in general, it must hold in the particular case that $n = 1$, $l = 4$, $\rho = 1$. In that case, S_A and S_B become numbers, a and b, and $M_{\rho-1}$ becomes a 2 by 2 matrix,

$$\begin{bmatrix} a & b \\ -\bar{b} & a \end{bmatrix}.$$

So R becomes

$$\left[\begin{array}{cc|cc|cc|cc} a & b & 0 & 0 & 0 & 0 & 0 & 0 \\ -\bar{b} & a & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & a & b & 0 & 0 & 0 & 0 \\ 0 & 0 & -\bar{b} & a & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & a & b & 0 & 0 \\ 0 & 0 & 0 & 0 & -\bar{b} & a & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & a & b \\ 0 & 0 & 0 & 0 & 0 & 0 & -\bar{b} & a \end{array}\right].$$

In the case that $\alpha = \beta = 0$, $a = -2$ and $b = -.5 + -.5\imath$, carrying out the matrix multiplication, we get

$$\tilde{U}^T R \tilde{V} \approx \begin{bmatrix} -1.91 & -.5+.5i & -.56 & 0 & -.10 & 0 & -.20 & 0 \\ .5+.5i & -1.91 & 0 & -.56 & 0 & -.09 & 0 & -.1954 \\ .5627 & 0 & -1.65 & .5-.5i & -.93 & 0 & -.2994 & 0 \\ 0 & .56 & -.5-.5i & -1.65 & 0 & -.93 & 0 & -.30 \\ -.09 & 0 & .93 & 0 & -1.33 & -.5+.5i & -1.15 & 0 \\ 0 & -.09 & 0 & .93 & .5+.5i & -1.33 & 0 & -1.1615 \\ .20 & 0 & -.30 & 0 & 1.16 & 0 & -1.59 & .5-.5i \\ 0 & .20 & 0 & -.30 & 0 & 1.16 & -.5-.5i & -1.59 \end{bmatrix}.$$

Unfortunately, this matrix is not diagonal, nor is it sparse. Therefore, this particular Singular Value Decomposition does not allow us to decouple the domain wall system. However, the Singular

Value Decomposition of this matrix is not uniquely defined since each singular value has multiplicity 2. Therefore, it is still possible that another Singular Value Decomposition could allow us to decouple the system.

Let us consider the possible Singular Value Decompositions of D into $U\Sigma V^T$. Σ is uniquely defined since it is the diagonal matrix, with the singular values of D (in descending order) as its non-zero entries. For a given V, U is required to be $D\Sigma^{-1}V$, thus our choice of V uniquely determines the entire Singular Value Decomposition. If $\sigma = \Sigma_{k,k}$, then the k^{th} column of V, $v^{(k)}$, is an eigenvector of D^TD, with eigenvalue σ^2. For this problem, each eigenvector of D^TD has a two dimension eigenspace. Thus, for $1 \leq k \leq l$, $v^{(2k-1)}$ and $v^{(2k)}$ must form an orthonormal basis for the eigenspace associated with $\sigma_{k/2}$.

Earlier, we determined that $\Phi_1^{(k)}$ and $\Phi_2^{(k)}$ formed an orthonormal basis of the eigenspace associated with each singular value, σ_k. We may generalize this by noting for any angle, $\omega^{(k)}$, $\Psi_1^{(k)} = \cos\omega^{(k)}\Phi_1^{(k)} + \sin\omega^{(k)}\Phi_2^{(k)}$ and $\Psi_2^{(k)} = -\sin\omega^{(k)}\Phi_1^{(k)} + \cos\omega^{(k)}\Phi_2^{(k)}$ also form an orthonormal basis for the eigenspace associated with σ_k. We will try to find an SVD which allows us to decouple the domain wall system for the very small case in which $L = 2$ and $n = 1$. It is our hope that we will be able to find a relation between the necessary ω's that will allow us to generalize this to greater values of L and n. Alternatively, if we cannot decouple this very small case of the domain wall system, that will suggest that SVD is not a good approach for this problem.

When $L = 2$, $n = 1$,

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad D^TD = \begin{bmatrix} 2 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 2 \end{bmatrix}.$$

And Σ is given by,

$$\Sigma = \begin{bmatrix} \varphi & 0 & 0 & 0 \\ 0 & \varphi & 0 & 0 \\ 0 & 0 & \frac{1}{\varphi} & 0 \\ 0 & 0 & 0 & \frac{1}{\varphi} \end{bmatrix},$$

where φ is the golden mean, $\frac{1+\sqrt{5}}{2} \approx 1.6180$. We let V be the eigenvector matrix of D^TD, without normalized columns resulting from our rule that $v_j^{(k)} = \frac{\sin(j\theta^{(k)}) - \sin((j-1)\theta^{(k)})}{\sin\theta^{(k)}}$. We do not normalize the columns because it is easier to do the computations with the non-normalized columns, and the "unnormalized SVD" will not distort which entries are non-zero. That is, $U\Sigma V^T$ will not be the same as D, but it will have the same shape. Plugging in the values of V and computing $U = DV\Sigma^{-1}$ gives us

$$V = \begin{bmatrix} 0 & -\varphi & 0 & \frac{1}{\varphi} \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ -\varphi & 0 & \frac{1}{\varphi} & 0 \end{bmatrix} \qquad U = \begin{bmatrix} 0 & -1 & 0 & 1 \\ \varphi & 0 & \frac{1}{\varphi} & 0 \\ 0 & \varphi & 0 & \frac{1}{\varphi} \\ -1 & 0 & 1 & 0 \end{bmatrix}.$$

We now construct general SVD of D, $D = T\Sigma M^T$. For any angles, ω_1 and ω_2, we define the columns of T by

$$T^{(1)} = c_1v^{(1)} + s_1v^{(2)}, \qquad T^{(2)} = -s_1v^{(1)} + c_1v^{(2)},$$
$$T^{(3)} = c_2v^{(3)} + s_2v^{(4)}, \qquad T^{(4)} = -s_2v^{(3)} + 2_1v^{(4)},$$

where, $T^{(i)}$ is the i^{th} column of T, $V^{(i)}$ is the i^{th} column of V, $c_i = \cos(\omega_i)$, and $s_i = \sin(\omega_i)$. Plugging in and computing $M = DT\Sigma^{-1}$ gives,

$$T = \begin{bmatrix} -s_1\varphi & -c_1\varphi & s_2\frac{1}{\varphi} & c_2\frac{1}{\varphi} \\ c_1 & -s_1 & c_2 & -s_2 \\ s_1 & c_1 & s_2 & c_2 \\ -c_1\varphi & s_1\varphi & c_2\frac{1}{\varphi} & -s_2\frac{1}{\varphi} \end{bmatrix} \qquad M = \begin{bmatrix} -s_1 & -c_1 & s_2 & c_2 \\ c_1\varphi & -s_1\varphi & c_2\frac{1}{\varphi} & -s_2\frac{1}{\varphi} \\ s_1\varphi & c_1\varphi & s_2\frac{1}{\varphi} & c_2\frac{1}{\varphi} \\ -c_1 & s_1\varphi & c_2 & -s_2 \end{bmatrix}.$$

It is now our hope that $M^T RT$ is diagonal, where $R = I_l \otimes M$ is the block diagonal component of the domain wall matrix. In the case that $n = 1$, $l = 2$, R reduces to the identity, I_4. Therefore, we need to find values of ω_1 and ω_2 such that $M^T T$ is diagonal. Setting $A = M^T T$, direct computation shows that

$$\begin{aligned} &A_{1,1} = A_{2,2} = 2\varphi, \qquad A_{3,3} = A_{4,4} = \frac{2}{\varphi} \\ &A_{1,2} = A_{2,1} = A_{3,4} = A_{4,3} = 0 \\ &A_{1,3} = A_{3,1} = -A_{2,4} = -A_{4,2} = \cos(\omega_1 - \omega_2)(\frac{1}{\varphi} - \varphi) \\ &A_{1_4} = A_{3,2} = -A_{4,1} = -A_{2,3} = \sin(\omega_1 - \omega_2)(\varphi - \frac{1}{\varphi}). \end{aligned}$$

Thus, in order for A to be diagonal, we need $\sin(\omega_1 - \omega_2) = \cos(\omega_1 - \omega_2) = 0$. This is an obvious contradiction. Therefore, there is no choice of ω_1 and ω_2 that will allow us to decouple the domain wall system for $n = 1$, $l = 2$. Given that SVD cannot decouple this very small version of the system, we conclude that we must use another approach to solve the domain wall system.

0.4 MultiGrid Methods

We now shift gears and try to solve the domain wall System using BICGSTAB with a multigrid preconditioner. For physically realistic values of ρ, the domain wall is indefinite. (It has eigenvalues with both positive and negative real parts.) BICGSTAB with a Multigrid Preconditioner has been shown to be effective for a wide range Helmholtz type problems which have been resistant to other iterative approaches. We will first explore Multigrid Methods and then briefly discuss BICGSTAB and how Multigrid Methods may be incorporated as preconditioner. We will then discuss a method by which we can solve the entire domain wall system if we can solve the A block which appears on the diagonal.

Multigrid methods are a family of iterative methods, initially developed to numerically solve discretized PDEs, that use a number of successive levels of discretization [1]. We will first discuss two basic iterative methods. Then, we shall discuss why these techniques tend to stall and how they can be made more effective by incorporating them into a Multigrid scheme.

An iterative method may be loosely thought of as any technique in which successive guesses are made as to the solution of a system with the goal that each guess is closer to the solution than the previous one. We will denote the linear system we are trying to solve as $A\mathbf{u} = \mathbf{f}$, and use $\mathbf{v}$ to refer to our guess of the exact solution, $\mathbf{u}$. We define the error, $\mathbf{e}$, by the rule that $\mathbf{e} = \mathbf{u} - \mathbf{v}$. If $||\cdot||$ is a norm function, then $||\mathbf{e}||$ represents the size of the error. It is our goal for $||\mathbf{e}||$ to decrease after each iteration since $||\mathbf{e}|| = 0$ if and only if $\mathbf{e} = \overrightarrow{\mathbf{0}}$, the vector of all zeros.

We now discuss the Jacobi Method, one of the simplest iterative techniques. To illustrate this technique, we shall use our discretization of the two-dimensional Poisson Equation, $-\nabla^2 u(x, y) =$

$f(x,y)$, as an example. Recall that this system was approximated by $A\mathbf{u} = \mathbf{f}$, where $\mathbf{u}$ and $\mathbf{f}$ are vector representations of $u(x,y)$ and $f(x,y)$, and A is the matrix representing $-\nabla^2$, given by

$$A = \frac{1}{h^2}\left[\begin{array}{cccc|cccc|cccc|cccc}
4 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
-1 & 0 & -1 & 4 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
\hline
-1 & 0 & 0 & 0 & 4 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & -1 & 0 & -1 & 4 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 4 & -1 & 0 & -1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & -1 & 4 & 0 & 0 & 0 & -1 \\
\hline
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 4 & -1 & 0 & -1 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & -1 & 4
\end{array}\right].$$

In order for the equality, $A\mathbf{u} = \mathbf{f}$, to hold. We need, except for on rows representing boundary points, that

$$\frac{1}{h^2}(-\mathbf{u}_{j-n} - \mathbf{u}_{j-1} + 4\mathbf{u}_j - \mathbf{u}_{j+1} - \mathbf{u}_{j+n}) = \mathbf{f}_j$$
$$\Leftrightarrow \mathbf{u}_j = \frac{1}{4}(\mathbf{u}_{j-n} + \mathbf{u}_{j-1} + \mathbf{u}_{j+1} + \mathbf{u}_{j+n} + h^2\mathbf{f}_j).$$

For each $\mathbf{v}_j$, we shall use our current guesses of $\mathbf{v}_{j\pm 1}$ and $\mathbf{v}_{j\pm n}$, to update $\mathbf{v}$. That is, if $\mathbf{v}^{(0)}$ is our current guess of $\mathbf{u}$, we define our updated guess, $\mathbf{v}^{(1)}$ by the rule,

$$\mathbf{v}_j^{(1)} = \frac{1}{4}\left(\mathbf{v}_{j-n}^{(0)} + \mathbf{v}_{j-1}^{(0)} + \mathbf{v}_{j+1}^{(0)} + \mathbf{v}_{j+n}^{(0)} + h^2\mathbf{f}_j\right).$$

We use analogous rules for the rows corresponding to boundary points. To keep notation simple, for each iteration, we shall refer to the guess at the beginning of the iteration as $\mathbf{v}^{(0)}$, and we shall refer to the guess at the end of that iteration as $\mathbf{v}^{(1)}$. It is useful to consider the Jacobi Method represented by matrices. Note that for any matrix, A, we may write $A = D - (L + U)$, where D is the matrix containing the diagonal entries of A, and $-L$ and $-U$ are matrices containing the strictly lower triangular and strictly upper triangular entries of A, respectively. This gives us

$$A\mathbf{u} = \mathbf{f} \Leftrightarrow D\mathbf{u} = (L + U)\mathbf{u} + \mathbf{f}.$$

As before, we want to update each v_j using the neighboring points. Thus, we associate the $\mathbf{u}$ on the left-hand side of the above equation with $\mathbf{v}^{(1)}$, and we associate the $\mathbf{u}$ on the right hand side with $\mathbf{v}^{(0)}$, we get

$$D\mathbf{v}^{(1)} = (L + U)\mathbf{v}^{(0)} + \mathbf{f} \Leftrightarrow \mathbf{v}^{(1)} = D^{-1}(L + U)\mathbf{v}^{(0)} + D^{-1}\mathbf{f}.$$

Letting $R_J = D^{-1}(L + U)$, this equation becomes

$$\mathbf{v}^{(1)} = R_J\mathbf{v}^{(0)} + D^{-1}\mathbf{f}.$$

The Jacobi method can be generalized into the Weighted Jacobi Method. For this method, we create the intermediate value $\mathbf{v}^*$ defined by

$$\begin{aligned}\mathbf{v}_j^* &= \frac{1}{4}(\mathbf{v}_{j-n}^{(0)} + \mathbf{v}_{j-1}^{(0)} + \mathbf{v}_{j+1}^{(0)} + \mathbf{v}_{j+n}^{(0)} + h^2\mathbf{f}_j),\\ \text{and set } \mathbf{v}^{(1)} &= \omega\mathbf{v}^* + (1-\omega)\mathbf{v}^{(0)},\end{aligned}$$

where ω is some number in $(0, 1]$. With matrix representation, this becomes

$$\begin{aligned}\mathbf{v}^{(1)} &= R_\omega\mathbf{v}^{(0)} + \omega D^{-1}\mathbf{f},\\ \text{where } R_\omega &= \omega R_J + (1-\omega)I.\end{aligned}$$

Both the Jacobi method and the weighted Jacobi method are examples of stationary linear iterations. In this context, the term stationary refers to the fact that the rule for updating our guess does not change from iteration to iteration. Linear refers to the fact that the rule for updating our guess is a linear operation. We will now examine the general form of a stationary linear iterative method.

By construction, $\mathbf{u} = \mathbf{e} + \mathbf{v}$. Using the fact that $A\mathbf{e} = \mathbf{r}$, we can write $\mathbf{u} = A^{-1}\mathbf{r} + \mathbf{v}$. For problems to which we apply iterative techniques, we do not generally know A^{-1} since otherwise we would solve the problem directly. However, if we are to able to construct a matrix, B, that approximates A^{-1}, then it is natural to use the iterative technique,

$$\begin{aligned}&\mathbf{v}^{(1)} = B\mathbf{r} + \mathbf{v}^{(0)} = B(\mathbf{f} - A\mathbf{v}^{(0)}) + I\mathbf{v}^{(0)} = (I - BA)\mathbf{v}^{(0)} + B\mathbf{f} = R\mathbf{v}^{(0)} + \mathbf{g},\\ &\text{where } R = I - BA \text{ and } \mathbf{g} = B\mathbf{f}.\end{aligned}$$

For any stationary linear iteration, we know that $\mathbf{v}^{(1)} = R\mathbf{v}^{(0)} + \mathbf{g}$. Intuitively, once an iteration returns the exact solution, all subsequent iterations must also return the exact solution, that is, if $\mathbf{v}^{(0)} = \mathbf{u}$ then $\mathbf{v}^{(1)} = \mathbf{u}$. Therefore, $\mathbf{u} = R\mathbf{u} + \mathbf{g}$. Subtracting the general equation from this shows that

$$\begin{aligned}&\mathbf{u} - \mathbf{v}^{(1)} = R\mathbf{u} + g - (R\mathbf{v}^{(0)} + g) = R(\mathbf{u} - \mathbf{v}^{(0)})\\ &\Rightarrow \mathbf{e}^{(1)} = R\mathbf{e}^{(0)}.\end{aligned}$$

It follows by induction that

$$\mathbf{e}^{(m)} = R^m\mathbf{e}^{(0)} \Rightarrow ||\mathbf{e}^{(m)}|| \leq ||R||^m||\mathbf{e}^{(0)}||.$$

Where $||R||$ refers to the 2 norm of R, which is given by $||R||_2 = \sqrt{\rho(R^TR)}$, where $\rho(R^TR)$ is the spectral radius of R^TR (the largest eigenvalue of R^TR in absolute value). Thus, if $||R|| < 1$, then

$$\lim_{n\to\infty}\mathbf{e}^{(n)} = \overrightarrow{\mathbf{0}}.$$

To numerically examine the effectiveness of these methods, we shall perform numerical experiments where $\mathbf{f} = \overrightarrow{\mathbf{0}}$, $\mathbf{v}$ is given by a discrete Fourier mode (or a combination of discrete fourier modes), and A is the n by n matrix representing the one dimensional Poisson operator. It is advantageous to examine the homogeneous case that $\mathbf{f} = \overrightarrow{\mathbf{0}}$, since the exact solution, $\mathbf{u} = \overrightarrow{\mathbf{0}}$, is known. Therefore, $\mathbf{e} = -\mathbf{v}$ and $||\mathbf{e}|| = ||\mathbf{v}||$. The discrete Fourier modes are vectors of the form

$$v_j^{(k)} = \sin\left(\frac{jk\pi}{n}\right),$$

which are obtained by discretizing the continuous function $\sin(\pi kx)$. k is a parameter representing frequency and is equal to the number of half-periods of $\sin(\pi kx)$, which are represented in $\mathbf{v}^{(k)}$. Discrete Fourier modes are attractive guesses since periodic functions can be represented as the sum of continuous Fourier modes, which suggests that our solution can be expressed as the sum of discrete Fourier modes. We refer to the modes with larger values of k as oscillatory modes, and we call the modes with smaller values of k as smooth modes.

Figure 2: Relaxation of Fourier Modes

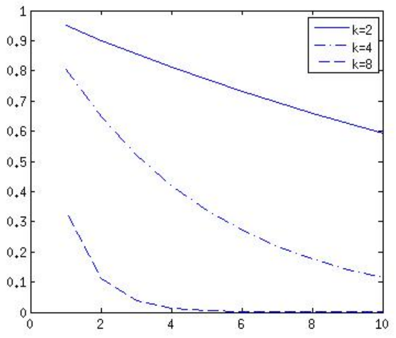


Figure 2 graphs the relative residual after successive sweeps of Weighted Jacobi Relaxation for Fourier modes with different wavenumbers. Note that for $k = 2$, Weighted Jacobi Relaxation is not very effective, for $k = 4$ it is pretty effective, and for $k = 8$ it is very effective. This example illustrates a trend, which is true in general, that Weighted Jacobi Relaxation is effective if the error is oscillatory and ineffective if the error is smooth. Therefore, if the error is expressed as the sum of discrete Fourier modes, Weighted Jacobi Relaxation will rapidly eliminate the portion of the error that is represented by the oscillatory modes. After a few iterations, the error will be almost entirely represented by smooth Fourier modes. This is known as the smoothing effect. Once the error is sufficiently smooth, the reduction of error norms by Jacobi iteration will become very small. At this point, Jacobi Relaxation is said to have stalled. Similar problems occur for other stationary linear methods. Multigrid methods were developed, in part, to resolve this problem.

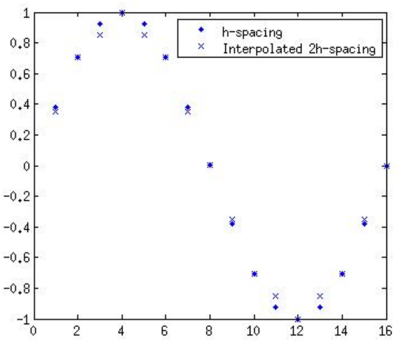
One way to improve the performance of an iterative method is to make a good initial guess. For a system, $A^{(h)}\mathbf{u}^{(h)} = \mathbf{f}^{(h)}$, arising from PDEs, the natural way to do this is to solve the analogous system for a coarser level of discretization, $A^{(2h)}\mathbf{u}^{(2h)} = \mathbf{f}^{(2h)}$. The superscripts refer to the grid spacing of the discretization, that is, a superscript of (h) implies that node points are $h = \frac{1}{n}$ apart. Thus, $\mathbf{u}^{(2h)}$ and $\mathbf{f}^{(2h)}$ are the $\frac{n^2}{4}$ by 1 vectors obtained by discretizing $u(x, y)$ and $f(x, y)$ with grid spacing $2h$. For now, let $A^{(2h)}$ be the matrix which arises from applying finite differences, with grid spacing $2h$, to same differential operator from which $A^{(h)}$ arises. Intuitively, if h is small, $\mathbf{u}^{(2h)}$ should give us a good representation of $\mathbf{u}^{(h)}$ since $u(x, y)$ is a continuous function.

This raises some natural questions. Given $\mathbf{u}^{(2h)}$, how do we define $\mathbf{u}^{(h)}$? Conversely, given $\mathbf{u}^{(h)}$, how do we define $\mathbf{u}^{(2h)}$? These processes are referred to as prolongation and restriction, respectively. In this context, the grid with $2h$ spacing is called the coarse grid, and the grid with h spacing shall be called the fine grid. Later, when we have multiple grids, we shall say that a grid with h spacing is finer than a grid with $2h$ spacing, which is in turn finer than a grid with $4h$ spacing, etc. For now, we consider vectors which arise from one-dimensional functions.

The simplest prolongation procedure is called linear interpolation, defined by

$$\mathbf{u}_{2j}^{(h)} = \mathbf{u}_j^{(2h)} \qquad \mathbf{u}_{2j+1}^{(h)} = \frac{1}{2}\left(\mathbf{u}_j^{(2h)} + \mathbf{u}_{j+1}^{(2h)}\right).$$

For example, interpolation the vector, $\mathbf{v}^{(2h)} = (1,2,1,0)^T$ would give $\mathbf{v}^{(h)} = (.5,1,1.5,2,1.5,1,0.5,0)^T$. The figure below compares $\mathbf{v}^{(h)}$, the vector we get by discretizing $\sin(x)$ with grid spacing $h = \frac{1}{16}$, to $P\mathbf{v}^{(2h)}$, the vector we get by discretizing $\sin(x)$ with grid spacing $2h$ and then interpolating.



The most natural process for restriction is injection, which is defined by

$$\mathbf{u}_j^{(2h)} = \mathbf{u}_{2j}^{(h)}.$$

However, the obvious drawback of injection is that only half the points of $\mathbf{u}^{(h)}$ are used to define $\mathbf{u}^{(2h)}$. To use all points of $\mathbf{u}^{(h)}$ equally, we define full-weighting by

$$\mathbf{u}_j^{(2h)} = \frac{1}{4}\left(\mathbf{u}_{2j-1}^{(h)} + 2\mathbf{u}_{2j}^{(h)} + \mathbf{u}_{2j+1}^{(h)}\right).$$

It is useful to think of these operations as a matrix. For linear interpolation, we have the rule that

$$\mathbf{u}_j^{(h)} = \left\{ \begin{array}{cc} \mathbf{u}_{j/2}^{(2h)} & \text{if j is even} \\ \frac{1}{2}\left(\mathbf{u}_{(j-1)/2}^{(2h)} + \mathbf{u}_{(j+1)/2}^{(2h)}\right) & \text{if j is odd} \end{array} \right\}.$$

Therefore, interpolation has matrix representation

$$P = \frac{1}{2}\begin{pmatrix} 1 & & & & & & 1 \\ 2 & & & & & & \\ 1 & 1 & & & & & \\ & 2 & & & & & \\ & 1 & 1 & & & & \\ & & 2 & & & & \\ & & & \ddots & \ddots & & \\ & & & & & 1 & 1 \\ & & & & & & 2 \end{pmatrix}.$$

By similar analysis, we may represent 1-dimensional injection and 1-dimensional full weighting. We are, however, actually interested in interpolating and restricting the discretization of a two-dimensional function. While our equation requires our discretization to be a vector, the definitions of our operators are much more intuitive when considering our discretization as a square matrix.

If i and j are even, then $\mathbf{u}_{i,j}^{(h)}$ corresponds to the same value of $u(x,y)$ as $\mathbf{u}_{i/2,j/2}^{(2h)}$ so we set $\mathbf{u}_{i,j}^{(h)} = \mathbf{u}_{i/2,j/2}^{(2h)}$. If i is even and j is odd, then $\mathbf{u}_{i,j}^{(h)}$ corresponds to the value of $u(x,y)$ at the point half way between $\mathbf{u}^{(2h)}_{i/2,(j-1)/2}$ and $\mathbf{u}^{(2h)}_{i/2,(j+1)/2}$, so we set $\mathbf{u}^{(h)}_{i,j} = \frac{1}{2}\left(\mathbf{u}^{(2h)}_{i/2,(j-1)/2} + \mathbf{u}^{(2h)}_{i/2,(j+1)/2}\right)$. Similarly, if j is even and i is odd, we set $\mathbf{u}^{(h)}_{i,j} = \frac{1}{2}\left(\mathbf{u}^{(2h)}_{(i-1)/2,j/2} + \mathbf{u}^{(2h)}_{(i+1)/2,j/2}\right)$. If both j and i are odd, then $\mathbf{u}^{(h)}_{i,j}$ is evenly spaced between 4 coarse-grid points, so we set $\mathbf{u}^{(h)}_{i,j} = \frac{1}{4}\left(\mathbf{u}^{(2h)}_{(i-1)/2,(j-1)/2} + \mathbf{u}^{(2h)}_{(i+1)/2,(j-1)/2} + \mathbf{u}^{(2h)}_{(i-1)/2,(j+1)/2}\right.$

Representing this operation as a matrix is more complicated than in the one-dimensional case. Here we must consider $\mathbf{u}^{(h)}$ as an n^2 by 1 vector, rather than an n by n matrix. Each entry in the vector, $\mathbf{u}^{(h)}_k$, corresponds to a point, $\mathbf{u}^{(h)}_{i,j}$, in the matrix representation of $u(x, y)$, where $k = i + n(j-1)$. In defining the k^{th} row, we must consider the parity of both i and j in terms of k. For simplicity, we will only use grids with even values of n. It is clear that the parity of i is equal to the parity of k, since $k - i = (j-1)n \equiv 0 \mod 2$. A little bit of arithmetic shows that $j = \lceil \frac{k}{n} \rceil$, which allows us to determine the parity of the corresponding j to a given k.

When constructing P, the n^2 by $\frac{n^2}{4}$ interpolation matrix that maps from a coarse grid with $\frac{n}{2} \times \frac{n}{2}$ points to a fine grid with $n \times n$ points, it is easiest to first construct four $\frac{n^2}{4}$ by $\frac{n^2}{4}$ matrices, A, B, C, and D, with A corresponding to odd values of i and j, B corresponding to even values of i and odd values of j, C corresponding odd values of i and even values of j, and D corresponding to even values of both i and j. After constructing these four matrices, we can construct P by taking each row of P from either A, B, C, or D. In particular, for rows of P corresponding to odd values of both i and j, we use a row of A, for rows corresponding to an even value of i and an odd value of j, we use a row of B, for rows corresponding to an odd value of i and an even value of j, we use a row of C, and for rows corresponding to even values of both i and j, we use a row of D. Thus, the first n rows of P will alternate between rows of A and rows of B. The next n rows will alternate between rows of C and rows of D and so on, with each set of n rows corresponding to a column of the $n \times n$ grid. This gives

$$P = \begin{pmatrix} A(1,:) \\ B(1,:) \\ A(2,:) \\ B(2,:) \\ \vdots \\ A(\frac{n}{4},:) \\ B(\frac{n}{4},:) \\ C(1,:) \\ D(1,:) \\ \vdots \\ C(\frac{n}{4},:) \\ D(\frac{n}{4},:) \\ A(\frac{n}{4}+1,:) \\ B(\frac{n}{4}+1,:) \\ \vdots \\ \vdots \\ C(\frac{n^2}{4},:) \\ D(\frac{n^2}{4},:) \end{pmatrix}.$$

We must now consider how to construct these 4 matrices. D is the simplest to construct. Consider the l^{th} row of D. This will determine the l^{th} point in $\mathbf{u}^{(h)}$ that corresponds to even values of both i and j. A point in $\mathbf{u}^{(h)}$ has even values of both i and j if and only if it corresponds to a point on the coarse grid. Thus, we want the l^{th} such point in $\mathbf{u}^{(h)}$ to be set equal to the l^{th} point on the coarse grid. Thus D is the multiplicative identity matrix $I_{n^2/4}$.

The l^{th} row of B determines the value of the l^{th} fine-grid point with an even value of i and an odd value of j. This fine-grid point is the midpoint of the horizontal line segment between two fine-grid points. Careful inspection shows that these are the l^{th} and $(l-n)^{th}$ coarse-grid points. Thus we set $B_{l,l} = B_{l,l-\frac{n}{2}} = \frac{1}{2}$, $B_{l,m} = 0$, otherwise. To take into account the periodic boundary condition, we also set $B_{l,\frac{n^2}{4}-\frac{n}{2}+l} = \frac{1}{2}$, if $l \leq \frac{n}{2}$. Similarly, for most rows of C, we set $C_{l,l} = C_{l,l-1} = \frac{1}{2}$, $C_{l,m} = 0$, otherwise. To account for the periodicity, if $l \equiv 1 \mod \frac{n}{2}$, we set $C_{l,l-1} = 0$ and $C_{l,l+n-1} = \frac{1}{2}$. For $n = 4$, this becomes

$$B = \frac{1}{2}\begin{bmatrix} I_4 & & & I_4 \\ I_4 & I_4 & & \\ & I_4 & I_4 & \\ & & I_4 & I_4 \end{bmatrix} \qquad C = \begin{bmatrix} M & & & \\ & M & & \\ & & M & \\ & & & M \end{bmatrix},$$

where

$$M = \frac{1}{2}\begin{bmatrix} 1 & & & 1 \\ 1 & 1 & & \\ & 1 & 1 & \\ & & 1 & 1 \end{bmatrix}.$$

Similar analysis shows that

$$A = \frac{1}{2}\begin{bmatrix} M & & & M \\ M & M & & \\ & M & M & \\ & & M & M \end{bmatrix},$$

with M defined as above.

It is worth noting that using the Kronecker product we can write this concisely as

$$A = M \otimes M \qquad B = M \otimes I \qquad C = I \otimes M \qquad D = I \otimes I.$$

Also note the one-dimensional linear interpolation matrix, P_1, permutes to to $P_1^* = \begin{pmatrix} I \\ M \end{pmatrix}$. It follows by direct computation that $P_1^* \otimes P_1^*$ is given by

$$\begin{bmatrix} P_1^* & & & \\ & P_1^* & & \\ & & \ddots & \\ & & & P_1^* \\ \frac{1}{2}P_1^* & & & \frac{1}{2}P_1^* \\ \frac{1}{2}P_1^* & \frac{1}{2}P_1^* & & \\ & \ddots & \ddots & \\ & & \frac{1}{2}P_1^* & \frac{1}{2}P_1^* \end{bmatrix} = \begin{bmatrix} I & & & & \\ M & & & & \\ & I & & & \\ & M & & & \\ & & \ddots & \ddots & \\ & & & & I \\ & & & & M \\ \frac{1}{2}I & & & & \frac{1}{2}I \\ \frac{1}{2}M & & & & \frac{1}{2}M \\ \frac{1}{2}I & \frac{1}{2}I & & & \\ \frac{1}{2}M & \frac{1}{2}M & & & \\ & & \ddots & \ddots & \\ & & & \frac{1}{2}I & \frac{1}{2}I \\ & & & \frac{1}{2}M & \frac{1}{2}M \end{bmatrix}.$$

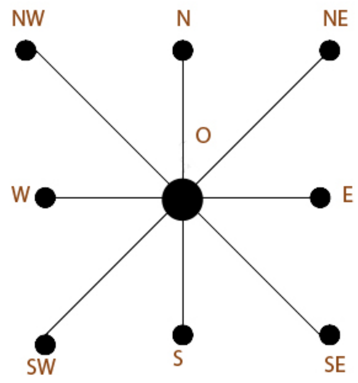
It is clear that this matrix permutes to the two-dimensional linear interpolation matrix.

Similar analysis allows us to develop a two-dimensional version of full weighting and to develop a matrix representation for it. As in the one-dimensional case, each coarse-grid point is defined as the weighted average of all of its neighbors. In particular, if O is a point on the coarse grid we set

$$\mathbf{u}_O^{(2h)} = \frac{1}{4}\mathbf{u}_O^{(h)} + \frac{1}{8}\left(\mathbf{u}_N^{(h)} + \mathbf{u}_E^{(h)} + \mathbf{u}_S^{(h)} + \mathbf{u}_W^{(h)}\right) + \frac{1}{16}\left(\mathbf{u}_{NE}^{(h)} + \mathbf{u}_{SE}^{(h)} + \mathbf{u}_{SW}^{(h)} + \mathbf{u}_{NW}^{(h)}\right),$$

where N, S, W, E, NE, SE, SW, and NW are the neighbors of O on the fine grid as shown in figure 3.

Figure 3: The Neighbors of a Coarse-Grid Point, O



Careful analysis allows us to develop a matrix representation of full-weighting; however, in practice it suffices to notice that the matrix representation is given by $R = \frac{1}{4}P^T$.

We now have the tools we need to begin applying multigrid methods to two-dimensional problems. Later, we will need a different type of prolongation operator, but for now linear interpolation will suffice. We first consider the simplest multigrid method, the Two Grid Scheme. As its name suggests, the Two-Grid Scheme uses two levels of discretization, the first with grid spacing h, the second with grid spacing $2h$. As mentioned earlier, weighted Jacobi relaxation will be more effective for solving $A^{(h)}\mathbf{u}^{(h)} = \mathbf{f}^{(h)}$ if we have a good initial guess.

Intuitively, since $u(x, y)$ is a continuous function, $\mathbf{u}^{(h)}$ should approximately equal $P\mathbf{u}^{(2h)}$. Similarly, $\mathbf{e}^{(h)}$ should approximately equal $P\mathbf{e}^{(2h)}$. Since $\mathbf{u}^{(h)} = \mathbf{v}^{(h)} + \mathbf{e}^{(h)}$, finding $\mathbf{e}^{(h)}$ is equivalent to finding $\mathbf{u}^{(h)}$. Therefore, if can find $\mathbf{e}^{(2h)}$, we can find $\mathbf{u}^{(2h)}$ and know how to make a good initial guess of $\mathbf{u}^{(h)}$.

This, of course, raises a new question. How do we find $\mathbf{e}^{(2h)}$? We compute the residual on the coarse grid by restricting the residual on the fine grid, that is, $\mathbf{r}^{(2h)} = R\mathbf{r}^{(h)}$, where R is a matrix representing restriction. Then, we find the coarse-grid error, by solving $A^{(2h)}\mathbf{e}^{(2h)} = \mathbf{r}^{(2h)}$. We then overwrite $\mathbf{v}^{(h)} \leftarrow \mathbf{v}^{(h)} + P\mathbf{e}^{(2h)}$. If $P\mathbf{e}^{(2h)}$ is a good representation of $\mathbf{e}^{(h)}$, then this correction will be highly effective. One way to help ensure that $P\mathbf{e}^{2h}$ is a good representation of $\mathbf{e}^{(h)}$ is have $\mathbf{e}^{(h)}$ be smooth. Thus, we begin the Two-Grid Scheme by relaxing $A\mathbf{e}^{(h)} = \mathbf{r}^{(h)}$ until the error is smooth. This gives us the two-grid scheme.

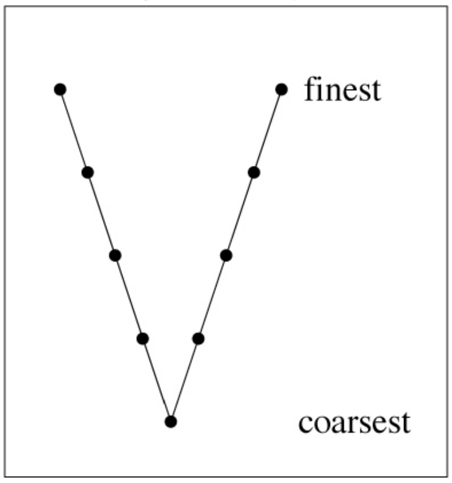
1) Start with the fine-grid equation, $A^{(h)}\mathbf{u}^{(h)} = \mathbf{f}^{(h)}$.
   With initial guess $\mathbf{v}^{(h)}$, relax the fine-grid equation ν_1 times.
2) Compute the fine-grid residual, $\mathbf{r}^{(h)} = \mathbf{f}^{(h)} - A^{(h)}\mathbf{v}^{(h)}$, and
   calculate $\mathbf{r}^{(2h)} = R\mathbf{r}^{(h)}$.
3) Solve $A^{(2h)}\mathbf{e}^{(2h)} = \mathbf{r}^{(2h)}$
4) Set $\mathbf{e}^{(h)} = P\mathbf{e}^{(2h)}$ and overwrite $\mathbf{v}^{(h)} \leftarrow \mathbf{v}^{(h)} + \mathbf{e}^{(h)}$.
5) Relax $A^{(h)}\mathbf{u}^{(h)} = \mathbf{f}^{(h)}$ ν_2 times with the updated value of $\mathbf{v}^{(h)}$ as the initial guess.

This process will return a better guess of the exact solution. Iterating this process will eventually lead us to a guess which is arbitrarily close to the exact solution. However, we can improve this process through recursion by creating what is known as the V-Cycle. The slowest part of the two-grid scheme is step 3, solve $A^{(2h)}\mathbf{e}^{(2h)} = \mathbf{r}^{(2h)}$, since in the two-grid scheme, we solve this equation directly. However, in the V-cycle, we solve this equation using the two-grid scheme. That is, we relax $A^{(2h)}\mathbf{e}^{(2h)} = \mathbf{r}^{(2h)}$, ν_1 times, then restrict $\mathbf{r}^{(2h)}$ to define $\mathbf{r}^{(4h)}$, and solve $A^{(4h)}\mathbf{e}^{(4h)} = \mathbf{r}^{(4h)}$, etc.. Naturally, we again solve this equation using grids with $4h$ and $8h$ spacing, etc. We continue this process until we get to the coarsest grid, which has $2^l h$ spacing. On the coarsest grid, we solve $A^{(2^l h)}\mathbf{e}^{(2^l h)} = \mathbf{r}^{(2^l h)}$ directly by Gaussian elimination or a similar method. We are able to do this efficiently because the problem size is small on the coarsest grid. Therefore, the V-cycle with l levels of discretization can be defined recursively as

1) Start with the fine-grid equation, $A^{(h)}\mathbf{u}^{(h)} = \mathbf{f}^{(h)}$.
   With initial guess $\mathbf{v}^{(h)}$, relax the fine-grid equation ν_1 times.
2) Compute the fine-grid residual, $\mathbf{r}^{(h)} = \mathbf{f}^{(h)} - A^{(h)}\mathbf{v}^{(h)}$, and
   calculate $\mathbf{r}^{(2h)} = R\mathbf{r}^{(h)}$.
3) If the current grid, Ω, is the coarsest grid, solve $A^{(\Omega)}\mathbf{e}^{(\Omega)} = \mathbf{r}^{(\Omega)}$directly.
   Otherwise, apply V-cycle to, $A^{(\Omega)}\mathbf{e}^{(\Omega)} = \mathbf{r}^{(\Omega)}$.
4) Overwrite $\mathbf{v}^{(h)} \leftarrow \mathbf{v}^{(h)} + P\mathbf{e}^{(2h)}$.
5) Relax $A^{(h)}\mathbf{u}^{(h)} = \mathbf{f}^{(h)}$ ν_2 times with the updated value of $\mathbf{v}^{(h)}$ as the initial guess.

Another multigrid method, the F-Cycle, consists of a sequence of V-cycles, each using more grids. The V-cycle starts on the grid with h spacing, Ω^h, and uses a series of coarser grids with the coarsest being $\Omega^{2^l h}$, for some integer l. We shall write V^{Ω_1} to refer to a V-cycle which uses Ω_1 as its finest grid. The F-Cycle, which uses l grids, is defined as follows.

Figure 4: V-Cycle



- Start with $A^{(h)}\mathbf{u}^{(h)} = \mathbf{f}^{(h)}$. Relax ν_1 times. Compute residual.
- Set $\mathbf{r}^{(2h)} = R\mathbf{r}^{(h)}$. Relax $A^{(2h)}\mathbf{e}^{(2h)} = \mathbf{r}^{(2h)}\nu_1$ times. Compute residual.
- Set $\mathbf{r}^{(4h)} = R\mathbf{r}^{(2h)}$. Relax $A^{(4h)}\mathbf{e}^{(4h)} = \mathbf{r}^{(4h)}\nu_1$ times. Compute residual.

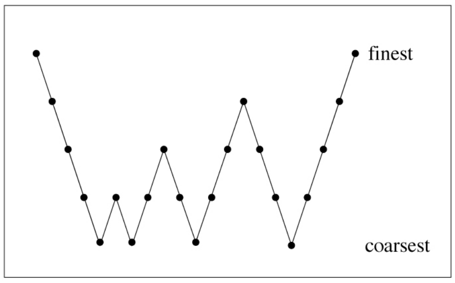
$\vdots$

- Set $\mathbf{r}^{(2^l h)} = R\mathbf{r}^{(2^{l-1}h)}$. Solve $A^{(2^l h)}\mathbf{e}^{(2^l h)} = \mathbf{r}^{(2^l h)}\ \nu_1$ directly. Compute residual.
- Update $\mathbf{e}^{(2^{l-1}h)} \leftarrow \mathbf{e}^{(2^{l-1}h)} + P\mathbf{e}^{(2^l h)}$. Update $\mathbf{e}^{(2^{l-1}h)}$ with $V^{\Omega^{2^{l-1}h}}$.
- Update $\mathbf{e}^{(2^{l-2}h)} \leftarrow \mathbf{e}^{(2^{l-2}h)} + P\mathbf{e}^{(2^{l-1}h)}$. Update $\mathbf{e}^{(2^{l-2}h)}$ with $V^{\Omega^{2^{l-2}h}}$.

$\vdots$

- Update $\mathbf{e}^{(2h)} \leftarrow \mathbf{e}^{(2h)} + P\mathbf{e}^{(4h)}$. Update $\mathbf{e}^{(2h)}$ with $V^{\Omega^{2h}}$.
- Update $\mathbf{e}^{(h)} \leftarrow \mathbf{e}^{(h)} + P\mathbf{e}^{(2h)}$. Relax $A^{(h)}\mathbf{e}^{(h)} = \mathbf{r}^{(h)}\ \nu_2$ times.

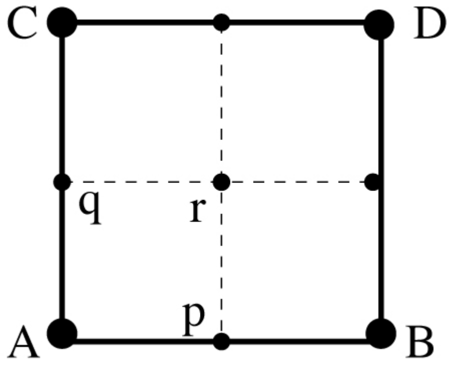
Figure 5: F-Cycle



Now let us consider $A^{(2h)}$. Earlier, we said if $A^{(h)}$ was the discretization of a differential operator on the grid with h spacing, $\Omega^{(h)}$, then $A^{(2h)}$ was the discretization of the same differential operator on the grid with $2h$ spacing, $\omega^{(2h)}$. This definition is a good way to initially understand multigrid methods; however, it is not always used in practice. One obvious disadvantage of this definition is that it is only defined for systems resulting from a PDE. This motivates us to determine a new method of defining $A^{(2h)}$, variational coarsening.

Suppose we start with the equation $A^{(h)}u^{(h)} = f^{(h)}$. If P is a prolongation operator, then we can naturally substitute $Pu^{(2h)}$ for $u^{(h)}$ to get $A^{(h)}Pu^{(2h)} = f^{(h)}$. Left multiplying both sides by R and using the associativity of matrix multiplication, this gives us $(RA^{(h)}P)u^{(2h)} = Rf^{(h)}$. Substituting $f^{(2h)}$ for $Rf^{(h)}$ gives us $(RA^{(h)}P)u^{(2h)} = f^{(2h)}$, which implies $A^{(2h)} = RA^{(h)}P$.

For the domain wall problem, we will use a different type of prolongation operator, known as de Zeeuw Prolongation [10], which has proven effective for many Helmholtz type problems [2]. For standard linear interpolation, we take a fine-grid point which also lies on the coarse grid, $v^{(h)}_{2i,2j}$, to be equal to $v^{(2h)}_{i,j}$. For a fine-grid point that lies directly between two coarse-grid points, the value at the fine-grid point is computed by averaging the value at the two coarse-grid points. Similarly, for points on the fine grid which are equally spaced between four coarse-grid points, the value of v^h at the fine-grid point is taken to be the average of the four neighboring coarse-grid points.



In the notation of figure 0.4, this can be written

$$v^{(h)}_A = v^{(2h)}_A, \quad v^{(h)}_p = \frac{1}{2}(v^{(2h)}_A + v^{(2h)}_B), \quad v^{(h)}_q = \frac{1}{2}(v^{(2h)}_A + v^{(2h)}_C), \quad v^{(h)}_r = \frac{1}{4}(v^{(2h)}_A + v^{(2h)}_B + v^{(2h)}_C + v^{(2h)}_D).$$

This can be naturally generalized into the class of weighted interpolation operators, in which fine-grid points are defined by a weighted averaging of their neighbors on the coarse grid. In general, these operators can be written as

$$v^{(h)}_A = v^{(2h)}_A, \quad v^{(h)}_p = \frac{w_A v^{(2h)}_A + w_B v^{(2h)}_B}{w_A + w_B}, \quad v^{(h)}_q = \frac{w_A v^{(2h)}_A + w_C v^{(2h)}_C}{w_A + w_C},$$
$$v^{(h)}_r = \frac{w_A v^{(2h)}_A + w_B v^{(2h)}_B + w_C v^{(2h)}_C + w_D v^{(2h)}_D}{w_A + w_B + w_C + w_D}.$$

Note that the matrix representation of any prolongation operator of this type will have the same locations of non-zero entries as the matrix representing linear interpolation. De Zeeuw Prolongation is a type of operator-dependent prolongation that is closely related to this class. Operator-dependent refers to the fact that the weights are computed based off of the values of the entries of the matrix, A, which represents the linear system we are trying to solve. The weights in the i^{th} row of the de Zeeuw prolongation matrix, P, will be based off of the i^{th} row of A. In particular, if A is n^2 by n^2, and $\mathbf{v}$ is a n^2 by 1 vector resulting from discretizing a two-dimensional function on a grid with spacing $h = \frac{1}{n}$, we are interested in the entries of this row which act on the $\mathbf{v}_i$ and its neighbors, when $\mathbf{v}$ is left-multiplied by A.

One goal of de Zeeuw prolongation is that if $\mathbf{v}_i$ is a point on the fine-grid such that none of its cardinal neighbors, N, E, S, or W, are coarse-grid points, then we want $\left(A^{(h)}Pv^{(2h)}\right)_i = 0$. This condition is intuitively desirable since such points are the "farthest" from the coarse grid. Thus, they are the points most likely to be misrepresented by prolongation. However, this condition controls what happens at these points which increases the overall efficiency of our algorithm.

To gain intuition, we first consider this condition in one dimension. In one dimension, the desired condition becomes that if $\mathbf{v}_i^{(h)}$ is not a coarse-grid point, we want $(AP\mathbf{v}^{(2h)})_i = 0$. Since we are interested in discretized differential operators, we will assume that A is tridiagonal. Thus the necessary condition is given by

$$\begin{aligned} A_{i,i-1}^{(h)}\mathbf{v}_{i-1}^{(h)} + A_{i,i}^{(h)}\mathbf{v}_i^{(h)} + A_{i,i+1}^{(h)}\mathbf{v}_{i+1}^{(h)} &= 0 \\ \Leftrightarrow \mathbf{v}_i^{(h)} &= -\frac{A_{i,i-1}^{(h)}\mathbf{v}_{i-1}^{(h)} + A_{i,i+1}^{(h)}\mathbf{v}_{i+1}^{(h)}}{A_{i,i}^{(h)}}. \end{aligned}$$

Moreover, $\mathbf{v}_{i-1}^{(h)}$ and $\mathbf{v}_{i+1}^{(h)}$ are coarse-grid points, so we can already know their value. Thus we may determine the value of $\mathbf{v}_i^{(h)}$ by setting

$$\begin{aligned} \mathbf{v}_i^{(h)} &= w_{i-1}\mathbf{v}_{i-1}^{(h)} + w_{i+1}\mathbf{v}_{i+1}^{(h)}, \\ \text{where } w_{i-1} &= -\frac{A_{i,i-1}^{(h)}}{A_{i,i}^{(h)}} \qquad w_{i+1} = -\frac{A_{i,i+1}^{(h)}}{A_{i,i}^{(h)}}. \end{aligned}$$

For the two-dimensional case, we will again want to determine the Prolongation weights based off of A so that points where A is larger in absolute value are given larger weight. It is useful to think of the neighbors of $\mathbf{v}_i$ in terms of the cardinal and ordinal directions similar to the way they are displayed in figure 3. Note that O is a fine-grid point which is also a coarse-grid point, the cardinal directions are located directly between 2 coarse-grid points, and that the ordinal directions are located between 4 coarse-grid points. We will refer to the entry of A, which acts on a neighbor of a point, x, with the syntax a_x^d, where d is some direction. For example, if $p = \mathbf{v}_i$, then a_p^O, refers to the entry of A which acts of p itself, $A_{i,i}$.

For a point, p, whose West and East neighbors, A, and B, are coarse-grid points, we define

$$\begin{aligned} d_w &= \max(|a_p^{sw} + a_p^w + a_p^{nw}|, |a_p^{sw}|, |a_p^{nw}|) \\ d_e &= \max(|a_p^{se} + a_p^e + a_p^{ne}|, |a_p^{se}|, |a_p^{ne}|) \\ w_w &= \frac{d_w}{d_w + d_e} \qquad w_e = \frac{d_e}{d_w + d_e}, \end{aligned}$$

and set $v_p^h = w_w v_A^{(2h)} + w_e v_B^{(2h)}$. The intermediate weights d_e and d_w measure how big neighbors of a_p^O are in magnitude. The finalized weights w_w and w_e are introduced so that they sum to one. For a point q, whose South and North neighbors, A and C, are coarse-grid points, we use similar reasoning to set $v_q^h = w_s v_A^{(2h)} + w_n v_C^{(2h)}$, where

$$\begin{aligned} d_s &= \max(|a_p^{sw} + a_p^s + a_p^{se}|, |a_p^{sw}|, |a_p^{se}|) \\ d_n &= \max(|a_p^{ne} + a_p^n + a_p^{nw}|, |a_p^{ne}|, |a_p^{nw}|) \\ w_s &= \frac{d_s}{d_s + d_n} \qquad w_n = \frac{d_n}{d_s + d_n}. \end{aligned}$$

The remaining class of points is those which are not coarse-grid points and do not lie directly between two coarse-grid points. For such a point, r, we want to construct that P so that the entry of $A^{(h)}\mathbf{v}^{(h)}$ corresponding to r is equal to 0. That is, if r is $\mathbf{v}_i^{(h)}$ and $\mathbf{u} = A^{(h)}\mathbf{v}^{(h)}$, we want $\mathbf{u}_i = 0$. Therefore, we need that $A(i,:) \cdot \mathbf{v}^{(h)} = 0$, where $A(i,:)$ is the i^{th} row of A. Since $\mathbf{v}^{(h)} = P\mathbf{v}^{(2h)}$, we get

$$\begin{aligned} 0 = \sum_{j=1}^{n^2} A_{i,j} v_j^{(h)} &= \sum_{j=1}^{n^2} A_{i,j} (Pv^{(2h)})_j = \sum_{j=1}^{n^2} \left(A_{i,j} \sum_{k=1}^{n^2/4} P_{j,k} v_k^{(2h)} \right) \\ &= \sum_{j=1}^{n^2} \sum_{k=1}^{n^2/4} A_{i,j} P_{j,k} v_k^{(2h)} = \sum_{k=1}^{n^2/4} \left(\sum_{j=1}^{n^2} A_{i,j} P_{j,k} \right) v_k^{(2h)}. \end{aligned}$$

A sufficient criterion for this to hold is

$$0 = \sum_{j=1}^{n^2} A_{i,j} P_{j,k} \text{ for all } k.$$

Since A results from discretizing a PDE, $A_{i,j} = 0$ except at i, j such that $\mathbf{v}_j^{(h)}$ is a neighbor of $\mathbf{v}_i^{(h)}$. Moreover, if $\mathbf{v}_i^{(h)}$ is a point that does not lie on the coarse grid or directly in between two coarse-grid points, then we have already defined P in all of the rows corresponding to a neighbor of v_i. For each index i and direction d, we will let i^d be the index of $\mathbf{v}_i^{(h)}$'s neighbor in the d direction. For example, A_{i,i^w} is equivalent to $A_{i,i-n}$ in most rows. Also, note that i^O is equivalent to i, since $\mathbf{v}_i$'s neighbor in the O direction is just $\mathbf{v}_i$. Plugging in the zeroes in A, our condition becomes

$$0 = \sum A_{i,i^d} P_{i^d,k} \Rightarrow P_{i,k} = P_{i^O,k} = -\frac{1}{A_{i,i}} \sum_{d \neq O} A_{i,i^d} P_{i^d,k} \text{ for all } k.$$

Thus, we have completely defined the de Zeeuw Prolongation matrix, P, which we will be using in our F-cycle scheme. However, we will not be using an F-cycle directly in our attempts to solve the domain wall system. Instead we will be using our F-cycle as a preconditioner for a BICGSTAB algorithm.

BICGSTAB is one of a family of iterative methods known as polynomial methods. In such a method, we pick an initial guess, $\mathbf{x}_0$, to approximate the actual solution, $\mathbf{x}$, of a linear system, $A\mathbf{x} = \mathbf{b}$. Setting $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, we seek to find $c_0, c_1, c_2 \ldots$ such that

$$\mathbf{x} = \mathbf{x}_0 + c_0\mathbf{r}_0 + c_1 A\mathbf{r}_0 + c_2 A^2\mathbf{r}_0 + \ldots$$

We will be able to do this if $(\mathbf{x} - \mathbf{x}_0) \in \mathit{span}\{\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, \ldots\}$. This motivates us to define the m-dimensional Krylov Space as,

$$K_m(A, \mathbf{r}_0) = \mathit{span}\{\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, \ldots, A^{m-1}\mathbf{r}_0\}.$$

While it is clear that $\{\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, \ldots, A^{m-1}\mathbf{r}_0\}$ is basis for $K_m(A, \mathbf{r}_0)$, unfortunately, it is not an orthogonal basis. The Lanczos Algorithm provides an efficient way of producing an orthogonal basis of $K_m(A, \mathbf{r}_0)$, in the special case that A is Hermitian. For the more general case, we need to use the Lanczos method for Bi-orthoganalization, which generates bases, $\{\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_m\}$ and $\{\mathbf{w}_1, \mathbf{w}_2, \ldots \mathbf{w}_m\}$ for $K_m(A, \mathbf{r}_0)$ and $K_m(A^*, \mathbf{w}_0)$, respectively, such that $\mathbf{v}_i \cdot \mathbf{w}_i = 1$ and $\mathbf{v}_i \cdot \mathbf{w}_j = 0$, if $i \neq j$, where $\mathbf{w}_0$ is some vector, and A^* denotes the Hermitian transpose of A. Letting, $V_m = (\mathbf{v}_1|\mathbf{v}_2|\ldots|\mathbf{v}_m)$ and $W_m = (\mathbf{w}_1|\mathbf{w}_2|\ldots|\mathbf{w}_m)$, it follows that $W_m^* V_m = I_m$.

The BICG algorithm picks $\mathbf{v}_1 = \frac{\mathbf{r}_0}{\beta}$, for some scalar β, and searches for a solution of the form $\mathbf{x} = \mathbf{x}_0 + V_m \mathbf{y}_m$. Plugging this in implies

$$\begin{aligned} &A\left(\mathbf{x}_0 + V_m \mathbf{y}_m\right) = \mathbf{b} \\ \Leftrightarrow &A V_m \mathbf{y}_m = \mathbf{b} - A\mathbf{x}_0 = \mathbf{r}_0 = \beta \mathbf{v}_1 \\ \Rightarrow &W^* A V_m \mathbf{y}_m = \beta W_m^* \mathbf{v}_1. \end{aligned}$$

It turns out that $W^* A V_m$ is tridiagonal, which should allow us to solve the system efficiently. However, this method is somewhat computationally expensive and tends to have unstable convergence. The Conjugate Gradient Squared (CGS) algorithm chooses $\mathbf{w}_0 = \mathbf{r}_0$. This reduces the computational cost of the algorithm since the entries of $A^*\mathbf{w}_0$ are simply the conjugates of the entries of $A\mathbf{r}_0$. Unfortunately, the CGS method also tends to suffer from unstable convergence. The BICGSTAB algorithm is created by choosing bases such that the norm of the residual is minimized after each iteration. For a system $A\mathbf{x} = \mathbf{b}$, we start with initial guess, $\mathbf{x}_0$, and

```
• Set $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$
• Choose an arbitrary vector $\mathbf{r}^*$ such that $(\mathbf{r}_0, \mathbf{r}^*) \neq 0$
• $\rho_0 = a = \omega_0 = 1$
• $\mathbf{v}_0 = \mathbf{p}_0 = \overrightarrow{\mathbf{0}}$
• For $i = 1, 2, 3, \ldots$
    • $\rho_i = (\mathbf{r}^*, \mathbf{r}_{i-1})$
    • $\beta = (\rho_i/\rho_{i-1})(a/\omega_{i-1})$
    • $\mathbf{p}_i = \mathbf{r}_{i-1} + \beta(\mathbf{p}_{i-1} - \omega_{i-1}\mathbf{v}_{i-1})$
    • $\mathbf{v}_i = A\mathbf{p}_i$
    • $a = \rho_i/(\mathbf{r}^*, \mathbf{v}_i)$
    • $\mathbf{s} = \mathbf{r}_{i-1} - a\mathbf{v}_i$
    • If $||\mathbf{s}||$ is small enough, quit
    • $\mathbf{t} = A\mathbf{s}$
    • $\omega_i = (\mathbf{t}, \mathbf{s})/(\mathbf{t}, \mathbf{t})$
    • $\mathbf{x}_i = \mathbf{x}_{i-1} + a\mathbf{p}_i + \omega_i s$
    • $\mathbf{r}_i = \mathbf{s} - \omega_i \mathbf{t}$,
    • If $\mathbf{x}_i$ is accurate enough, quit
```

where $(\mathbf{v}, \mathbf{w}) = \mathbf{v} \cdot \mathbf{w} = \sum \mathbf{v}_i \overline{\mathbf{w}_i}$.

In practice, we often use BICGSTAB with a preconditioner. If M is some invertible matrix, then $MA\mathbf{x} = M\mathbf{b}$ has the same solutions as $A\mathbf{x} = \mathbf{b}$. Thus, we may apply BICGSTAB to whichever of those systems is easier to solve. This raises a natural question. How do we pick M so that $MA\mathbf{x} = M\mathbf{b}$ is easier to solve than $A\mathbf{x} = \mathbf{b}$?

For any Krylov method, at the m^{th} stage of iteration, our current guess $\mathbf{x}_m$ is given

$$\mathbf{x}_m = \mathbf{x}_0 + p_{m-1}(A)\mathbf{r}_0,$$

where $p_{m-1}(x)$ is some polynomial of degree $m-1$. Subtracting both sides from the correct solution, $\mathbf{x}$, and using the identity $A\mathbf{e}_0 = \mathbf{r}_0$ we see

$$\begin{aligned} \mathbf{x} - \mathbf{x}_m &= \mathbf{x} - [\mathbf{x}_0 + p_{m-1}(A)\mathbf{r}_0] \\ \Rightarrow \mathbf{e}_m &= \mathbf{e}_0 - p_{m-1}(A)A\mathbf{e}_0 \\ \Rightarrow \mathbf{e}_m &= p_m^{(1)}(A)\mathbf{e}_0, \end{aligned}$$

where $p_m^{(1)}(x)$ is some m^{th} degree polynomial with $p_m^{(1)}(0) = 1$. Since the eigenvectors of A form a basis for $span(A)$, we may write

$$\mathbf{e}_0 = \sum_{j=1}^{n} c_j \mathbf{v}^{(j)},$$

where $\mathbf{v}^{(j)}$ is an eigenvector of A with eigenvalue λ_j. Thus,

$$\mathbf{e}_m = p_m^{(1)}(A)\mathbf{e}_0 = \sum_{j=1}^{n} c_j p_m^{(1)}(A)\mathbf{v}^{(j)} = \sum_{j=1}^{n} c_j p_m^{(1)}(\lambda_j)\mathbf{v}^{(j)}.$$

Therefore, the error after m iterations will be small is $p_m^{(1)}(\lambda_j)$ is small for each λ_j.

Thus, we may find a Krylov method that converges within m iterations if there is a polynomial, $p(x)$, with degree m and $p(0) = 1$ such that $p(\lambda_j)$ is sufficiently small at each eigenvalue, λ_j. Intuitively, it is clear that the closer together the eigenvalues are, the lower the order of the polynomial we will need to meet this criteria. Thus, it is our goal to find M such that the eigenvalues are MA "bunched" closer together than the eigenvalues of A.

During the BICGSTAB algorithm, there are a number of steps which involve multiplying some vector by A. If we are using a preconditioner, M, each of those steps becomes instead to multiply the same vector by MA. Since matrix multiplication is associative, $(MA)\mathbf{v} = M(A\mathbf{v})$, for any vector, $\mathbf{v}$. Computationally though, it requires less floating point operations to perform matrix-vector multiplication as opposed to matrix-matrix multiplication. Thus, in practice, we never actually form MA.

Since we never actually form MA, we don't need M to actually be a matrix, instead it may be any function which takes in a vector and produces another vector. In particular, the preconditioner may be a Multi-Grid cycle. It has been shown that Helmholtz problems with high wave numbers can be solved effectively by preconditioning with an F-cycle applied to the same problem with an imaginary shift [2]. Inspired by this success, we test whether this same approach can be applied to the domain wall problem.

Now, let us consider how we may solve the entire domain wall system once we have an effective technique for solving the gauged Helmholtz problem, i.e. the A block of the domain wall [5]. Recall the domain wall is written,

$$W(m,\rho) = \begin{bmatrix} M_\rho & -P_- & & & +mP_+ \\ -P_+ & M_\rho & -P_- & & \\ & -P_+ & M_\rho & -P_- & \\ & & \ddots & \ddots & \ddots \\ +mP_- & & & -P_+ & M_\rho \end{bmatrix}.$$

Expanding this out with $l = 4$, we get

$$W = \left[\begin{array}{cc|cc|cc|cc}
A & B & 0 & 0 & 0 & 0 & mI & 0 \\
-B^* & A & 0 & -I & 0 & 0 & 0 & 0 \\ \hline
-I & 0 & A & B & 0 & 0 & 0 & 0 \\
0 & 0 & -B^* & A & 0 & -I & 0 & 0 \\ \hline
0 & 0 & -I & 0 & A & B & 0 & 0 \\
0 & 0 & 0 & 0 & -B^* & A & 0 & -I \\ \hline
0 & 0 & 0 & 0 & -I & 0 & A & B \\
0 & mI & 0 & 0 & 0 & 0 & -B^* & A
\end{array}\right].$$

We can permute the matrix separating the entries by parity to get

$$W' = \left[\begin{array}{cccc|cccc}
A & 0 & 0 & mI & B & 0 & 0 & 0 \\
-I & A & 0 & 0 & 0 & B & 0 & 0 \\
0 & -I & A & 0 & 0 & 0 & B & 0 \\
0 & 0 & -I & A & 0 & 0 & 0 & B \\ \hline
-B^* & 0 & 0 & 0 & A & -I & 0 & 0 \\
0 & -B^* & 0 & 0 & 0 & A & -I & 0 \\
0 & 0 & -B^* & 0 & 0 & 0 & A & -I \\
0 & 0 & 0 & -B^* & mI & 0 & 0 & A
\end{array}\right],$$

where the top left corner is taken from odd columns and odd rows of W, the top right corner is taken from even columns and odd rows of W, and so forth. Denoting the top left block $\mathbb{A}$ and the top right block $\mathbb{B}$, we may write

$$W' = \begin{bmatrix} \mathbb{A} & \mathbb{B} \\ -\mathbb{B}^* & \mathbb{A}^* \end{bmatrix}.$$

Thus, we may rewrite the system $W\mathbf{v} = \mathbf{f}$

$$\begin{bmatrix} \mathbb{A} & \mathbb{B} \\ -\mathbb{B}^* & \mathbb{A}^* \end{bmatrix} \begin{pmatrix} \mathbf{v}_o \\ \mathbf{v}_e \end{pmatrix} = \begin{pmatrix} \mathbf{f}_o \\ \mathbf{f}_e \end{pmatrix},$$

where $\mathbf{v}_o, \mathbf{v}_e, \mathbf{f}_o, \mathbf{f}_e$ are the vectors containing the odd and even entries of $\mathbf{v}$ and $\mathbf{f}$ respectively. Thus, we have

$$\begin{aligned} \mathbb{A}\mathbf{v}_o &= \mathbf{f}_o - \mathbb{B}\mathbf{v}_e, \\ \mathbb{A}^*\mathbf{v}_e &= \mathbf{f}_e + \mathbb{B}^*\mathbf{v}_o. \end{aligned}$$

Assuming we can solve the systems with matrix A efficiently, we will be able to solve $\mathbb{A}$ and $\mathbb{A}^*$ efficiently since they are block bidiagonal if we treat the mass term as negligible. Thus, we may solve the entire system quickly using an Uzawa iteration.

0.5 Results

We will test the effectiveness of our F-Cycle, both as a direct means of solution, and as a preconditioner for a BICGSTAB scheme. First, we shall test the F-cycle directly on the discretization of the two-dimensional Poisson equation, $-\nabla^2 u(x,y) = f(x,y)$. There are 4 parameters in which this scheme can vary. The most important is, of course, $n = \frac{1}{h}$, which represents the level of discretization. We must also consider the value of the weight, ω, within the weighted Jacobi Relaxation Scheme and the number of relaxations on the “way up” and “way down”, ν_1 and ν_2, within the

F-cycle and V-cycle schemes. We will fix ν_1 and ν_2 to be equal to one. We will allow n to be 32, 64, and 128. The "best" value of ω will be determined on the smallest grid, i.e. $n = 32$, and we will use this value when testing fine grids.

We shall start off with a random initial guess and a zero right-hand side and consider our method to have converged when the residual has been decreased (in norm) by a relative factor of 10^6. Upon inspection, it appears that unweighted relaxation, i.e. $\omega = 1$, is effective for applying the F-cycle to this problem.

Table 1: F-Cycle iterations necessary for convergence, Poisson problem, n=32

ω	Iterations
$\frac{1}{3}$	15
$\frac{2}{3}$	13
$\frac{7}{9}$	12
$\frac{8}{9}$	11
1	10

Taking $\omega = 1$ and applying our F-cycle to various size grids, we see that the F-cycle is an effective means of solving the discrete Poisson problems for levels of discretization corresponding to $n = 32$, 64 and 128. In fact, the number of iterations necessary to achieve convergence does not seem to depend on the grid size.

Table 2: F-Cycle iterations necessary for convergence, Poisson problem, $\omega = 1$

n	Iterations
32	10
64	10
128	10

When testing effectiveness of a the F-cycle as a precondition for BICGSTAB in solving the Poisson problem, we cannot use a zero right-hand side since standard BICGSTAB algorithms are set to automatically return the correct answer of $\overrightarrow{\mathbf{0}}$ without running any iterations. Therefore, we use a right-hand side given by the discretization of $\cos(2\pi x)\sin(2\pi y)$. We choose $\cos(2\pi x)\sin(2\pi y)$ since we expect $f(x, y)$ to be periodic. ω is again taken to be 1. As shown below, BICGSTAB is very effective for solving the Poisson problem for all grids. Note that converging on a half-iteration means that BICGSTAB exited after testing the norm of $||\mathbf{s}||$.

Table 3: BICGSTAB iterations necessary for convergence, Poisson problem

n	Iterations
32	3
64	2.5
128	3

Poisson's equation is, of course, the special case of Helmholtz's equation $-\left(\nabla^2 - \sigma\right) u(x, y) = f(x, y)$, where $\sigma = 0$. We are interested in solving the case where $\sigma = k^2(1 + b\imath)$, $k \in \mathbb{R}$. For any given n, we can uniquely define k in terms of $a = \frac{k}{n}$. If we think of σ as a function of a and b, then we can think of k as the wave number associated with the system.

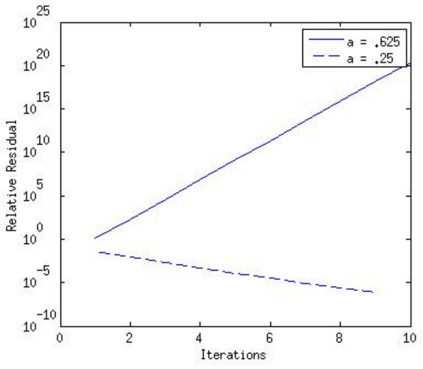
When we apply our F-cycle to the Helmholtz problem, with grid spacing $n = 32$, we see that the F-cycle is effective for all values of b when $a = .25$. However, for $a = .625$, the F-cycle is not effective for any value of b. Figure (6) shows the relative residual after the first ten iterations on a log scale for both $a = .25$ and $a = .625$ with $b = .5$. As you can see, for $a = .25$, the relative residual quickly goes to zero; however, when $a = .625$ the relative residual grows rapidly.

Table 4: F-Cycle iterations necessary for convergence, Helmholtz problem, n=32

a\b	.1	.3	.5	.8
0	10	10	10	10
.25	10	10	10	10
.625	*	*	*	*

.

Figure 6: Relative residual, F-Cycle, Helmholtz problem



When we apply preconditioned BICGSTAB to this system with $n = 32$, we see convergence in each case except for when $a = .625$ and $b = .1$ or $.3$. Note that when $a = 0$, the choice of b does not matter since $a = 0$ implies that $k = 0$. It also appears that BICGSTAB is more effective for smaller values of a and larger values of b. We expect this trend to hold in general.

Table 5: BICGSTAB iterations necessary for convergence, Helmholtz problem, n=32

a\b	.1	.3	.5	.8
0	3	3	3	3
.25	3	3	3.5	3.5
.625	*	*	19.5	8

.

Applying preconditioned BICGSTAB to the grids with $n = 64$ and $n = 128$, we again fail to have convergence when $a = .625$ and b is either .1 or .3. Additionally, we do not have have convergence for these values of b for $a = .25$. On the $n = 64$ grid, in the case that $a = .25, b = .1$ or .3, it appears that the algorithm may have converged if we were to have increased the maximum number of iterations since the iterate returned had very small relative residuals. For all of the other cases, it appears that more iterations would have been helpful since the iterates returned had large relative residuals. Note that on the grid with $n = 128$, preconditioned BICGSTAB is more effective when $b = .5$ than it is when $b = .8$.

We then sought to test our F-Cycle on the Gauged Helmholtz Operator, i.e., the A block of the domain wall problem. As mentioned earlier, if we can find a way to effectively solve this system, we will be able to solve the entire domain wall problem efficiently. Since the F-cycle was ineffective

Table 6: BICGSTAB iterations necessary for convergence, Helmholtz problem, n=64

a\b	.1	.3	.5	.8
0	2.5	2.5	2.5	2.5
.25	*	*	33	11.5
.625	*	*	19.5	8

.

Table 7: BICGSTAB iterations necessary for convergence, Helmholtz problem, n=128

a\b	.1	.3	.5	.8
0	3	3	3	3
.25	*	*	33	11.5
.625	*	*	38.5	49.5

.

at solving the regular Helmholtz Operator directly, we do not bother to test it here. We first apply BICGSTAB with our F-Cycle preconditioner to the Gauged Helmholtz problem where our Gauge field is taken to be 99.99% constant and .01% random. That is, each entry in the gauge field, G is determined by the rule that $G_{i,j} = .9999 + .0001r_{i,j}$, where $r_{i,j}$ is a random number between 0 and 1.

Table 8: BICGSTAB iterations necessary for convergence, gauged Helmholtz problem, .01% random, n=32

a\b	.1	.3	.5	.8
0	*	*	*	*
.25	*	*	*	*
.625	*	*	92	26.5

For all grid sizes, we are able to find some b for which we achieve convergence when $a = .625$. Moreover, in the cases where we did not achieve convergence, the relative residual of the returned iterate was often nearly small enough to be considered convergence. However, if we let our gauge field be 99% constant and 1% random, we are not able to achieve convergence for any value of a and b on any grid. In the case that BICGSTAB fails to converge, MATLAB's BICGSTAB program returns the iterate with the minimum residual. Tables 11 and 12 display the iterate returned and the relative residual for this case with $n = 32$.

While we do not achieve convergence, we do notice an interesting trend that our preconditioning is less effective on the gauged Poisson problem ($a = 0$) than on the gauged Helmholtz problem with $a = .625$. This is of interest because the gauged Poisson problem can be solved effectively using just an F-Cycle with an adaptive interpolation operator[4][6]. Therefore, we believe that BICGSTAB will be able to solve the general gauged Helmholtz problem with an F-Cycle preconditioner that uses adaptive interpolation.

Table 9: BICGSTAB iterations necessary for convergence, gauged Helmholtz problem, .01% random, n=64

a\b	.1	.3	.5	.8
0	*	*	*	*
.25	*	*	39	17
.625	*	*	73	39.5

Table 10: BICGSTAB iterations necessary for convergence, gauged Helmholtz problem, .01% random, n=128

a\b	.1	.3	.5	.8
0	*	*	*	*
.25	*	*	*	28.5
.625	*	*	58.5	84.5

Table 11: BICGSTAB iterate returned, gauged Helmholtz problem, 1% random

a\b	.1	.3	.5	.8
0	5	5	5	5
.25	79	32	71	29
.625	77	100	73	59

Table 12: Relative residual returned by BICGSTAB, gauged Helmholtz problem, 1% random

a\b	.1	.3	.5	.8
0	.32	.32	.32	.32
.25	5.1×10^{-4}	1.5×10^{-4}	3.7×10^{-5}	9.8×10^{-5}
.625	.58	2.9×10^{-4}	1.5×10^{-5}	1.2×10^{-6}

Bibliography

[1] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial, Second Edition*, SIAM, 2000.

[2] Y. Erlangga, C. Oosterlee, and C. Vuik, *A novel multigrid based preconditioner for heterogeneous helmholtz problems*, SIAM J. Sci. Comput., 27 (2006), pp. 1471–1492.

[3] P. D. Group.

[4] J.Brannick, A. Frommer, K. Kahl, S. MacLachlan, and L. Zikatanov, *Adaptive reduction-based multigrid for nearly singular and highly disordered physicial systems*, Electron. Trans. Numer. Analysis, (2009). Submitted.

[5] M. Kilmer. Working Note.

[6] S. MacLachlan and C. Oosterlee, *Algebraic multigrid solvers for complex-valued matrices*, SIAM J. Sci. Comput., 30 (2008), pp. 1548–1571.

[7] S. P. MacLachlan and C. W. Oosterlee.

[8] G. Strang, *Linear Algebra and Its Applications, Fourth Edition*, Brooks/Cole, 2006.

[9] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, SIAM, 2004.

[10] P. D. Zeeuw, *Matrix-dependent prolongations and restrictions in a blackbox multigrid solver*, Journal of Computational and Applied Mathematics, (1990), pp. 1–27.