An Exploration of Multigrid Methods

Avi Schwarzschild and Andres Soto

Numerical Methods for PDEs 4301

Department of Applied Physics and Applied Math
Columbia University

UNIs: aks2203 and ads2206

Abstract

We used both and one and two dimenssional poisson problems to study multigrid methods for solving partial differential equations. Using iterative solvers for linear systems we show how coarsening the descritization can lead to approximations which converge to the true solution of the PDE with fewer iterations of the solver.

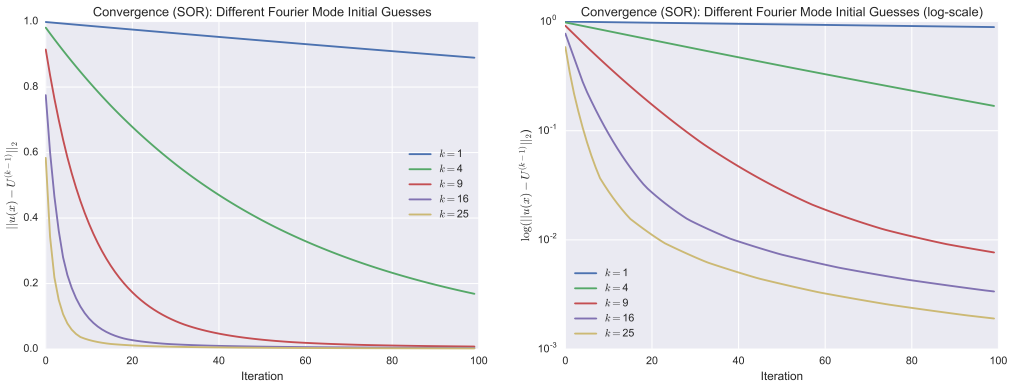
1 Motivation

The motivation for using coarser grids, while convergences analysis shows that finer grids should lead to more accurate approximations, comes from an observation about waves, descritizations, and aliasing. The iterative methods, often referred to as smoothers, smooth out error such that in the earlier iterations it is the high frequency components of the error that vanish first. As the algorithms sweep more times the error gets smoother, containing lower frequencies and tending toward zero. The trouble with the classical iterative methods like this is that low frequencies in the error can take many iterations to smooth. In general these iterative methods are $\mathcal{O}(n^2)$. Thus, some improvement in speed is desired.

The standard iterative methods for solving elliptic PDEs, often have a propety in which the smoothing rate is not only small but also independent of the grid spacing h. Looking at a numerical example in which we apply weighted Jacobi iteration with $\omega = \frac{2}{3}$ to the problem $u''(x) = 0$ with zero boundary conditions gives us insight into this phenomena. Our initial guess is a fourier mode that looks like:

$$w_j = \sin\left(\frac{jk\pi}{N}\right) \quad 0 \leq j \leq N \text{ and } 1 \leq k \leq N-1$$

Here $N-1$ is the number of interior points. A convergence analysis gives us:



We see that error clearly decreases with each relaxation sweep but that the rate of decrease is much more faster for high frequency initial guesses than low frequency ones. Of course the first way to improve a relaxation is to increase the number of iterations, or improve the initial guess, however, relaxations on a coarser grid is less expensive. Moreover, smooth components on a fine grid projected onto a coarser grid look more

oscillatory. If we look at fourier modes associated with a mesh of discretization h, then we have:

$$w^h_{k,2j} = \sin\left(\frac{2jk\pi}{N}\right) = \sin\left(\frac{jk\pi}{N/2}\right) = w^{2h}_{k,j} \quad 1 \leq k < \frac{N}{2}$$

. This seemingly trivial manipulation reeaveals that in passing from a fine grid to coarse grid, the smooth wave becomes more sinusoidal and gains frequency.

From the Shannon Sampling Theorem we know that to retain all of the wave information, we need the descritization to have just over two points per wavelength.[1] The implication of this on our work is that the highest frequency in the error is determined by the mesh grid. Knowing this, we can coarsen the grid so that we have fewer smaple points of the error function and then the low frequencies with be among the higher ones still contained in the coarse-grid error. By smoothing the error on this coarser grid we can eliminate more components of the error than we could without the coarsening. We leave the explanation and implimentation of this for later sections.

2 Multigrid Method

In this section, we will consider the theory and analyze why multigrid methods work so well. We will delve deep into gaining some understanding of the linear algebraic implications of mesh restriction and interpolation. More importantly, we will look at the spectral picture of the multigrid method. The standard cycle of multigrids is the standard V cycle which looks like:

2.1 A Brief Overview

The basic structure of the multigrid method is that the relaxation scheme underlying each layer possess the smoothing property. This motivated the coarser grid due to their slow convergence rate. Multigrid methods further integrate error correction methods by first approximating the residual $r = f - Av$, where v is an approximation and A represents the matrix associated with the iterative method.

There are two main ideas that come into play in multigrid. The firstis that coarser grids can be used to improve initial guesses and is the basis of a strategy called nested iteration. The second idea incorporates utilizing the residual of each smoothing operation at some mesh level in order to correct. This is the basis of coarse grid correction. Both of these strategies together lead us to the Full Multigrid Cycle.

One main question to consider is how integrid transfers work. Common practice in the multigrid literature for 1d and 2d problems is to interpolate and restrict across different mesh grids. These linear mappings are of course defined by matrices which are T^h_{2h} and R^{2h}_h (interpolation and restriction, respectively). Our interpolation matrix in 1d does $T^h_{2h}\mathbf{v}^{2h} = v^{2h}_j$:

$$v^h_{2j} = v^{2h}_j \quad \text{and} \quad v^h_{2j+1} = \frac{1}{2}(v^{2h}_j + v^{2h}_{j+1}) \quad 0 \leq j \leq \frac{N}{2} - 1$$

[1] C. E. Shannon, "Communications in the presence of noise", Proc. IRE, vol. 37, pp. 10-21, Jan. 1949.

In two dimensions this looks like:

$$v^h_{2i,2j} = v^{2h}_{ij} \quad \text{and } v^h_{2i+1,2j} = \frac{1}{2}\left(v^{2h}_{ij} + v^{2h}_{i+1,j}\right) \text{ and } v^h_{2i,2j+1} = \frac{1}{2}\left(v^{2h}_{ij} + v^{2h}_{i,j+1}\right)$$

$$v^h_{2i+1,2j+1} = \frac{1}{4}\left(v^h_{ij} + v^{2h}_{i+1,j} + v^{2h}_{i,j+1} + v^{2h}_{i+1,j+1}\right) \quad 0 \le i,j \le \frac{N}{2} - 1$$

The restrictive matrix in 1d is defined by $R^{2h}_h \mathbf{v}^h = \mathbf{v}^{2h}$ and looks like:

$$v^{2h}_j = \frac{1}{4}\left(v^h_{2j-1} + 2v^h_{2j} + v^h_{2j+1}\right)$$

In two dimensions this is:

$$\begin{aligned} v^{2h}_{ij} = \frac{1}{16}(&v^h_{2i-1,2j-1} + v^h_{2i-1,2j+1} + v^h_{2i+1,2j-1} + v^h_{2i+1,2j+1} + \\ &2\left(v^h_{2i,2j-1} + v^h_{2i,2j+1} + v^h_{2i-1,2j} + v^h_{2i+1,2j}\right) + 4v^h_{2i,2j}) \end{aligned} \tag{1}$$

An interesting property is that these matrices are transposes of each other up to a constant.

Figure 1: The left figure is a standard Multigrid V Cycle. The Right figure is a Full Multigrid Cycle (FMV).

With grid transfers well defined, we can discuss a bit more intergrid transfers. At its core, multigrid methods simply smooth an approximation on some mesh with discretization h and use a coarse grid corrections scheme to understand the error at a coarser mesh to get a better approximation. This is done iteratively in many fashions. Below we see two standard cycles of multigrid, both of which we implemented.

2.2 The Basic Spectral Picture of Multigrid Methods

*Now that we have a better idea of multigrid and their power, we can turn towards understanding some of the mathematical basis behind this powerful method. We will further develop a spectral picture.

Transitions from one discretization to another discretization require a mapping of some kind from the current approximation of the true solution or vector to the same vector on the next mesh. We will adopt the same notation that Briggs introduces. Ω^h represents a grid with spacing $h = \Delta x$, and naturally Ω^{2h} represents a coarser grid. $\mathbf{v}^h$ is an approximation of the true solution $\mathbf{u}^h$, and therefore error is $\mathbf{e}^h = \mathbf{u}^h - \mathbf{v}^h$. The interpolation matrix maps a vector on a coarse mesh, say Ω^{2h} to Ω^h and is denoted T^h_{2h}, while the restriction matrix maps a vector from $\Omega^h \to \Omega^{2h}$ and is denoted by R^{2h}_h.

Assuming, we have $N+1$ total points in our boundary, and $N-1$ interior points, then the restriction matrix R_h^{2h} maps $\mathbb{R}^{N-1} \to \mathbb{R}^{N/2-1}$. Since we have $N-1$ unknown values and are mapping to a lower dimension subspace, then the rank of our restriction matrix is $\frac{N}{2}-1$ and the null space of the restriction matrix is $\frac{N}{2}$. Let us explore how this matrix transforms the eigenvalues of our iterative matrix A^h on mesh Ω^h. The eigenvectors or modes of A^h are given by

$$\mathbf{w}_k^h = \left(\sin\left(\frac{k\pi}{N}\right), \ldots, \sin\left(\frac{(N-1)k\pi}{N}\right)\right)$$

where the jth component of the vector is $w_{k,j}^h = \sin\left(\frac{jk\pi}{N}\right)$. When the restriction operator is applied directly to these vectors we get:

$$R_h^{2h}\mathbf{w}_k^h = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & \\ & 1 & 2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 2 & 1 \end{bmatrix}\mathbf{w}_k^h \tag{2}$$

The jth component of this is:

$$\Rightarrow \left(R_h^{2h}\mathbf{w}_k^h\right)_j = \frac{1}{4}\left(\sin\left(\frac{(j-1)k\pi}{N}\right) + 2\sin\left(\frac{jk\pi}{N}\right) + \sin\left(\frac{(j+1)k\pi}{N}\right)\right) \tag{3}$$

Using the following two trigonmetric identities, $\sin(\theta_1+\theta_2) = \sin\theta_1\cos\theta_2 + \cos\theta_1\sin\theta_2$ and $\cos(2\theta) = 2\cos\theta^2 - 1$, leads us to:

$$\frac{1}{4}\left[2\sin\left(\frac{2jk\pi}{N}\right)\cos\left(\frac{k\pi}{N}\right) + 2\sin\left(\frac{2jk\pi}{N}\right)\right] = \sin\left(\frac{jk\pi}{N/2}\right)\cos^2\left(\frac{k\pi}{2N}\right) \text{ for } 1 \le k \le \frac{N}{2}$$

What we find is that this result holds for the first $k/2$ eigenvectors or one half of the unknowns.

$$R_h^{2h}\mathbf{w}_k^h = \cos^2\left(\frac{k\pi}{2N}\right)\mathbf{w}_k^{2h},\ 1 \le k \le \frac{N}{2} \tag{4}$$

The restriction matrix produces a new kth mode of A^{2h} such that the old kth mode of A^h gets multiplied by a constant.

Applying a similar analysis for when $\frac{N}{2} \le k \le N-1$, yields

$$R_h^{2h}\mathbf{w}_{k'}^h = -\sin^2\left(\frac{k\pi}{2N}\right)\mathbf{w}_k^{2h},\quad 1 \le k \le \frac{N}{2} \text{ where } k' = N - ks$$

Essentially, what is going on here is the phenomena of aliasing since the restriction matrix is acting on the $(N-k)$th mode of A^h and multiplying it by a constant multiple of the . The interpretation behind this result is that the oscillatory modes on Ω^h cannot be represented in the coarser mesh Ω^{2h} and the restriction operator transforms these nodes into relatively smooth modes on the coarser mesh. The reason this matrix has no inverse is because there are two eigenvectors (eigenvector k and eigenvector $N-k$) on A^h that get mapped to the same kth mode on A^{2h}.

INCLUDE SOME GRAPHS HERE TO DESCRIBE THIS ALIASING

We define a set W_k^h that takes the span of these complementary nodes (the $(N-k)$th eigenvector and the kth eigenvector), $W_k^h = span\{\mathbf{w}_k^h, \mathbf{w}_{N-k}^h\}$. We note that:

$$w_{N-k,j}^h = \sin\left(\frac{j(N-k)\pi}{N}\right) = \sin\left(j\pi - \frac{jk\pi}{N}\right)$$

$$\sin(j\pi)\cos\left(\frac{jk\pi}{N}\right) - \cos(j\pi)\sin\left(\frac{jk\pi}{N}\right) = (-1)^{j+1}\sin\left(\frac{jk\pi}{N}\right) = (-1)^{j+1}w_{k,j}^h \tag{5}$$

INCLUDE IMAGE OF COMPLEMENTARY PAIRS

Of course, this two element to one element mapping means that R_h^{2h} contains non-trivial elements in its null space. Let's take the unit vector $\mathbf{e}_i^h$ on Ω^h, and see how it gets transformed by R_h^{2h}.

$$R_h^{2h}\mathbf{e}_i^h = \begin{bmatrix} 1 & 2 & 1 & & & & \\ & & 1 & 2 & 1 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \tag{6}$$

We notice that there are two cases due to overlapping ones in the odd columns of R_h^{2h}. If i is odd in $\mathbf{e}_i^h$, then we have a contribution from two rows whereas if i is even, there is only one row contribution from only the component with 2. We have the following:

$$R_h^{2h}\mathbf{e}_i^h = \begin{cases} \frac{1}{4}\left(\mathbf{e}_{\frac{i-1}{2}}^{2h} + \mathbf{e}_{\frac{i+1}{2}}^{2h}\right) & i \text{ odd} \\ \frac{1}{4}\left(2\mathbf{e}_{\frac{i}{2}}^{2h}\right) & i \text{ even} \end{cases} \tag{7}$$

These unit vectors do not form the basis of the null space because this matrix product is not zero. How about we look at the transformation of these unit vectors under the iterative method first, namely $R_h^{2h}A^h\mathbf{e}_i^h$.

$$R_h^{2h}A^h\mathbf{e}_i^h = R_h^{2h}\left(-\mathbf{e}_{i-1}^h + 2\mathbf{e}_i^h - \mathbf{e}_{i+1}^h\right) \tag{8}$$

If the i is even then $i-1$ and $i+1$ are odd and we'll get extra terms that do not go to zero, whereas if i is odd, we get:

$$R_h^{2h}A^h\mathbf{e}_i^h = -\frac{1}{4}\left(2\mathbf{e}_{\frac{i-1}{2}}^{2h}\right) + \frac{1}{2}\left(\mathbf{e}_{\frac{i-1}{2}}^{2h} + \mathbf{e}_{\frac{i+1}{2}}^{2h}\right) - \frac{1}{2}\mathbf{e}_{\frac{i+1}{2}}^{2h} = 0 \tag{9}$$

We see that the vectors $\mathbf{n}_i = A^h\mathbf{e}_i^h$ when i is odd form the basis of the null space of the full weighting operator. The null space contains both smooth and oscillatory modes of A^h because the vector $A^h\mathbf{e}_i^h$, which is a vector in the range of A^h, can be decomposed as a linear combination of the eigenvectors mentioned above.

Turning our attention to the interpolation matrix, T_{2h}^h, which has a reverse mapping from $\mathbb{R}^{N/2-1} \to \mathbb{R}^{N-1}$ and has full rank because there are fewer columns than rows. Like before, we want to ask how this matrix affects the eigenvectors or modes of matrix A^{2h}. Note that this time, we are considering matrix A^{2h} to be consistent with the analysis in restriction. We know that the jth component of the kth eigenvector of A^{2h} is:

$$w_{k,j}^{2h} = \sin\left(\frac{jk\pi}{N/2}\right) \text{ where } 1 \leq k \leq \frac{N}{2} \text{ and } 0 \leq j \leq \frac{N}{2}$$

The jth component of $T_{2h}^{h}\mathbf{w}_k^{2h}$ looks like:

$$T_{2h}^{h}\mathbf{w}_k^{2h} = \frac{1}{2}\begin{bmatrix} 1 & & & \\ 2 & & & \\ 1 & 1 & & \\ & 2 & & \\ & 1 & 1 & \\ & & \ddots & \ddots \end{bmatrix}\mathbf{w}_k^{2h}$$

$$(T_{2h}^{h}\mathbf{w}_k^{2h})_j = \begin{cases} \sin\left(\frac{k(j/2)\pi}{N/2}\right) & j \text{ even} \\ \frac{1}{2}\sin\left(\frac{k\frac{j-1}{2}\pi}{N/2}\right) + \frac{1}{2}\sin\left(\frac{k\frac{j+1}{2}\pi}{N/2}\right) & j \text{ odd} \end{cases}$$

For j odd we have:

$$(T_{2h}^{h}\mathbf{w}_k^{2h})_j = \frac{1}{2}\sin\left(\frac{k(j-1)\pi}{N}\right) + \frac{1}{2}\sin\left(\frac{k(j+1)\pi}{N}\right)$$

Using the trigonemtric sum identity and double angle cosine identity $2\cos^2\theta - 1 = \cos 2\theta$, we get that for the j is odd case:

$$\Rightarrow \sin\left(\frac{kj\pi}{N}\right)\cos\left(\frac{k\pi}{N}\right) = \sin\left(\frac{kj\pi}{N}\right)\left(2\cos^2\left(\frac{k\pi}{2N}\right) - 1\right)$$

$$\Rightarrow \sin\left(\frac{kj\pi}{N}\right)\cos^2\left(\frac{k\pi}{2N}\right) + \sin\left(\frac{kj\pi}{N}\right)\left(\cos^2\left(\frac{k\pi}{2N}\right) - 1\right) \text{ and we have } k = N - k'$$

$$\Rightarrow \sin\left(\frac{kj\pi}{N}\right)\cos^2\left(\frac{k\pi}{2N}\right) - \sin\left(\frac{(N-k')j\pi}{N}\right)\sin^2\left(\frac{k\pi}{2N}\right)$$

$$\Rightarrow \sin\left(\frac{kj\pi}{N}\right)\cos^2\left(\frac{k\pi}{2N}\right) - \sin^2\left(\frac{k\pi}{2N}\right)\left[\sin(j\pi)\cos\left(\frac{k'j\pi}{N}\right) - \cos(j\pi)\sin\left(\frac{k'j\pi}{N}\right)\right]$$

$$\Rightarrow \sin\left(\frac{kj\pi}{N}\right)\cos^2\left(\frac{k\pi}{2N}\right) - \sin^2\left(\frac{k\pi}{2N}\right)\left[(-1)^{j+1}\sin\left(\frac{k'j\pi}{N}\right)\right] \text{ and because j is odd ...}$$

$$\cos^2\left(\frac{k\pi}{2N}\right)\sin\left(\frac{kj\pi}{N}\right) - \sin^2\left(\frac{k\pi}{2N}\right)\sin\left(\frac{k'j\pi}{N}\right) \tag{10}$$

For the even case, we can do the following:

$$(T_{2h}^{h}\mathbf{w}_k^{2h})_j = \sin\left(\frac{kj\pi}{N}\right) = \cos^2\left(\frac{k\pi}{2N}\right)\sin\left(\frac{kj\pi}{N}\right) + \sin^2\left(\frac{k\pi}{2N}\right)\sin\left(\frac{kj\pi}{N}\right)$$

In the previous page we showed that complementary nodes are related by just a sign. Using this we can turn the above expression to take on a similar form. Just like Briggs, we'll define $c_k = \cos^2\left(\frac{k\pi}{2N}\right)$ and $s_k = \sin^2\left(\frac{k\pi}{2N}\right)$. Therefore, for both the even and odd case we can express the new eigenvectors as:

$$[T_{2h}^h \mathbf{w}_k^{2h}]_j = \cos^2\left(\frac{k\pi}{2N}\right)\sin\left(\frac{kj\pi}{N}\right) - \sin^2\left(\frac{k\pi}{2N}\right)\sin\left(\frac{kj\pi}{N}\right) \Rightarrow T_{2h}^h \mathbf{w}_k^{2h} = c_k \mathbf{w}_k^h - s_k \mathbf{w}_k^h \tag{11}$$

The takeaway of all this tedious, trigonemtric algebra is that the interpolation matrix in some way excistes the kth mode in Ω^{2h} to produce two modes on the finer mesh Ω^h.

A basis for the range of interpolation is of course given by the columns of T_{2h}^h because it is full rank (all of its columns are linearly independent). Just like the null space of the restriction matrix, these basis vectors can be decomposed as a linear combination of the modes of A^h and hence reveal that the column space of the interpolation matrix contains both smooth and oscillatory nodes of A^h (see Briggs).

The purpose of all this analysis on these two matrices subspaces and their affect on the modes of A^h is because the crux of multigrid methods is the *coarse grid correction scheme*. This scheme is a residual correction scheme that maps the residual of n iterative method on some mesh Ω^h to a coarser mesh to solve for error. This error is then interpolated upwards to obtain a better approximation. Essentially, coarse grid correction (CGC) is a series of matrix products.

1. We start with an approximation, $\mathbf{v}^h$ to our true solution, $\mathbf{u}^h$ and relax ν times. $\mathbf{v}^h \leftarrow P^\nu \mathbf{v}^h$, where P is a relaxation scheme
2. Obtain a residual $\mathbf{r}^h$ and restrict it: $\mathbf{r}^{2h} = R_h^{2h}(\mathbf{f}^h - A^h\mathbf{v}^h)$
3. Solve the residual equation: $\mathbf{v}^{2h} = (A^{2h})^{-1}\mathbf{f}^{2h}$
4. Correct the approximation on the original mesh Ω^h: $\mathbf{v}^h \leftarrow \mathbf{v}^h + T_{2h}^h \mathbf{v}^{2h}$

These 4 steps can be summarized as a series of matrix products:

$$\mathbf{v}^h \leftarrow P^\nu \mathbf{v}^h + T_{2h}^h (A^{2h})^{-1} R_h^{2h}(\mathbf{f}^h - A^h P^\nu \mathbf{v}^h) \tag{12}$$

Naturally, this must statisfy:

$$\mathbf{u}^h = P^\nu \mathbf{u}^h + T_{2h}^h (A^{2h})^{-1} R_h^{2h}(\mathbf{f}^h - A^h P^\nu \mathbf{u}^h) \tag{13}$$

Taking their difference and considering this difference as the coarse grid correction operator (as Brigsgs does, page 74), we get a matrix operator:

$$\left[I - T_{2h}^h (A^{2h})^{-1} R_h^{2h} A^h\right] P^\nu \tag{14}$$

Considering this whole series of products and sums as one operation, Briggs states that in the scenario in which we do not relax, $\nu = 0$, then the subspaces $W_k^h = \text{span}\{w_k^h, w_{k'}^h\}$ is an invariant subspace.

We will endeavour to prove this. Without any relaxation, our operator is:

$$I - T_{2h}^{h}(A^{2h})^{-1}R_{h}^{2h}A^{h}$$

In a homework problem, we found that kth eigenvalue of the matrix A^h is $\lambda_k^h = 4\sin^2\left(\frac{k\pi}{2N}\right)$. We therefore have:

1. $A^h\mathbf{w}_k^h = \lambda_k^h\mathbf{w}_k^h$
2. $R_j^{2h}A^h\mathbf{w}_k^h = \lambda_k^h\cos^2\left(\frac{k\pi}{2N}\right)\mathbf{w}_k^h$
3. $(A^{2h})^{-1}R_j^{2h}A^h\mathbf{w}_k^h = \lambda_k^h\cos^2\left(\frac{k\pi}{2N}\right)(A^{2h})^{-1}\mathbf{w}_k^{2h} = \frac{\lambda_k^h}{\lambda_k^{2h}}\cos^2\left(\frac{k\pi}{2N}\right)\mathbf{w}_k^{2h}$
4. $T_{2h}^{h}(A^{2h})^{-1}R_j^{2h}A^h\mathbf{w}_k^h = \frac{\lambda_k^h}{\lambda_k^{2h}}\cos^2\left(\frac{k\pi}{2N}\right)T_{2h}^{h}\mathbf{w}_k^{2h} = \frac{\lambda_k^h}{\lambda_k^{2h}}\cos^2\left(\frac{k\pi}{2N}\right)\left[c_k\mathbf{w}_k^h - s_k\mathbf{w}_{k'}^h\right]$
5. Our final expression for the change on $\mathbf{w}_k^h$ by this coarse grid correction operator is:

$$\mathbf{w}_k^h - \frac{\lambda_k^h}{\lambda_k^{2h}}\cos^2\left(\frac{k\pi}{2N}\right)\left[c_k\mathbf{w}_k^h - s_k\mathbf{w}_{k'}^h\right]$$

Plugging in our eigenvalues into this, we obtain:

$$\mathbf{w}_k^h - \frac{4\sin^2\left(\frac{k\pi}{2N}\right)}{4\sin^2\left(\frac{k\pi}{N}\right)}\cos^2\left(\frac{k\pi}{2N}\right)\left[c_k\mathbf{w}_k^h - s_k\mathbf{w}_{k'}^h\right] \tag{15}$$

$$\left(1 - \frac{\sin^2\left(\frac{k\pi}{2N}\right)\cos^4\left(\frac{k\pi}{2N}\right)}{\sin^2\left(\frac{k\pi}{N}\right)}\right)\mathbf{w}_k^h + \frac{\sin^4\left(\frac{k\pi}{2N}\right)\cos^2\left(\frac{k\pi}{2N}\right)}{\sin^2\left(\frac{k\pi}{N}\right)}\mathbf{w}_{k'}^h \tag{16}$$

Using the double angle formula $\sin 2\theta = 2\sin\theta\cos\theta$ and $\cos^2\theta + \sin^2\theta = 1$, we can show (a lot of algebra):

$$\left(1 - \frac{\cos^2\left(\frac{k\pi}{2N}\right)}{4}\right)\mathbf{w}_k^h + \frac{1}{4}\sin^2\left(\frac{k\pi}{2N}\right)\mathbf{w}_{k'}^h$$

$$\Rightarrow \left(\frac{3}{4} + \frac{1}{4}\sin^2\left(\frac{k\pi}{2N}\right)\right)\mathbf{w}_k^h + \frac{1}{4}\sin^2\left(\frac{k\pi}{2N}\right)\mathbf{w}_{k'}^h$$

If we continue and expand things out, we can show this equals to, $s_k\mathbf{w}_k + s_k w_{k'}$ and hence we obtain invariance of the elemtn $\mathbf{w}_k$. We need to do this for the element $\mathbf{w}_{k'}$. To avoid the same type of analysis, we'll rely on Briggs to assert that we get $c_k\mathbf{w}_k + c_k\mathbf{w}_{k'}$. What this means is that we get smooth and oscillatory results when this coarse grid correction scheme is applied to these complementary modes.

The power of this method comes from the spectral properties of iterative solvers. When we introduce ν steps of relaxation of some iterative method, like Gauss-Seidel or SOR, then we get that our new operator becomes:

$$\left(I - T_{2h}^{h}(A^{2h})^{-1}R_h^{2h}A^h\right)P^{\nu} \tag{17}$$

Note, Following the same 5 steps above, we get that (letting λ_k be the eigenvalue of P associated with the kth eigenvector $\mathbf{w}_k$)

$$\left(I - T_{2h}^{h}(A^{2h})^{-1}R_{h}^{2h}A^{h}\right)P^{\nu}\mathbf{w}_k^h = \left(I - T_{2h}^{h}(A^{2h})^{-1}R_{h}^{2h}A^{h}\right)\lambda_k^{\nu}\mathbf{w}_k^h$$

What is new this time is a constant term - the eigenvalue! So we get similar results! The coarse grid correction scheme when operating on $\mathbf{w}_k$ and $\mathbf{w}_{k'}$ returns:

$$\lambda_k^{\nu}s_k\mathbf{w}_k + \lambda_k^{\nu}s_k\mathbf{w}_{k'} \text{ and } \lambda_k^{\nu}c_k\mathbf{w}_k + \lambda_k^{\nu}c_k\mathbf{w}_{k'} \quad \text{where } 1 \le k \le \frac{N}{2}, \quad k' = N - k$$

The smoothing is the dominant term here! It is what makes the oscillatory modes small, while at the same time eliminating smooth modes, reflected in the s_k terms. When all terms are small particularyly for small modes k relative to $N/2$ or when the number of relaxation increases, the end result is a complete process where both types of eigenvectors (low frequency and high frequency modes) are well damped. This is the spectral picture of the multigrid method

3 One Dimensional Problem

For the one dimenssional problem we chose to demonstrate the Poisson problem with Dirichlet boundary conditions as follows:

$$\begin{aligned} &\frac{d^2u}{dx^2} = f(x) \ , \quad x \in (a,b), \\ &u(a) = \alpha, \\ &u(b) = \beta. \end{aligned} \tag{18}$$

The values of $f(x)$, a, b, α, and β are specified below for the test example.

- Example #1:

$$f(x) = -\sin(x) \ , \quad a = 0 \ , \quad b = 2\pi \ , \quad \alpha = \beta = 0$$

The results of our method are presented in the Results section, however we will describe the implementation in the next section first. The analytic solution to this ODE is not a difficult one to derive, and we will not show the derivation here, but we state that the solution is:

$$u(x) = \sin(x)$$

3.1 Implementation

Regarding implementation, choosing an appropriate data structure is essential to achieving a high level efficency otherwise, the benefits of the multigrid method is lost. Briggs, in *A Multigrid Tutorial*, adopts arrays as his primary data structure for one-dimensional problems. We adopted the same strategy and in general, any arbitrary grid will contain $2^n - 1$ interior grid points plus two boundary points for a total of $2^n + 1$. The size of the arrays are transformed across different mesh grids.

In a dedicated python class, we hold onto several attributes of the problem. The constructor for the one dimenssional problem, takes as input: The descrete domain x (a Numpy array), solution array u whose input serves as the initial guess, boundary endpoints, and the non-homogeneous term f as a function. Holding onto these variable we can then call `iterate(), v_sched(), or fmg()` to approximate the solution.

`iterate()` calls the iterative solver. The numerical method empolyed is either Gauss-Seidel or Successive Over Relaxation. The pseudocode as well as mathematical analysis of these methods was explained in previous assignments for this course, so we chose to leave that out of this paper.

`v_sched()` intiates a Multi Grid V-Cycle which solves a system $Au = f$ with a coarse grid correction scheme. The pseudocode for the recursive implementation used is as follows:

```
v_sched(u, A, rhs, dx, level):
    if this is the coarsest grid:
        solve exactly Au = rhs
    else:
        iterate(u) twice
        residue = Au - rhs
        coarsen(A, residue)
        e = v_sched(0, coarse A, coarse residue, 2*dx, level-1)
        interpolate(e) and update u by added the refined e
        iterate(new u) twice
    return u
```

The recursive call to `v_sched()` is solving a new linear system on the coarser grid. This coarsening is moves through the levels until the problem at hand is small enough to be solved exaclty, in our case using Numpy's linear solver. For the iterative methods called before and after the coarse grid correction, either Gauss-Seidel or SOR could be chosen, and the number of times of pre and post smoothing are a parameter that can be set.

`fmg()` intiates a Full Multi Grid Cycle which solves a system $Au = f$ with a coarse grid correction scheme. The pseudocode for the recursive implementation used is as follows:

```
fmg(u, A, rhs, dx, level):
    if this is the coarsest grid:
        solve exactly Au = rhs
    else:
        coarsen(A, u, rhs)
        u' = fmg(coarse u, coarse A, coarse rhs, 2*dx, level-1)
        u_new = interpolate(u')
        u = v_sched(u_new, A, rhs, dx, level)
    return u
```

The recursive call to `fmg()` is solving a coarser version of the same linear system, $Au = f$. At the coarses level it solves this exaclty and then interoplates the solution back up to the finer grid. After each coarsening/refining stage the fucntion calls `v_sched()` solving the finer problem using the coarse grid correction scheme outlined above in pseudocode.

It is important to note here that the pseudocode would look identical for the two dimenssional problem, becasue the differences are in the details. For two dimensions, there is a fair amount of reshaping vectors and matrices, and the coarsening and interpolating matrices are also slightly different, but because these are only outlined in previous sections and not highlited in the pseudocode, we will not rewrite any pseudocode for the two dimensional problem.

3.2 Results

From Example #1 we have solution plots and an error comparison from four runs. Each took the same initialization parameters, other than the choice of iterative solver. (a) GS was computed using only the Gauss-Seidel iteration method; (b) SOR was only using the SOR method; (c) V-Grid was a coarse grid correction cycle method using `v_sched()`, above; and (d) FMG was using the Full Multi Grid Cycle outlined above in `fmg()`. Below are the solutions and error plots:

4 Two Dimensional Problem

For the two dimenssional problem we chose to demonstrate the Poisson problem with Dirichlet boundary conditions as follows:

$$\begin{aligned} u_{xx} + u_{yy} &= f(x,y) \ , \quad (x,y) \in \Omega, \\ u|_{\partial\Omega} &= g(x,y), \end{aligned} \tag{19}$$

The values of $f(x,y)$, Ω, and $g(x,y)$ are specified below for the test example.

- Example #1:

$$f(x) = -\sin(x) \ , \quad a = 0 \ , \quad b = 2\pi \ , \quad \alpha = \beta = 0$$

The results of our method are presented in the Results section, however we will describe the implementation in the next section first. The analytic solution to this ODE is not a difficult one to derive, and we will not show the derivation here, but we state that the solution is:

$$u(x) = \sin(x)$$

4.1 Results

5 Error Analysis

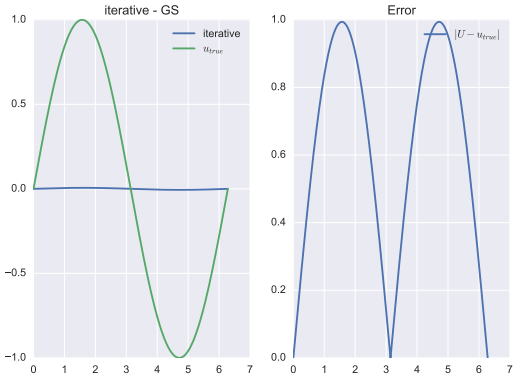
5.1 Multigrid

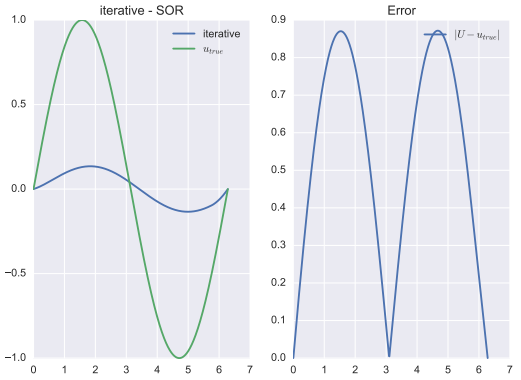
5.2 Gauss-Seidel

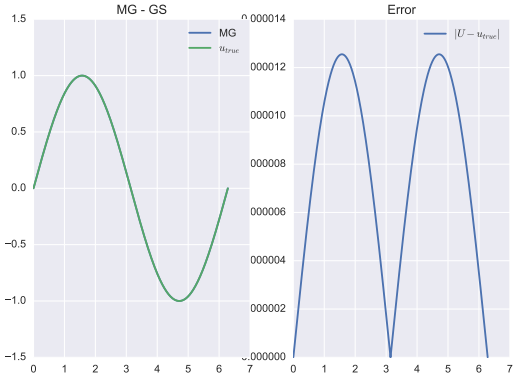
5.3 Successive Over Relaxation

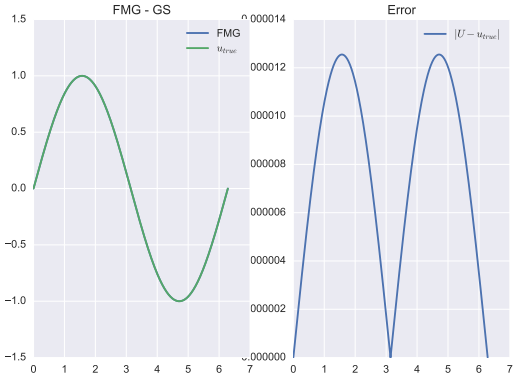
6 Conclusion

7 Other


iterative - GS
iterative
u_{true}
Error
$|U - u_{true}|$


iterative - SOR
iterative
u_{true}
Error
$|U - u_{true}|$


MG - GS
MG
u_{true}
Error
$|U - u_{true}|$


FMG - GS
FMG
u_{true}
Error
$|U - u_{true}|$