Brad Martin

March 11, 2013

Programming Assignment 4 Report (Multigrid Methods – APPM 6640)

Introduction

This programming assignment required us to code up and test out V-cycling (VC) and full multigrid (FMG) methods to find an approximate discrete solution to the Poisson/Helmholtz equation presented as equation 1.4 in *A Multigrid Tutorial*:

$$-u_{xx} - u_{yy} + \sigma u = f(x, y)$$

We're solving this within a closed square domain in 2 dimensions. As we learned in lecture, we found out through experimentation that both VC and FMG methods are powerful ways to solve this problem numerically, with FMG edging out VC in terms of efficiency.

Further, we found that using the variational Laplacian stencil:

$$\begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & -8/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix} /h^2$$

improved the convergence of our V-cycles considerably.

Approach

We chose our test domain as the closed unit square [0,1] x [0,1], and our test problem as:

$$-u_{xx} - u_{yy} + u = \sin(\pi x)\sin(\pi y)$$

with the exact (analytical) solution:

$$u(x, y) = \frac{\sin(\pi x)\sin(\pi y)}{1 + 2\pi^2}$$

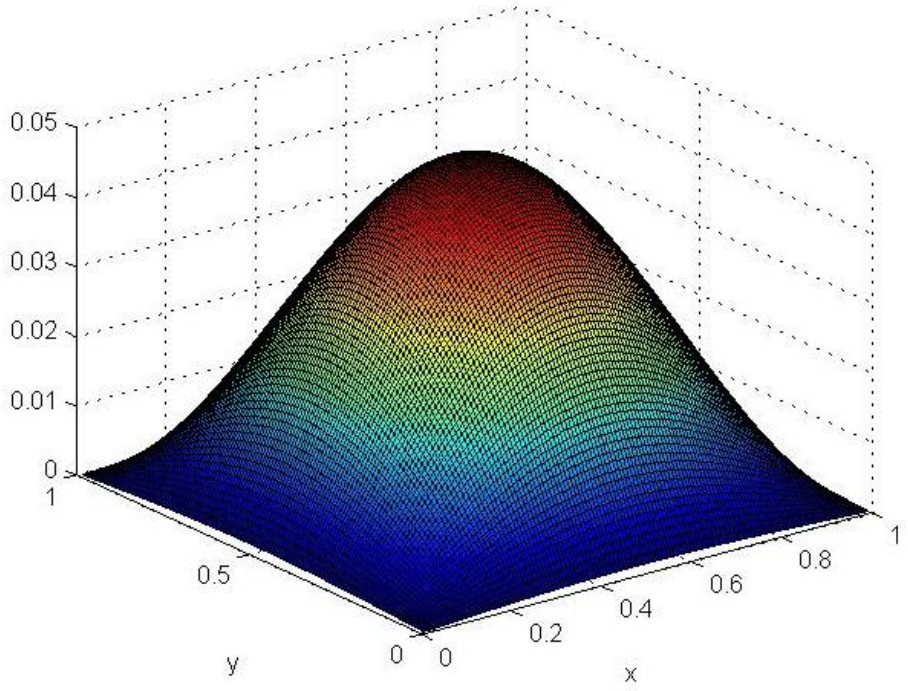

Figure 1. Analytical solution to our test problem: $\sin(\pi x)\sin(\pi y)/(1+2\pi^2)$

We took the following steps in our analysis:

1) We set up multigrid structures in C++ using linear interpolation stencils for prolongation (I_{2h}^{h}), full weighting stencils for restriction (I_{h}^{2h}), and the standard 5-point, 2nd order, 2D Laplacian operator (A^h) on each grid level:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} / h^2$$

2) Using a zero initial guess, we carried out (2,2) V-cycling on 2D fine grids containing n = 64, 128, and 256 points (in one direction), and we tabulated $\| r \|_h$ and $\| e \|_h$ after each cycle. This created data similar to Table 4.1 on page 65 of *A Multigrid Tutorial*, and gave us a good indication of how our V-cycles were performing.
3) We implemented several FMG schemes on these same grids, and compared performance and efficiency of the FMG schemes with the V-cycling efforts above.
4) We wrote a routine to create the variational 2D Laplacian operator shown on page 1, and performed (2,2) V-cycles on the test problem with n = 128 fine grid points. We compared these variational-operator results with those obtained in part 2 above.

Theory

There's a whole lot we could talk about in the theory section for this programming assignment, but we'll try to focus on one area relevant to the comparisons we're doing here between variational and non-variational operators.

Jumping to page 85 of *A Multigrid Tutorial*, we're considering a 2-level coarse-grid correction scheme (CG) between grid h (fine) and grid 2h (coarse). Our 2D prolongation (linear interpolation) and restriction (full weighting) operators satisfy the Galerkin condition. From this and the fundamental theorem of linear algebra, we've established that the space of data and error vectors Ω^h (in $\mathbb{R}^n$) we have on grid h can be orthogonally decomposed into a direct sum of 1) the range of prolongation with 2) the null space of restriction acting on A^h:

$$\Omega^h = \mathcal{R}\left(I_{2h}^h\right) \oplus N(I_h^{2h} A^h)$$

We can thus express any error vector e^h on grid h as a sum of vectors in these two subspaces:

$$e^h = s^h \oplus t^h, \qquad s^h \in \mathcal{R}\left(I_{2h}^h\right), \qquad t^h \in N(I_h^{2h} A^h)$$

Assuming CG acts on some unknown error vector e^h on grid h, and assuming we solve for e^{2h} (e^h "interpreted" on grid 2h) exactly, we can express CG like so:

$$CG = I - I_{2h}^h (A^{2h})^{-1} I_h^{2h} A^h$$

We first look at what happens to the t^h component of e^h when acted upon by CG:

$$CG(t^h) = t^h - I_{2h}^h (A^{2h})^{-1} I_h^{2h} A^h (t^h) = t^h - I_{2h}^h (A^{2h})^{-1} \left(I_h^{2h} A^h (t^h)\right) = t^h - I_{2h}^h (A^{2h})^{-1} (0) = t^h$$

This identity occurs because t^h, as we defined it above, is in the null space of $I_h^{2h}A^h$. *[Note: this is true regardless of the variational relationships of I_{2h}^h and I_h^{2h} with our operators A^h and A^{2h}.]* But the theory of how CG acts on s^h is *much* more reliant on variational relationships:

$$CG(s^h) = s^h - I_{2h}^h(A^{2h})^{-1}I_h^{2h}A^h(s^h)$$

Now, since $s^h \in \mathcal{R}(I_{2h}^h)$, s^h can be written as $I_{2h}^h q^{2h}$ for some q^{2h} located on grid 2h. The following follows:

$$CG(s^h) = s^h - I_{2h}^h(A^{2h})^{-1}I_h^{2h}A^h I_{2h}^h q^{2h}$$

Here's the important part. If A^{2h} *does* satisfy a variational relationship with A^h ($A^{2h} = I_h^{2h}A^h I_{2h}^h$), then the above reduces to:

$$CG(s^h) = s^h - I_{2h}^h(A^{2h})^{-1}A^{2h}q^{2h} = s^h - I_{2h}^h q^{2h} = s^h - s^h = 0$$

If A^{2h} and A^h aren't related in this way, though, we're much more uncertain about how CG acts on s^h.

In addition to the orthogonal decomposition presented above, we also have another orthogonal decomposition from the symmetry of A^h that we've explored in previous homework assignments and projects:

$$\Omega^h = L \oplus H$$

where L are the "low" frequency components of data or error, and H are the "high" frequency components of the same. We know that relaxation helps to diminish the high frequency components of error significantly.

And now we've established that an accurate CG routine nullifies s^h (vectors in $\mathcal{R}\left(I_{2h}^h\right)$) for *variational* A^h/A^{2h} pairs:

$$\Omega^h = L \oplus H \quad \rightarrow \; stuff \; in \; H \; damped \; by \; relaxation$$

$$\Omega^h = \mathcal{R}\left(I_{2h}^h\right) \oplus N(I_h^{2h} A^h) \quad \rightarrow \; stuff \; in \; \mathcal{R}\left(I_{2h}^h\right) \; damped \; by \; CG$$

Even for pairs of variational operators, though, we're relying on the assumption that H is "like" $N(I_h^{2h} A^h)$ and that L is "like" $\mathcal{R}\left(I_{2h}^h\right)$ for assurance that multigrid methods work well. Fortunately this seems to be the case quite often – further details, discussion, and illustrations abound in Chapter 5 of *A Multigrid Tutorial*.

The bottom line for our experiments here is that we're not only relying on the assumptions above when we use non-variational operators (i.e. when we recreate A^h with the traditional 2nd order, 5-point stencil for the 2D Laplacian approximation). In some sense, we're also relying on the assumption that A^{2h} is in some sense "close" in operation to $I_h^{2h} A^h I_{2h}^h$. Maybe if we don't make this second assumption – and we instead use a variational operator – we'll see better convergence in our routines.

Numerics and Results

The spread of **Figure 2** (over the next several pages) displays the evolution of residual and error over several (2,2) V-cycles on our trial problem (with n = 128 as our fine grid). *[Note: errors here and for the rest of the report are* total *errors: the difference between the analytical solution and the solution guess at each grid point.]*

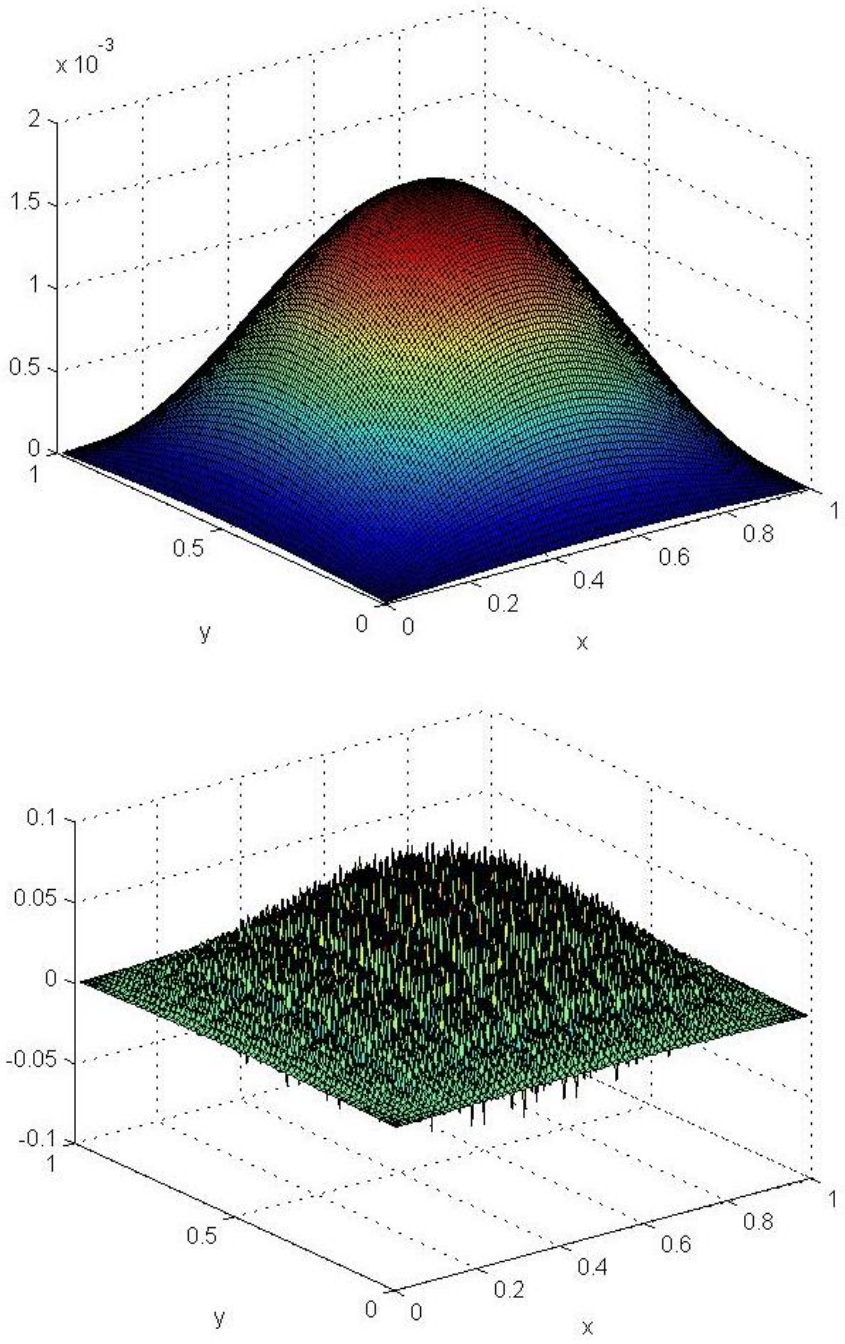

Figure 2a. Total error (above) and residual (below) for the trial problem, after 1 (2,2) V-cycle, with n = 128.

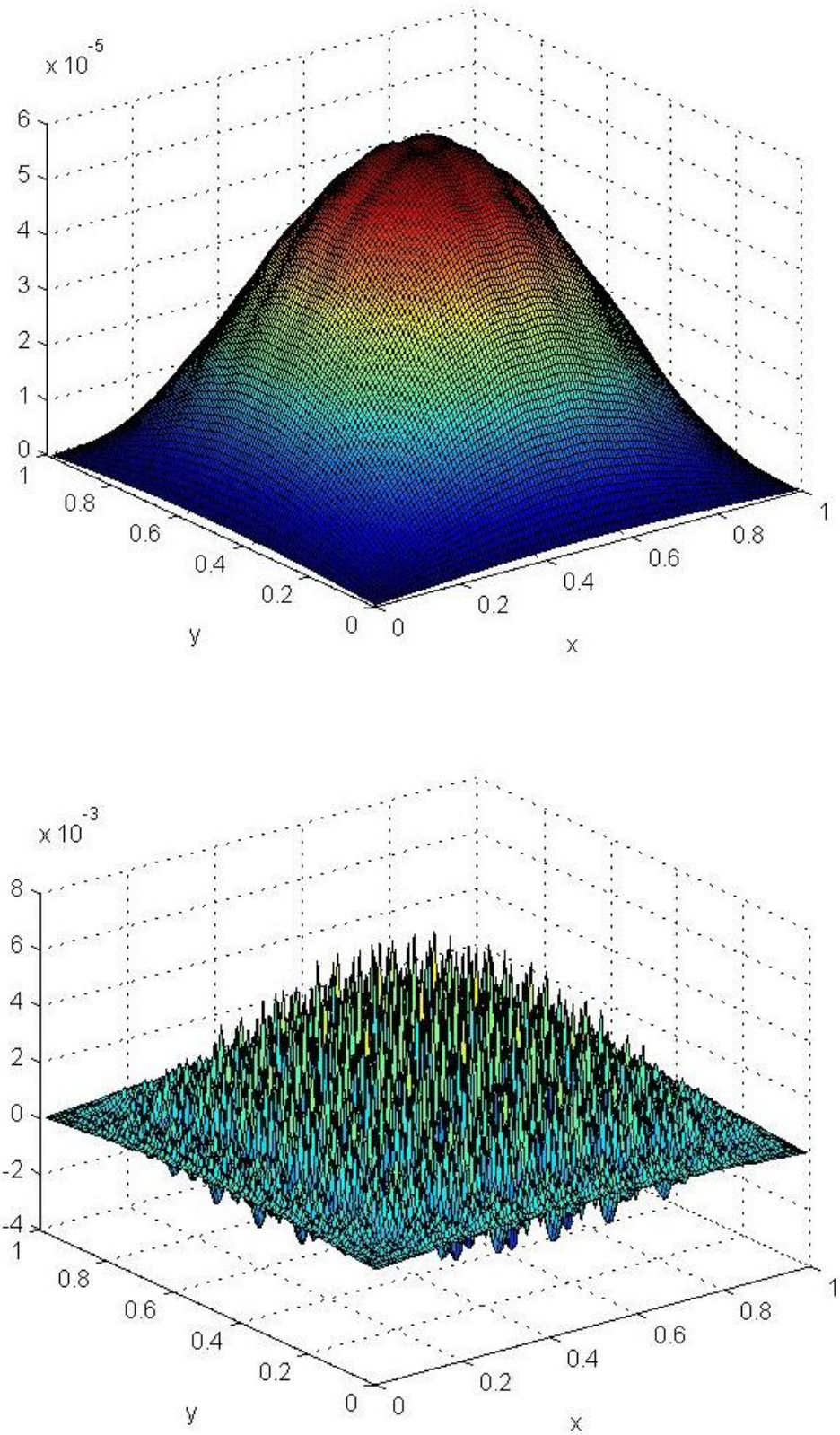

Figure 2b. Total error (above) and residual (below) for the trial problem, after 2 (2,2) V-cycles, with n = 128.

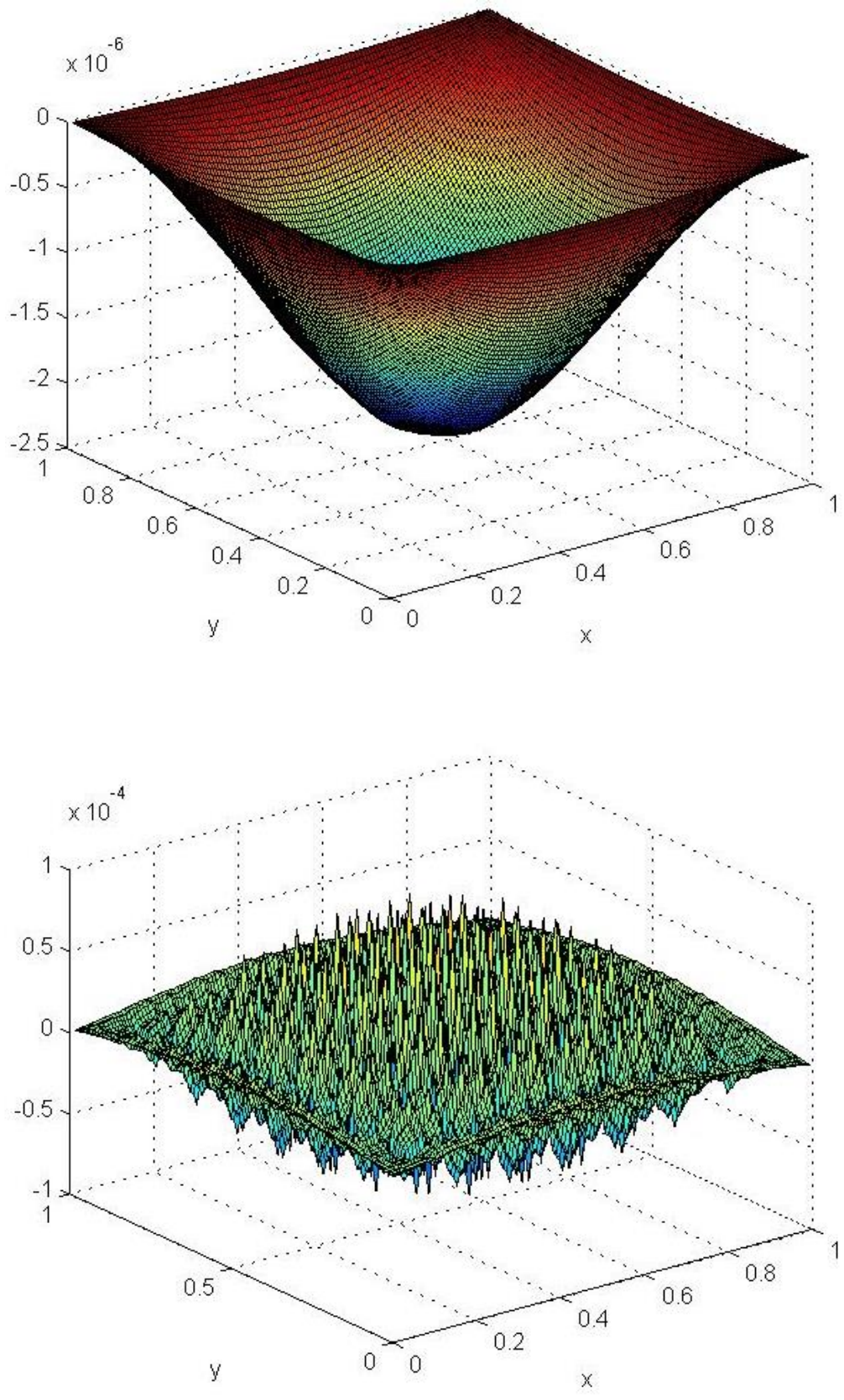

Figure 2c. Total error (above) and residual (below) for the trial problem, after 4 (2,2) V-cycles, with n = 128.

After 4 V-cycles, solution error on the n = 128 grid had stabilized. This can be seen in **Table 1**, where we present the results of our first objective: (2,2) V-cycling on fine grids of n = 64, n = 128, and n = 256 points.

Table 1. Discrete L^2 norms of residuals and total errors (and cycle-to-cycle ratios) after 1 to 10 V-cycles on n = 64, n = 128, and n = 256 (in solving the test problem presented on page 1)

	n = 64				n = 128			
V-cycle	$\|\| r \|\|_h$	ratio	$\|\| e \|\|_h$	ratio	$\|\| r \|\|_h$	ratio	$\|\| e \|\|_h$	ratio
1	2.34e-02		8.48e-04		2.41e-02		8.62e-04	
2	1.10e-03	0.05	2.56e-05	0.03	1.33e-03	0.06	2.97e-05	0.03
3	9.26e-05	0.08	3.54e-06	0.14	1.32e-04	0.10	5.84e-08	0.0020
4	1.05e-05	0.11	4.57e-06	1.29	1.61e-05	0.12	1.11e-06	19.0
5	1.35e-06	0.13	4.61e-06	1.01	2.15e-06	0.13	1.15e-06	1.04
6	1.91e-07	0.14	4.61e-06	1.00	3.10e-07	0.14	1.15e-06	1.00
7	2.89e-08	0.15	4.61e-06	1.00	4.71e-08	0.15	1.15e-06	1.00
8	4.52e-09	0.16	4.61e-06	1.00	7.41e-09	0.16	1.15e-06	1.00
9	7.16e-10	0.16	4.61e-06	1.00	1.19e-09	0.16	1.15e-06	1.00
10	1.14e-10	0.16	4.61e-06	1.00	1.94e-10	0.16	1.15e-06	1.00

	n = 256			
V-cycle	$\|\| r \|\|_h$	ratio	$\|\| e \|\|_h$	ratio
1	2.46e-02		8.65e-04	
2	1.52e-03	0.06	3.08e-05	0.04
3	1.65e-04	0.11	8.32e-07	0.03
4	2.09e-05	0.13	2.47e-07	0.30
5	2.86e-06	0.15	2.88e-07	1.16
6	4.20e-07	0.15	2.88e-07	1.01
7	6.49e-08	0.16	2.88e-07	1.00
8	1.04e-08	0.16	2.88e-07	1.00
9	1.69e-09	0.17	2.88e-07	1.00
10	2.79e-10	0.17	2.88e-07	1.00

While the convergence rates here (~ 0.16) aren't as impressive as those presented for red/black Gauss-Seidel (2,1) V-cycling on page 65 of *A Multigrid Tutorial* (~ 0.05-0.07), we realize that a variety of factors could be at play here, including the method of relaxation. We'll examine the effects of using a variational Laplacian operator later.

After seeing that our V-cycles worked pretty well, we implemented several FMG schemes to solve the same test problem. **Table 2** presents these results.

Table 2. FMG results

FMG type	n = 64		n = 128		n = 256	
	$\| r \|_h$	$\|e\|_h$	$\| r \|_h$	$\| e \|_h$	$\| r \|_h$	$\| e \|_h$
(1,1)	1.35e-02	9.90e-07	1.35e-02	2.41e-07	1.35e-02	6.02e-08
(2,1)	2.75e-03	3.35e-06	2.71e-03	8.34e-07	2.70e-03	2.08e-07
(2,2)	5.51e-04	3.37e-06	5.41e-04	8.38e-07	5.39e-04	2.09e-07
(3,2)	1.20e-04	3.88e-06	1.10e-04	9.66e-07	1.08e-04	2.41e-07
(3,3)	3.08e-05	3.89e-06	2.26e-05	9.69e-07	2.17e-06	2.42e-07

Table 2 shows the power of the FMG algorithm. For example, if we observe the (2,2) FMG row, we see that for less than the computational cost of 2 V-cycles on the fine grid, we've stabilized our solution guess near discretization error. Looking at **Table 1**, V-cycling alone on a zero initial guess took around 4 iterations to achieve this.

Other grid sizes and trial problems might highlight the advantage of FMG better than the example tested here; still, this is a good indication that our FMG routine is working well.

As discussed in the Theory section, we were interested in seeing how V-cycling with the variational Laplacian operator

$$\begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & -8/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix} /h^2$$

performed when compared with the 5-point-stencil Laplacian operator. Results for (2,2) variational V-cycling on n = 128 are presented in **Table 3.**

Table 3. Discrete L^2 norms (and cycle-to-cycle ratios) of residuals and total errors after 1 to 10 variational V-cycles on a fine grid of n = 128 (in solving the test problem presented on page 1)

n = 128				
V-cycle	$\| r \|_h$	ratio	$\| e \|_h$	ratio
1	1.53e-02		3.15e-04	
2	3.17e-04	0.02	7.60e-07	0.03
3	1.87e-05	0.06	3.40e-06	0.14
4	1.49e-06	0.08	3.46e-06	1.29
5	1.23e-07	0.08	3.46e-06	1.01
6	1.03e-08	0.08	3.46e-06	1.00
7	8.83e-10	0.09	3.46e-06	1.00
8	7.62e-11	0.09	3.46e-06	1.00
9	6.63e-12	0.09	3.46e-06	1.00
10	6.01e-13	0.09	3.46e-06	1.00

It looks like our intuition from the Theory section was correct – using the variational Laplacian operator at each grid level helped improve our convergence factor from around 0.16 to about 0.08.

Conclusions:

It was gratifying to see that our V-cycling and FMG schemes indeed worked (and worked pretty well!), and that the "good" feeling we had about the variational operator was confirmed through some numerical trials. But we only scratched the surface here, even if we're restricting our study to linear problems on closed squares. How would these schemes work on trial problems featuring more extreme behavior somewhere inside the domain? How would different relaxation schemes affect the convergence factors for V-cycling and the relative effectiveness of FMG vs. V-cycling alone?

And in these other scenarios, how would the results using a variational differential operator compare with those using operators designed independently of variational constraints on each grid level? How would the absence of a Galerkin condition between restriction and prolongation operators further affect convergence rates? These two questions in particular will probably come up when carrying out my course project – when using radial basis functions, it could be quite difficult to enforce these helpful conditions between operators in a multigrid structure.