MECH511 Programming Assignment 1
Multigrid Methods

Nick Earle

March 11, 2019

1 Transfer Operator Validation

One of the first necessities in any multigrid code is having good transfer operators. For this assignment an averaged fine-to-coarse transfer operator was used along with injection and interpolation type coarse-to-fine transfer operators all using a 2:1 cell ratio, that is $H = 2h$. They are defined as follows:

Fine-to-Coarse:

$$I_{h\to H}: \qquad e^H_{i,j} = \frac{e^h_{2i-1,2j} + e^h_{2i-1,2j-1} + e^h_{2i,2j} + e^h_{2i,2j-1}}{4} \tag{1}$$

Coarse-to-Fine by Injection:

$$I_{H\to h}: \qquad e^h_{2i-1,2j} = e^h_{2i-1,2j-1} = e^h_{2i,2j} = e^h_{2i,2j-1} = e^H_{i,j} \tag{2}$$

Coarse-to-Fine by Interpolation:

$$I_{H\to h}: \begin{cases} e^h_{2i-1,2j} & = \frac{9}{16}e^H_{i,j} + \frac{3}{16}e^H_{i-1,j} + \frac{3}{16}e^H_{i,j+1} + \frac{1}{16}e^H_{i-1,j+1} \\ e^h_{2i-1,2j-1} & = \frac{9}{16}e^H_{i,j} + \frac{3}{16}e^H_{i-1,j} + \frac{3}{16}e^H_{i,j-1} + \frac{1}{16}e^H_{i-1,j-1} \\ e^h_{2i,2j} & = \frac{9}{16}e^H_{i,j} + \frac{3}{16}e^H_{i+1,j} + \frac{3}{16}e^H_{i,j+1} + \frac{1}{16}e^H_{i+1,j+1} \\ e^h_{2i,2j-1} & = \frac{9}{16}e^H_{i,j} + \frac{3}{16}e^H_{i+1,j} + \frac{3}{16}e^H_{i,j-1} + \frac{1}{16}e^H_{i+1,j-1} \end{cases} \tag{3}$$

Testing of each of these transfer operators was done by initializing the domain with the function $sin(\pi x)sin(\pi y)$ and transferring from fine to coarse and back again. Figure 1, below, show the results of the transferred function and Figure 2 shows the error in each transfer.

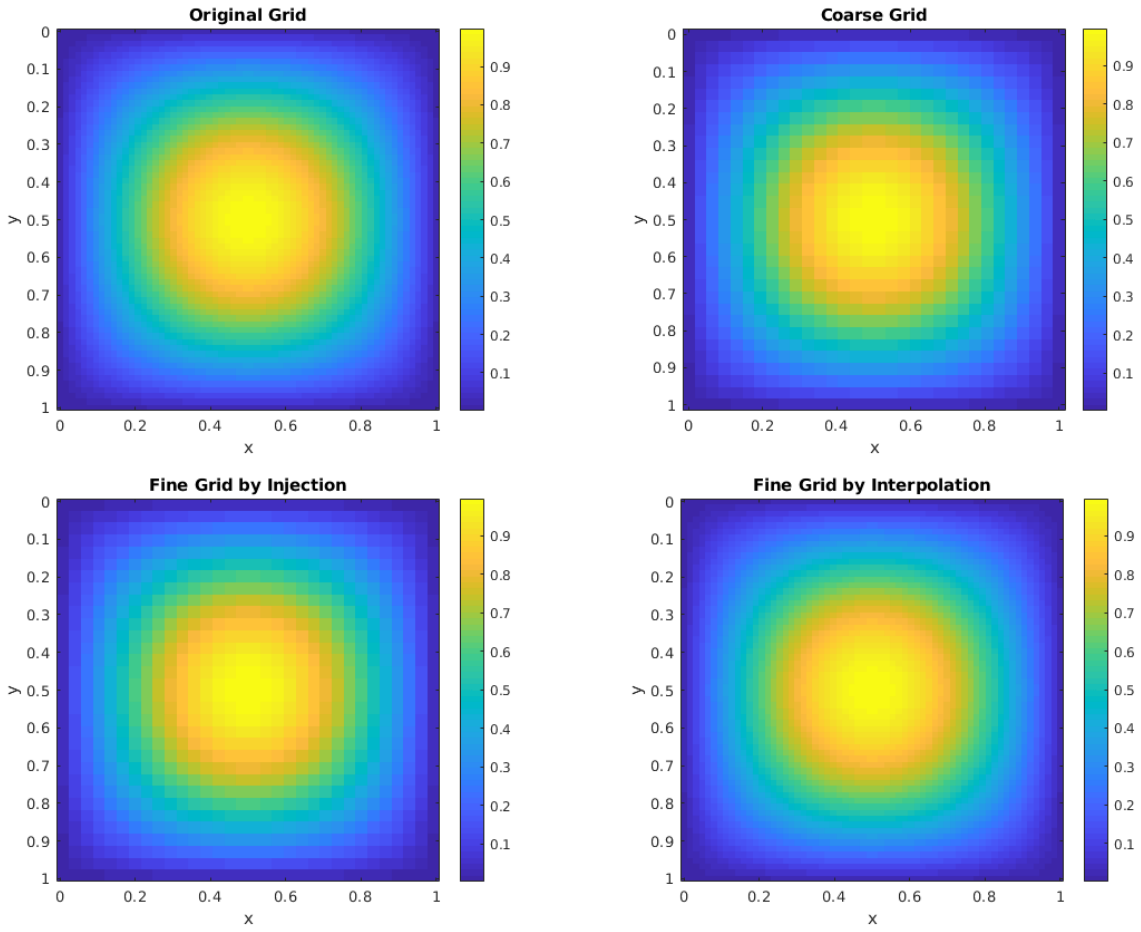

Figure 1: Original and transferred grids

As expected, the coarse grid is simply a coarser version of the original; the injected grid is qualitatively the same as the coarse grid, but with four cells for each on the coarse; and the interpolated grid is very similar to the original grid, as each of the cells is a weighted average of its neighbours. Looking at the error images of Figure 2, the coarse and interpolated grid errors are very much associated with the function values, as expected. However, the injected error is a little crazier as three cells out of every block of four cells have the value of that neighbour, so there are jumps from each group to the next. The L_2 and L_∞-norms of each grid are tabulated below.

Table 1: Grid Error

Grid	$L_2 - norm$	$L_\infty - norm$
Coarse	3.0e−4	6.0e−4
Injection	0.0172	0.0248
Interpolation	0.0012	0.0024

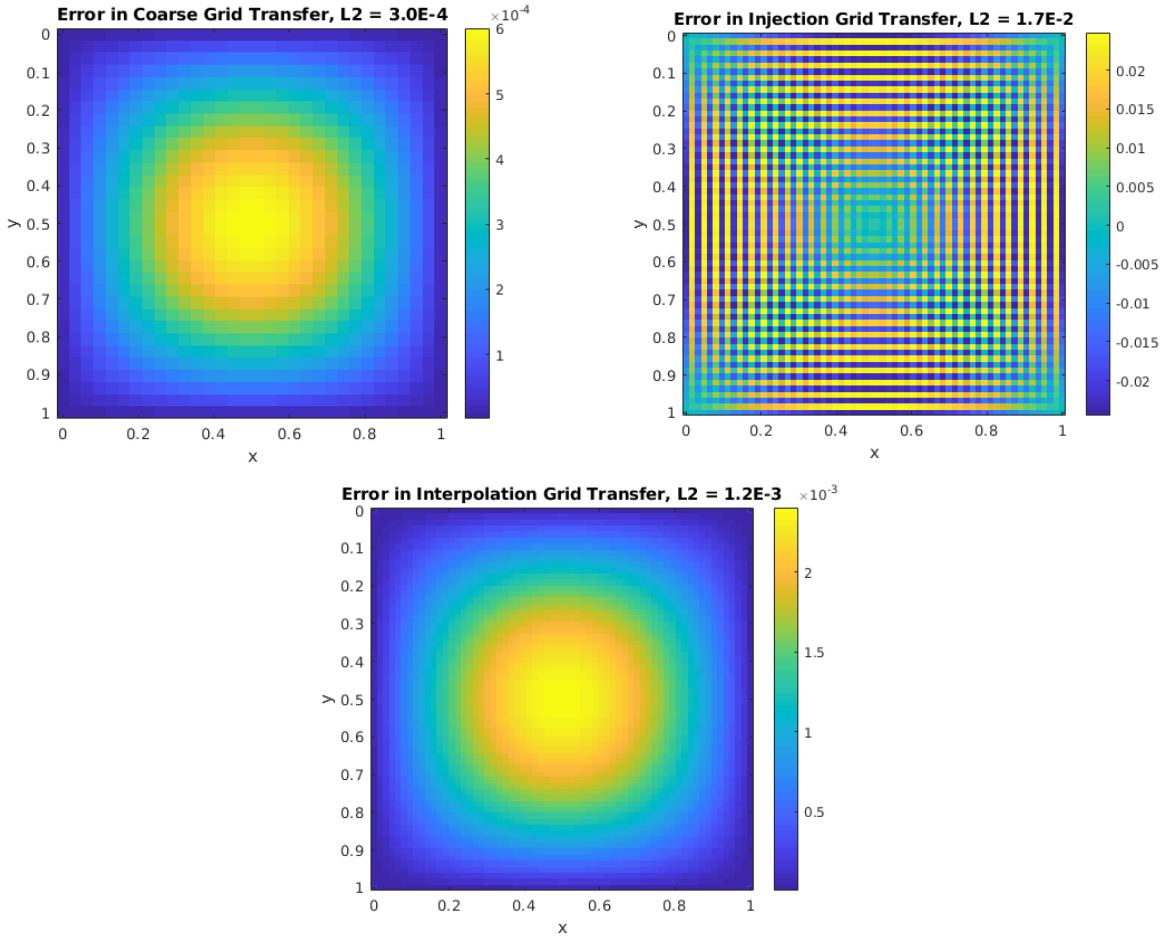

Figure 2: Errors in the grid transfers

2 Source Term Validation

To verify the source term a two level scheme was implemented with a single pass on the fine mesh, followed by 20 passes on the coarse mesh, and then another pass on the fine mesh. Starting again with initial data of $sin(\pi x)sin(\pi y)$ and zero boundary conditions the code was run until a L_2-norm of 10^{-9} was reached, using the theoretically ideal over-relaxation constant, $\omega = 2/3$. Using an interpolation fine-to-coarse transfer operator the scheme converged in 204 iterations, while it took 206 iterations using the injection operator.

3 V-cycle Convergence Rate

Next up we test the convergence rate of a simple V-cycle, a single pass at each level with an extra pass on the coarsest mesh, with varying mesh levels. Figure 3, below, shows the convergence rate as L_2-norm vs iteration for 1, 2, 3, 4, and 5 mesh levels. As expected, the scheme with 5 mesh levels is lightyears faster than the single mesh scheme. The total iterations for each are found in Table 2.

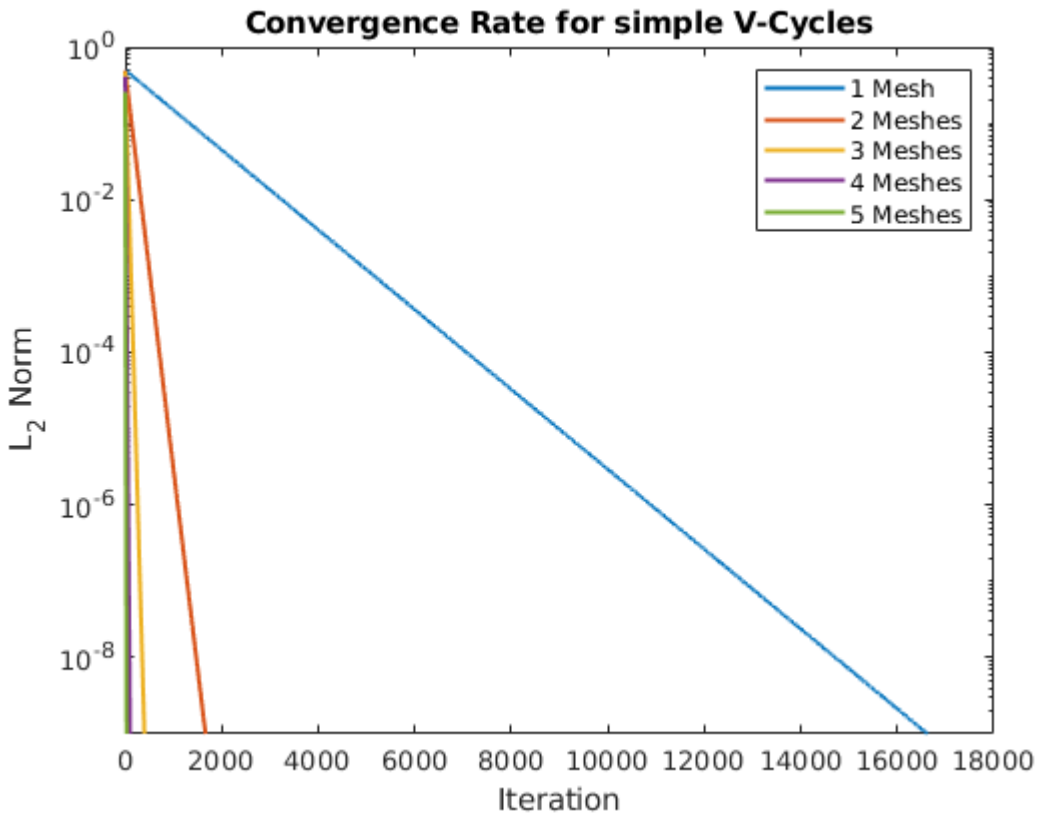

Figure 3: Convergence rate for a V-cycle with differing mesh levels

Table 2: Iterations to $L_2 \leq 10^{-9}$ for mesh levels

Mesh Levels	1	2	3	4	5
Iterations	16627	1666	400	102	29

Looking at the rates however, I was surprised by overall how slow the higher mesh levels were to converge. They did not meet the expectations set in class, leading me to believe there is a bug present, that being said, I had no luck finding it. It should also be noted that the number of iterations for the single mesh could be cut in half to better align with the others as each V-cycle actually has two passes on the finest mesh, while the single mesh case counts every single smoothing pass.

4 Over-Relaxation Convergence Rate

Figure 4, below, shows the convergence rate of a 4-level V-cycle on a 64×64 grid with differing over-relaxation parameters, as well as one case with a double smoothing pass. As you can see, the theoretically optimal over-relaxation constant of $\omega = 2/3$ does not perform the best, $\omega = 1.25$ is the winner of the constants tested. You can also see that at a certain

point (i.e. $\omega = 1.5$) the solver becomes a little unstable. While at $\omega = 1.5$ it does still converge, albeit slowly, increasing the over-relaxation constant more causes the solver to blow up. I theorise that this is because if we change the solution too much at once then the resulting residual will also be larger, and subsequently smoothing too few times won't damp out the errors that we have created. Additional smoothing passes at each step may correct this issue, although that would also be counter-productive. Unfortunately, this theory went untested. As for why the theoretically optimal over-relaxation constant was not the best...

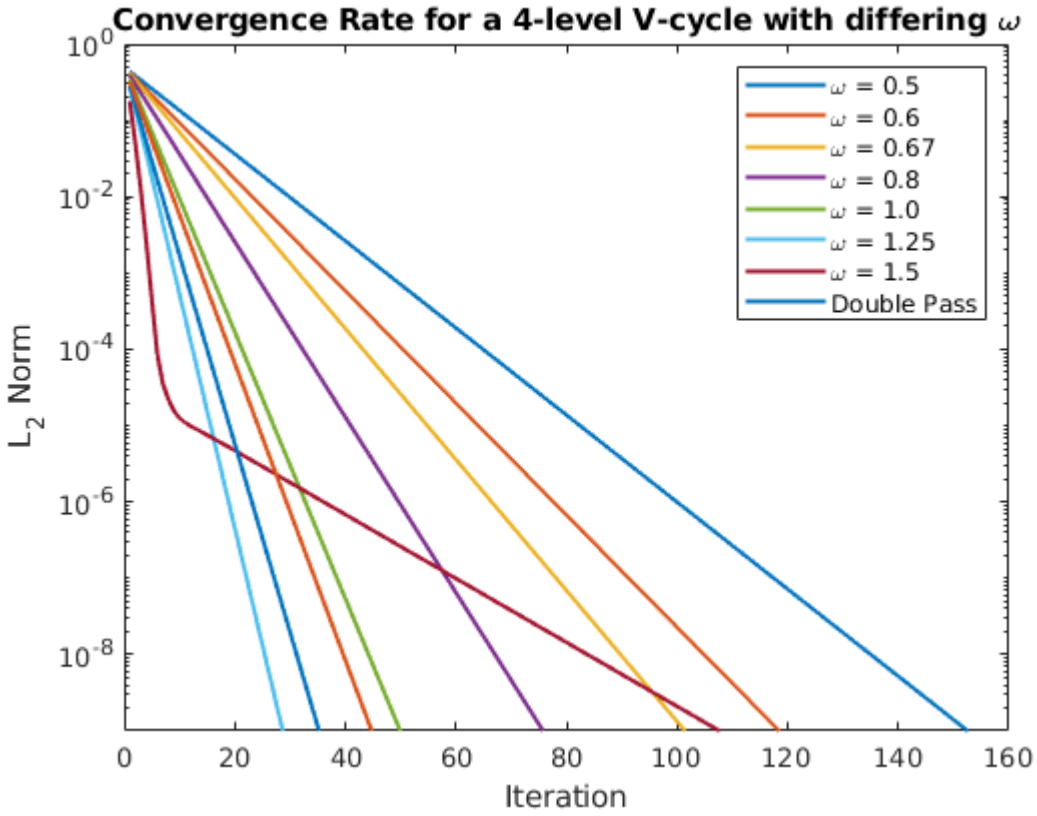

Figure 4: Convergence rate for 4-level V-cycle with differing over-relaxation

Table 3: Iterations to $L_2 \leq 10^{-9}$ for differing ω

ω	0.5	0.6	2/3	0.8	1.0	1.25	1.5	DP
Iterations	153	119	102	76	50	29	108	45

5 Convergence Rate Mesh Independence

To show that the multigrid method shows a convergence rate independent of the finest mesh size, a Poisson problem was solved with zero boundary conditions and the forcing function, $f(x, y) = x(1 - x)y(1 - y)$, given the initial condition of again, $sin(\pi x)sin(\pi y)$. Using a

standard V-cycle (one smoothing pass on each level and two on the coarsest mesh) fine mesh sizes of dimension 32, 64, 128, and 256 were used all with the coarsest mesh dimension of 4. The code was run until reaching an L_2-norm of the change in solution of 10^{-9}. Figure 5 shows the convergence rate of each while the data is summarised in Table 4, below. Based on these results I think it is safe to say that the convergence rate is indeed independent of the finest mesh size, as expected.

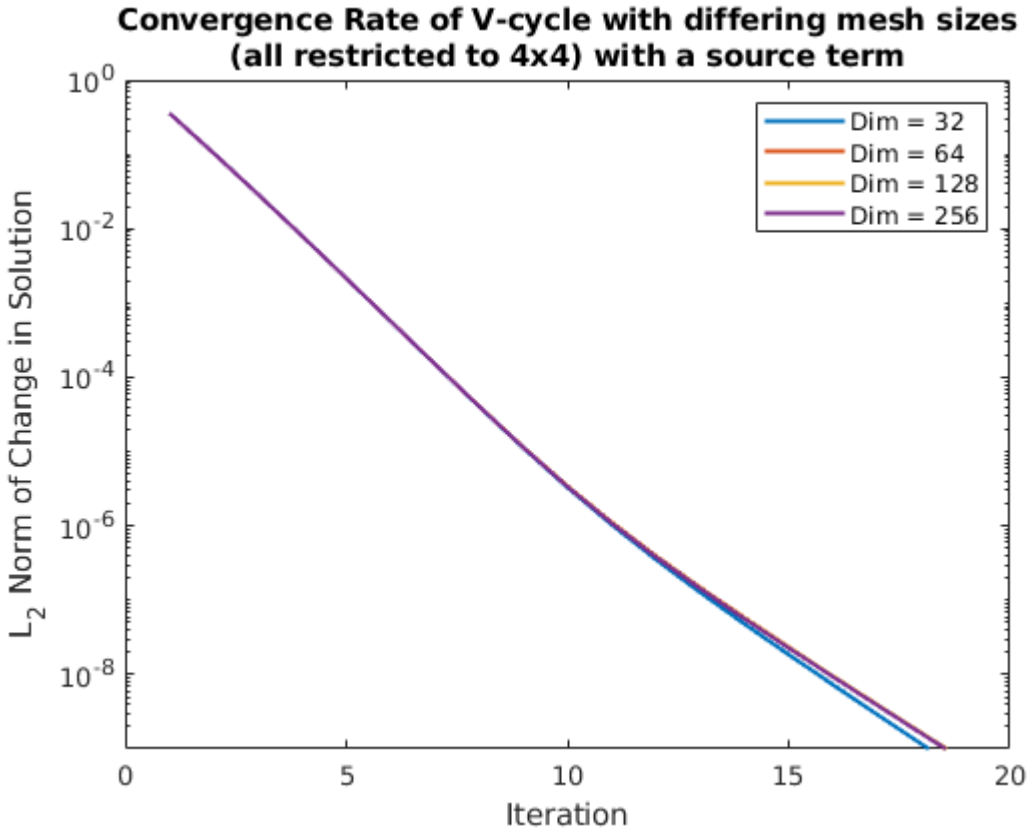

Figure 5: Convergence rate for differing fine mesh sizes

Table 4: Iterations to $L_2 \leq 10^{-9}$

Fine Mesh Size	Mesh Levels	Iterations
32×32	4	19
64×64	5	19
128×128	6	19
256×256	7	19