Multigrid Methods

Learning Objectives

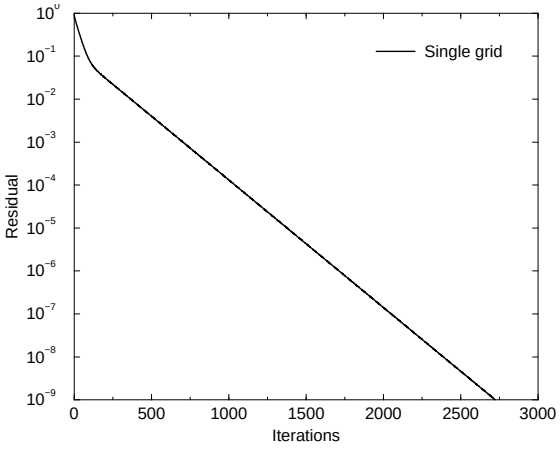
- Explain qualitatively why iterative schemes break down and how multigrid methods can ameliorate this problem.
- Describe mathematically how to apply multigrid to a linear elliptic problem by using the Correction Scheme, including answering the following questions:
  - How is the residual on the fine mesh computed?
  - How is data transfered from a fine mesh to a coarse mesh?
  - How is the coarse mesh problem formulated?
  - How is the correction transfered back to the fine mesh?
- Describe the Full Multigrid (FMG) approach and explain its advantages over simple multigrid.
- Describe the differences in approach when applying multigrid to a nonlinear problem.

1 Introduction

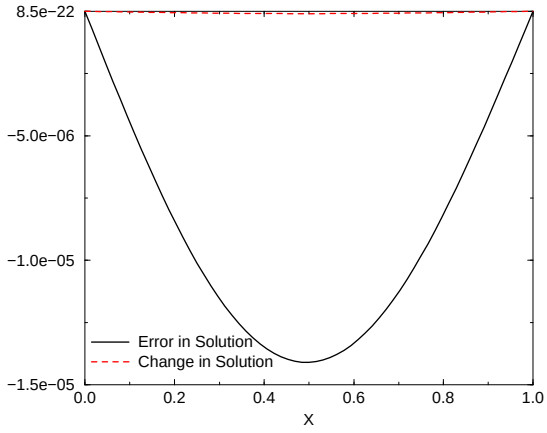
As a model for what happens with Laplace's equation in higher dimensions and for more complex boundary conditions, let's solve the extremely simple problem:

$$\frac{d^2u}{dx^2} = 0$$

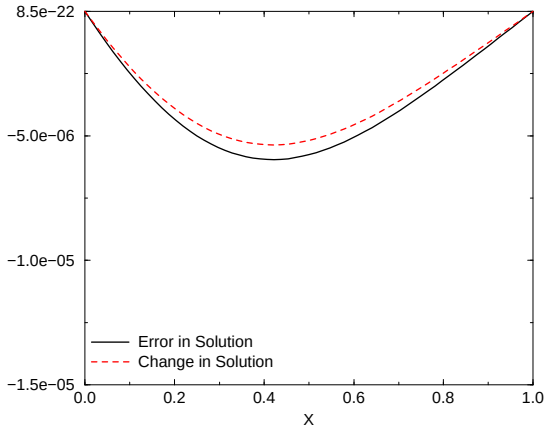
on the interval $[0, 1]$with random initial conditions and boundary conditions of $u = 0$ at both ends of the domain. Now, we know what the answer is, and even if we didn't, a single line solve would give it to us. But for harder problems those things aren't true. If we use point Gauss-Seidel with over-relaxation, the convergence history looks something like this (exact numbers depend on ω, etc).



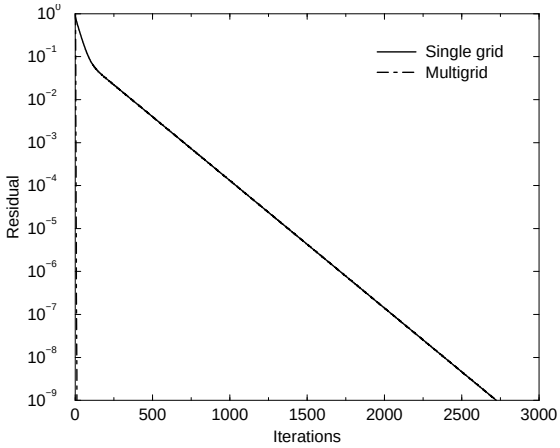
Now, why is the convergence rate so slow? Because the update to the solution at any iteration is very small compared to the error in the solution. For this case, we know the exact solution, so we can plot both the error and the change in the solution. This particular picture is for an L_∞ norm of the residual of 10^{-5}.



What we'd really like to see instead is something like this (for approximately the same convergence level)



The important point here is that the error in the solution and the change in the solution are very nearly the same in magnitude. Clearly, this will reduce the error much more quickly. The scheme that produced this picture is a multigrid scheme with five grid levels. Comparing the overall convergence rates for these two schemes (single vs 5-level multigrid), we get:



2 Multigrid

This looks great, but we need the answers to three questions:

1. Does this work for more general (and interesting) problems?
2. How does multigrid work?
3. What do we have to do to implement this? This is of course the hard question; we'll address this is in Section 3.

For a thorough introduction to multigrid analysis and code development, I strongly recommend [1, 2, 6, 3]

2.1 Does multigrid work for harder problems?

Yes, multigrid does work for harder problems. In fact, for most if not all elliptic problems, the miraculous kind of convergence rate improvement seen above can be demonstrated. The solution to such a problem can generally be obtained for the cost of doing about 10 single-mesh iterations on the finest mesh, given a properly designed scheme.

2.2 How does multigrid work?

The short answer is that multigrid works because we can always design a smoothing scheme that damps high-frequency errors.

Now, again more slowly. In multigrid terminology, one application of an iterative scheme (like point Gauss-Seidel with SOR or line Gauss-Seidel or equivalent) to update the solution at every mesh point is called a *relaxation pass* or *smoothing pass*, and the iterative scheme itself is often called the *smoother*.

High-frequency errors on a mesh with spacing Δx are those errors which can not be resolved without aliasing on a mesh with spacing $2\Delta x$. To put this another way, an error component is high-frequency if more than $N/2$ half-wavelengths (or $N/4$ wavelengths) of the error component will fit on the mesh. Given that N half-wavelengths will fit on a mesh with N points, this implies that any error mode with less than four points per wavelength must be damped on the finest mesh, because the next coarser mesh will not be able to resolve it. Fortunately, this requirement is not too hard to meet.

On the finest mesh, we apply a pass or two of the smoother to damp out the high frequency errors. Figure 1 shows an initial test function (a delta function) and the result of a applying two smoothing passes using point Gauss-Seidel with different values of ω, both of which are fairly small to optimize high-frequency damping.[1] The solution has clearly been strongly damped. The frequency-domain plots of Figure 2 are even more telling: the high-frequency errors are damped very strongly.

A multigrid scheme takes advantage of this rapid damping of the high-frequency error by performing only a few iterations on the finest mesh, then transfering the problem to a coarser mesh to damp another set of frequencies. The mechanics of this are described in the following sections.

[1] This is very different from our normal approach, which is to choose a large value of ω to optimize *low-frequency* damping.

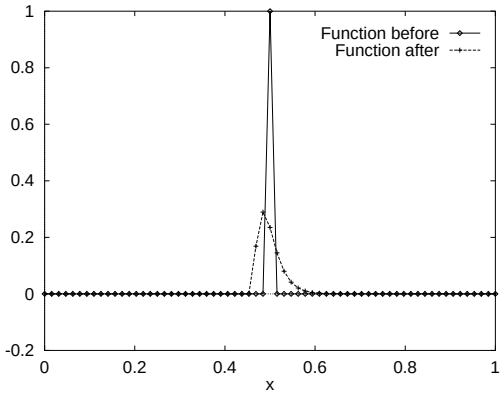


Figure 1: Solution before and after damping to remove high-frequency components

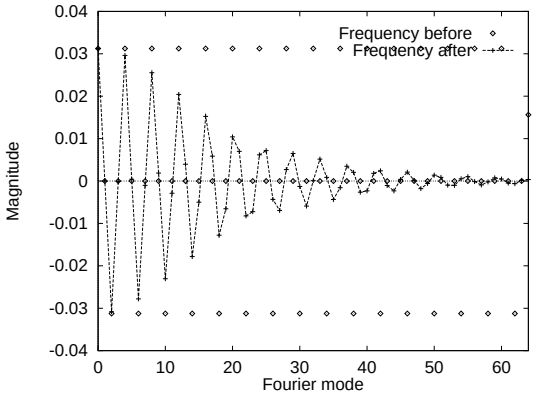


Figure 2: Frequency content of solution before and after damping to remove high-frequency components

3 Multigrid for Linear Problems: The Correction Scheme

Suppose we're trying to solve a general discrete linear problem on a mesh with characteristic size h:

$$L^h\left(u^h\right) = f^h \tag{1}$$

If we're using some iterative scheme to solve this problem, then at any point in the iterative process, we can say that the error in our approximate solution $\tilde{u}^h$ is $e^h \equiv u^h - \tilde{u}^h$. Because L^h is a linear operator, we can say that

$$f^h = L^h\left(u^h\right) = L^h\left(\tilde{u}^h + e^h\right) = L^h\left(\tilde{u}^h\right) + L^h\left(e^h\right)$$

and we can write an equation that must be satisfied by the error e^h:

$$L^h\left(e^h\right) = f^h - L^h\left(\tilde{u}^h\right) \equiv r^h \tag{2}$$

Previously, it was noted that the error will be smoothed out very rapidly. This implies that the equation for the error can be solved on a coarser mesh.[2] The problem that we must solve on the coarse mesh (with spacing $H = 2h$) is

$$L^H\left(e^H\right) = I_{h\to H}\left(r^h\right) \tag{3}$$

where $I_{h\to H}$ is a mesh-to-mesh transfer operator known as a *restriction* operator.

[2]The equation for u^h can not be transfered as easily, because there are high-frequency components to u^h that can not be resolved on a coarser mesh. See Section **??**

More will be said about these later. Once the coarse mesh problem for e^H has been solved, we transfer the result back to the fine mesh (by using a *prolongation* operator) and use it as a correction to our approximate solution $\tilde{u}^h$:

$$\tilde{u}^h \leftarrow \tilde{u}^h + I_{H\rightarrow h}\left(e^H\right) \tag{4}$$

That really is all there is to a basic multigrid scheme, except that we have to define the mesh-to-mesh transfer operators. Note that the final solution is the same with and without multigrid, because in each case on the fine mesh, the converged solution must satisfy Equation 1.

3.1 Mesh-to-Mesh Transfer Operators

For structured meshes, this really isn't a hard thing to do, because a coarse mesh is formed by taking every second point from a fine mesh. For finite-volume methods, this typically implies merging adjacent control volumes, so the fine-to-coarse mesh data transfer is typically performed as shown in Figure 3. That is, we have

$$\begin{aligned} e^H &= I_{h\rightarrow H}\left(e^h\right) \\ e_i^H &= \frac{e_{2i-1}^h + e_{2i}^h}{2} \end{aligned}$$

For transfer from a coarse mesh to a fine mesh, we can use either injection (see Figure 4) or interpolation (see Figure 5). Injection gives a simple form for

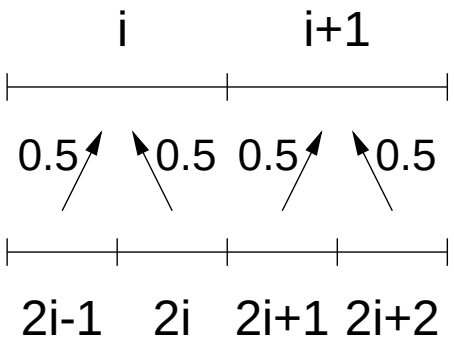


Figure 3: Fine-to-coarse mesh transfer of error in solution

$$\begin{aligned} e^h &= I_{H\rightarrow h}\left(e^H\right) \\ e_{2i-1}^h = e_{2i}^h &= e_i^H \end{aligned}$$

The interpolation scheme gives, in principle, a better transfer of the solution from coarse to fine meshes, because it is second-order accurate. In practice, these differences matter only in specialized circumstances. The interpolation scheme has the following mathematical formulation:

$$\begin{aligned} e_{2i-1}^h &= \frac{3}{4}e_i^H + \frac{1}{4}e_{i-1}^H \\ e_{2i}^h &= \frac{3}{4}e_i^H + \frac{1}{4}e_{i+1}^H \end{aligned}$$

3.2 Multigrid Cycles

The process described above is refered to in the literature as a two-level scheme.

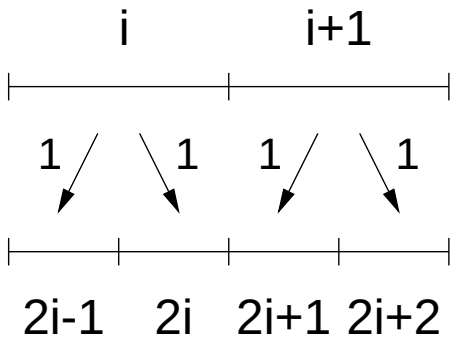


Figure 4: Coarse-to-fine mesh correction transfer by injection

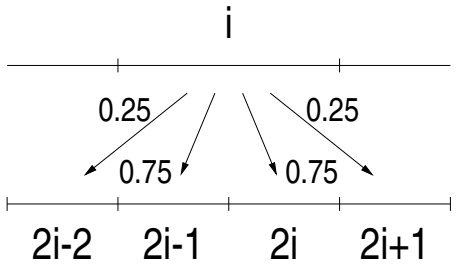
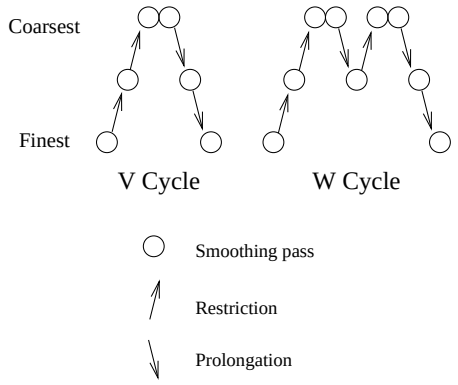


Figure 5: Coarse-to-fine mesh correction transfer by interpolation

While it is important to be able to prove good properties in a two-level scheme and to verify that one's multigrid code has those properties, few if any practical multigrid codes use a two-level scheme. Instead, coarsening is done recursively, resulting in a sequence of fine meshes. The procedure for moving from the finest mesh to coarser meshes and back again is called a *cycle*. A cycle is characterized by how many smoothing passes are performed on each mesh and by the pattern of transfer from mesh to mesh. A common example is the *V(1,1)-cycle*, so called because a schematic of it looks like a 'V' and because one smoothing pass is done while traversing the mesh heirarchy from fine to coarse and one while traversing the meshes from coarse to fine (see left of figure below). Another common choice is the *W(1,1)-cycle* (see right of figure).



Common Multigrid Cycles

A number of other commonly-used cycles exist, including sawtooth cycles, in

which there are no smoothing passes when traversing the mesh heirarchy from coarse to fine (or from fine to coarse, for another family of sawtooth cycles).

3.3 Example: CS Multigrid Applied to Poisson's Equation

Applying multigrid to Poisson's equation in one dimension is actually pretty straightforward in light of the preceding theory. In this case, the fine mesh problem (Equation 1) is

$$\frac{u_{i+1}^h - 2u_i^h + u_{i-1}^h}{h^2} = f_i \qquad (5)$$

and the corresponding residual (from Equation 2) for an approximate solution $\tilde{u}^h$ is

$$r_i^h \equiv L^h\left(e_i^h\right) = f_i - \frac{\tilde{u}_{i+1}^h - 2\tilde{u}_i^h + \tilde{u}_{i-1}^h}{h^2} \qquad (6)$$

This residual is restricted to a mesh with spacing $H = 2h$, where we must solve the problem:

$$\frac{e_{i+1}^H - 2e_i^H + e_{i-1}^H}{H^2} = I_{h\to H}\left(r_i^h\right)$$

After solving this problem, we transfer the correction e^H back to the fine mesh and use it to improve our solution.

Apply this approach to the example 1D Laplace problem for 1 to 5 meshes using V(1,1)-cycles. The thing that's interesting here is not the answer but the convergence rates, shown in the figure below.

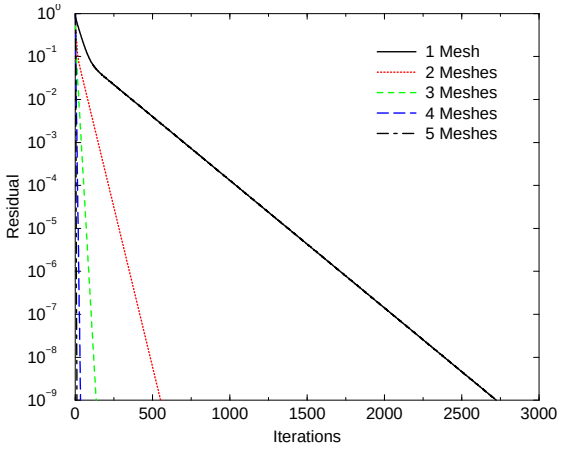


Table 1 quantifies these convergence rate improvement, including quantifying approximately the relative amount of work required for various depths of mesh nesting.

4 Multigrid for Non-Linear Problems: The Full Approximation Scheme

The discussion of the previous section assumed a linear operator L, for which

$$f^h = L^h\left(u^h\right) = L^h\left(\tilde{u}^h + e^h\right) = L^h\left(\tilde{u}^h\right) + L^h\left(e^h\right)$$

But what about non-linear problems? Well, we could simply linearize about the current solution state, but there are two problems here: first, we have to formulate the coarse mesh problem for the error differently than the fine mesh problem for the solution, and second, convergence

Levels	Cycles to 10^{-9}	Cycles 10^{-8}–10^{-9}	Cost/ Cycle	Cost 10^{-8}–10^{-9}
1	2724	337	1	337
2	552	67	$1\frac{1}{2}$	100.5
3	135	16	$1\frac{3}{4}$	28
4	34	4	$1\frac{7}{8}$	7.5
5	10	1	$1\frac{15}{16}$	$1\frac{15}{16}$

Table 1: Multigrid Efficiency for One-Dimensional Laplace Equation

rates will be adversely affected by the linearization. We can address both of these problems by transfering the entire problem to the coarse mesh, but we must be careful.

Again, we express the solution as

$$u^h = \tilde{u}^h + e^h$$

and we formulate the non-linear residual problem as

$$N^h\left(u^h\right) \equiv N^h\left(\tilde{u}^h + e^h\right) = f^h$$

Because N is non-linear, we can't find an expression for e^h directly. Instead, we solve the following coarse mesh problem:

$$N^H\left(\tilde{u}^H + e^H\right) = f^H \tag{7}$$

where $\tilde{u}^H = I_{h\to H}\tilde{u}^h$ and the source term is

$$f^H \equiv I_{h\to H}f^h - I_{h\to H}\left(N^h\tilde{u}^h\right) + N^H\left(I_{h\to H}\tilde{u}^h\right) \tag{8}$$

The first term here is the coarse mesh restriction of the source term on the fine mesh. The other two terms capture the difference in truncation error between the fine and coarse meshes. This is critical, because it ensures that the coarse mesh solution (if we converge fully on the coarse mesh) is really a coarse representation of the fine mesh solution rather than a solution with a higher discretization error. Note in particular that, with this source term, a converged fine mesh solution implies convergence on the coarse mesh as well.

Once the coarse solution $\tilde{u}^H$ has been computed, we need to transfer information back to the fine mesh. If we transfer the *solution,* we will have interpolation error associated with the entire solution, even when the change on the coarse mesh was very small. To avoid this problem, we transfer a correction instead. That is, we estimate the coarse mesh *correction* to the solution by

$$c^H = \tilde{u}^H - I_{h\to H}\tilde{u}^h \tag{9}$$

Because this correction goes to zero if the fine mesh problem has converged (because then the coarse mesh problem is automatically satisfied), then we also are assured that the fine mesh correction

$$c^h = I_{H\to h}c^H$$

will go to zero under these circumstances.

5 Miscellanea

5.1 Non-Elliptic Problems

The discussion above — and much of the multigrid theory — is based on elliptic (diffusion-dominated) problems. What about problems where the physics is more wave-like (mathematically hyperbolic)? Does multigrid still work? Yes, obviously, or it wouldn't be useful in CFD. The basic idea is still the same, that errors are eliminated on a mesh where they are well-resolved. But for hyperbolic problems some of this error elimination takes the form of propagation of errors out of the domain. This process is more rapid on coarse meshes because time steps tend to be larger and errors have fewer cells to cross to reach the domain boundary. Analysis of multigrid schemes for hyperbolic schemes still focuses on the effectiveness of smoothing schemes in reducing error on a single mesh.

5.2 Coarse Mesh Creation

For structured meshes, creating coarse meshes is easy: just take every second grid line from the fine mesh and you have a coarse mesh. For unstructured meshes, there are at least three choices: generate independent meshes at various mesh densities; generate a fine mesh and automatically coarsen it [5]; or generate only the fine mesh and combine fine mesh control volumes to create coarse mesh control volumes (introduced by Venkatakrishnan and Mavriplis [7, 4]). All three approaches work, and all three are used by various researchers. It's fair to say that agglomeration has the largest mindshare presently.

5.3 Full Multigrid

Okay, so if multigrid works so amazingly well with a lousy starting solution (i.e., one that has a lot of low-frequency error), shouldn't it work even better with a starting solution with small low-frequency error? Yes, and in fact the multigrid method itself suggests a route to creating such a starting solution. Basically, you start with a coarse mesh and get an approximate solution. Then you transfer that to a finer mesh, and use multigrid to converge that to an approximate solution, and repeat. With this approach, you don't pay the penalty of having to localize flow features like shocks and wakes on the finest, most expensive mesh. Instead, flow features are approximately located on the coarsest mesh and their location and structure become more precise with refinement.

In terms of convergence, it's fair to say that full multigrid gives you a modest improvement in the startup costs of a multigrid solver for non-linear problems (that

is, it reduces the CPU time spent getting a physical plausible solution, at which point the asymptotic convergence behavior of the multigrid scheme takes over). Also, it shouldn't surprise you too much that it turns out to be a waste of time to get too-well-converged a solution on the coarse meshes, because the discretization error is large anyway.

References

[1] Achi Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 31:333–390, 1977.

[2] Achi Brandt. Guide to multigrid development. In Wolfgang Hackbusch and Ulrich Trottenberg, editors, *Multigrid Methods*, volume 960 of *Lecture Notes in Mathematics*. Springer-Verlag, 1982.

[3] Wolfgang Hackbusch and Ulrich Trottenberg, editors. *Multigrid Methods*, volume 960 of *Lecture Notes in Mathematics*. Springer-Verlag, 1982.

[4] Dimitri J. Mavriplis and V. Venkatakrishnan. Agglomeration multigrid for viscous turbulent flows. AIAA paper 94-2332-CP, June 1994.

[5] Carl F. Ollivier-Gooch. Coarsening unstructured meshes by edge contraction. *International Journal for Numerical Methods in Engineering*, 57(3):391–414, May 2003.

[6] Klaus Stüben and Ulrich Trottenberg. Multigrid methods: Fundamental algorithms model problem analysis and applications. In Wolfgang Hackbusch and Ulrich Trottenberg, editors, *Multigrid Methods*, volume 960 of *Lecture Notes in Mathematics*. Springer-Verlag, 1982.

[7] V. Venkatakrishnan and Dimitri J. Mavriplis. Agglomeration multigrid for the 3D Euler equations. AIAA paper 94-0069, January 1994.