Chapter 7

Multigrid and Multilevel Methods

Let's now take a slightly different approach to solving our engineering field problems. Up until this point, we've concentrated primarily on finding an appropriate way to describe how our dependent variable changes spatially throughout our discretized domain. We've come up with various was to approximate this change and have defined basis functions which characterize these changes in a systematic way. In all the methods we've used thus far, this has resulted in a system of linear equations. For the finite difference method this yielded a block tridiagonal system, for finite elements, a symmetric, positive definite sparse system, and for the boundary element method, we found ourselves having to solve a smaller, but dense, nonsymmetric system.

Mutilgrid, or more generally, multilevel techniques, attack the problem from a different point of view - and can be used as a method to approximate the field problem and/or solve the resulting set of equations.

Suppose we have a system of equations that has come from one of the three methods just discussed, $Ax = b$. One way to approach the solution of such a system would be to use an iterative method (like Jacobi, Gauss-Seidel, SOR, etc.). We know that the convergence of such methods is based on, among other things, our initial guess. If we can come up with a good initial guess, we obtain our solution in fewer iterations.

One way we could come up with a "better" first guess, perhaps, is to obtain such a guess is to solve the problem on a *coarse grid*, one with fewer elements than the original problem. For example, let's assume that we have a regular grid of points (as a 5×5 system) such that we have N^2 points and the points are separated by a spacing of h. We'll call this the *fine grid*. Let's take our *coarse grid* to have $(N-1)^2/4$ points and be separated by a spacing of $2h$.

Now we solve the problem on the coarse grid. Since it is smaller, it will be easier to solve that the orginal problem. Once we have our coarse grid approximation, the idea is then to use linear interpolation to obtain approximate values at the other, fine grid, points. This now provides the starting approximation for the fine grid solution.

The hope is that the work we have done to find this better approximation will be repaid by having faster solution time for our full problem.

If one thinks recursively, it is not hard to see how this same idea might be applied to finding the coarse grid approximation and then to that approximation, and so on for increasingly coarse grids.

For the sake of arguement (and because it makes our life easier), let's assume that we have $N = (2^m - 1)$ points, where m is an integer. The coarse grids will have $(2^i - 1)^2$ points for $i = m - 1, m - 2, \ldots, 1$. Therefore, we begin solving a problem with 1 unknown and use this approximation for the next coarsest grid with $(2^2 - 1)^2 = 9$ points. This grid will then be then be used as the starting approximation to the next coarsest grid, continuing this process until we have reached the fine grid. This process of finer and finer approximations if often called *nested iteration*. An overview of the process could then be characterized as:

- Solve the problem on the coarsest grid
- Interpolate the solution to the next finer grid
- Approximate solution on this grid
- Interpolate to the next finer grid
- $\vdots$
- Interpolate to the finest grid
- Solve the fine grid problem

Now that we have the basic ideas, let's fill in some more of the details. Before we start, we'll need some basic concepts from linear algebra. Let's look at approximating the solution to the linear system,

$$A\mathbf{u} = \mathbf{b} \tag{7.1}$$

where $\mathbf{u}$ is the true solution to our problem. If we allow the approximate solution to our system to be denoted by $\mathbf{u}_m$, then a computable measure of how well $\mathbf{u}_m$ approximates $\mathbf{u}$ is given by the *residual*,

$$\mathbf{r} = \mathbf{b} - A\mathbf{u}_m \tag{7.2}$$

while the *error vector* is given by $\mathbf{e} = \mathbf{u} - {}_m$. If we rearrange the residual equation as $A\mathbf{u}_m = \mathbf{f} - \mathbf{r}$, then subtract this from our original definition of the problem, $A\mathbf{u} = \mathbf{b}$, we obtain a very important relationship between the error and the residual,

$$A\mathbf{e} = \mathbf{r} \tag{7.3}$$

which is called the *residual equation*. It basically says that the error, $\mathbf{e}$ between our true solution and our approximate solution satisfies the same set of equations as the unknown solution $\mathbf{u}$ when $\mathbf{b}$ is replaced by the residual $\mathbf{r}$.

Having previously given a brief overview of the multigrid method, one can imagine how it might be utilized. Assume that we've approximated $\mathbf{u}_m$ using some standard method. We then compute the resiudal, $\mathbf{r} = \mathbf{b} - A\mathbf{u}_m$. We know from the error equation that the true solution is $\mathbf{u} = \mathbf{u}_m + \mathbf{e}$, so if we solve the residual equation for $\mathbf{e}$ we can obtain a better solution to $\mathbf{u}$.

This can be summarized as,

- Smooth on $A\mathbf{u} = \mathbf{b}$ on Ω^h to obtain an approximation $\mathbf{u}_m^h$
- Compute the residual $\mathbf{r} = \mathbf{b} - A\mathbf{u}_m^h$
  - Smooth on the residual equation $A\mathbf{e} = \mathbf{r}$ on Ω^{2h} to obtain an approximation to the error $\mathbf{e}^{2h}$
- Correct the approximation obtain on Ω^h with the error estimate obtained on Ω^{2h} with $\mathbf{u}_m^h \leftarrow \mathbf{u}_m^h + \mathbf{e}^{2h}$

By the term *smooth* we mean the application of one of the classical relaxiation methods, i.e. Jacobi, Gauss-Seidel, etc. The procedure we have outlined in called the *coarse grid correction*. Having relaxed on the fine grid until the convergence deteriorates, we relax on the residual equaton on a coarser grid to obtain an approximation to the error itself. We then return to the fine grid to correct the approximation first obtained there.

Now we have to answer the question about how to computer the residual on Ω^h and transfer it to Ω^{2h}. The answer is to use linear interpolation. We will define a transformation that takes coarse grid vectors and produces fine grid vectors (usually denoted by $I_{2h}^h \mathbf{u}_m^{2h} = \mathbf{u}_m^h$.

To see how this works, take our standard example problem (i.e. Poisson's equation). Let's look at the one-dimensional case on the unit interval,

$$u''(x) = f(x), \quad 0 \leq x \leq 1 \tag{7.4}$$

with

$$u(0) = a \quad \text{and} \quad u(1) = b \tag{7.5}$$

Referencing the course and fine grids as show in Figure 7.1. we can obtain the values at the fine grid points by a linear interpolation.

$$u_{2i+1} = \frac{1}{2}(u_{2i+2} + u_{2i}), \quad i = 0, \ldots, (N-1)/2 \tag{7.6}$$

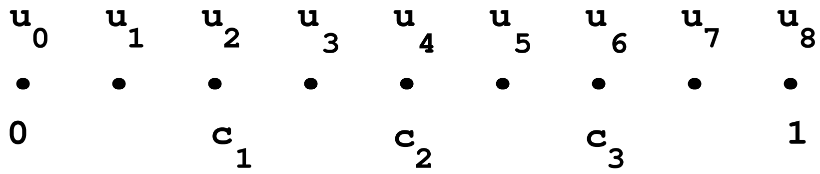

Figure 7.1: One-dimensional coarse and fine grids.

where N is the number of interior grid points (assumed to be odd). In matrix form, we have,

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_2 \\ u_4 \\ u_6 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} u_0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ u_8 \end{pmatrix} \tag{7.7}$$

You might note that this represents an *affine* transformation (a linear transformation plus a fixed vector term) in which the boundary values reside in a separate vector.

Likewise, we can easily extend this to our two dimensional grid problem (consider a 3×3 grid, for example). The interpolations are,

$$\begin{aligned} u_{i-1,j} &= \frac{1}{2}(u_{i-1,j+1} + u_{i-1,j-1}) \\ u_{i+1,j} &= \frac{1}{2}(u_{i+1,j+1} + u_{i+1,j-1}) \\ u_{i,j+1} &= \frac{1}{2}(u_{i-1,j+1} + u_{i+1,j+1}) \\ u_{i,j-1} &= \frac{1}{2}(u_{i-1,j-1} + u_{i+1,j-1}) \\ u_{i,j} &= \frac{1}{2}(u_{i-1,j+1} + u_{i+1,j+1} + u_{i-1,j-1} + u_{i+1,j-1})/4 \end{aligned} \tag{7.8}$$

this leads to the matrix system,

$$\begin{pmatrix} u_{11} \\ u_{21} \\ u_{31} \\ u_{41} \\ u_{51} \\ u_{12} \\ u_{22} \\ u_{32} \\ u_{42} \\ \vdots \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ & \vdots & & \end{pmatrix} \begin{pmatrix} u_{22} \\ u_{42} \\ u_{24} \\ u_{44} \end{pmatrix} + \frac{1}{4} \begin{pmatrix} u_{00} + u_{20} + u_{02} \\ 2u_{20} \\ u_{20} + u_{40} \\ 2u_{40} \\ u_{40} + u_{60} + u_{62} \\ \vdots \end{pmatrix} \tag{7.9}$$

Now that we have described the interpolation (which is sometimes called *prolongation* by the multigrid people) is not quite a mutigrid (MG) procedure. The MG procedure utilizes all the information on each of the coarse grids throughout the solution process.

Let's denote $G_1, \ldots, G_m$ to be our set of grids with G_1 being the coarsest grid (which corresponds to Ω^{nh}) and G_m being the fine grid (which corresponds to Ω^h. We now need to pick a particular iterative solution technique. Let's use Gauss-Seidel iteration to solve the problem. Remember that Gauss-Seidel can be represented by

$$x_i^{k+1} = \frac{1}{a_{ii}}(b_i - \sum_{i<j} a_{ij}x_j^k - \sum_{j>i} a_{ij}x_j^k), \quad i = 1, \ldots, n \tag{7.10}$$

or in vector notation,

$$\mathbf{x}^{k+1} = (D-L)^{-1}(U\mathbf{x}^k + \mathbf{b}), \quad k = 0, 1, \ldots \tag{7.11}$$

where D, L, U represent the diagonal, and strictly lower and upper triangular parts of the matrix A.

Let u_m be the approximate solution of u on the fine grid (which may have been found by nest iteration). We form the residual, $\mathbf{r} = \mathbf{b} - A\mathbf{u}_m$ and then restrict r_m to the grid, G_{m-1} to obtain values $\mathbf{r}_{m-1}$. By restriction we mean the operation of moving vectors from a fine grid toa coarse grid. The simplest requirement is to have the values of $\mathbf{r}_m$ at the grid points, G_{m-1} be taken to be the elements of $\mathbf{r}_{m-1}$. This is usually called *injection* and is denonted by the operation $I_h^{2h}\mathbf{u}_m^h = \mathbf{u}_m^{2h}$.

Next, we perform k Gauss-Seidel iterations on the equation,

$$A_{m-1}\mathbf{e}_{m-1} = \mathbf{r}_{m-1} \tag{7.12}$$

in which we wish to find an approximation to $\mathbf{e}_{m-1}$ which can be considered a crrection to the approximate solution, $\mathbf{u}_m$. Here A_{m-1} is the coefficient matrix corresponding to G_{m-1}.

This is the residual equation that we mentioned before. If we could solve the error equation on the fine grid, $A_{\mathbf{e}} = \mathbf{r}_m$. Then,

$$A(\mathbf{u}_m + \mathbf{e}) = \mathbf{b} - \mathbf{r}_m + \mathbf{r}_m = \mathbf{b} \tag{7.13}$$

such that $\mathbf{u}_m + \mathbf{e}$ is the desired solution to the system $A\mathbf{u} = \mathbf{b}$. Where again, $\mathbf{e}$ is considered the correction to the approximate solution. If $\mathbf{e}_{m-1}$ is obtained from our equation and then is interpolated back to G_m to give $\mathbf{e}_m$ on the fine grid, then $\mathbf{e}_m$ is an approximation to $\mathbf{e}$ and may be considered as an approximate correction to $\mathbf{u}_m$. We now continue this process on coarser grids,

- Smooth current $\mathbf{u}_m$ on G_m to give $\tilde{\mathbf{u}}_m$

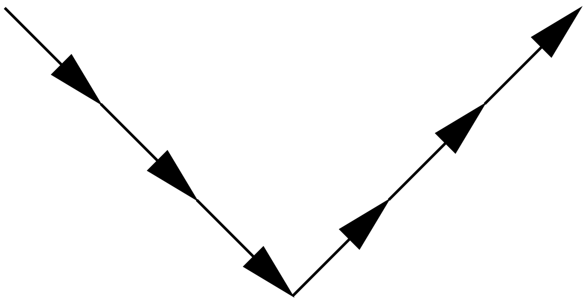

Figure 7.2: V-cycle.

- Form $\mathbf{r}_m = \mathbf{b} - A_m\tilde{\mathbf{u}}_m$
- For i=k down to 3
  - Restrict $\mathbf{r}_i$ onto G_m to give $\tilde{\mathbf{r}}_{i-1}$
  - Smooth on $A_{i-1}\mathbf{e} = \tilde{\mathbf{r}}_{i-1}$ starting with 0 to give $\mathbf{e}_{i-1}$
  - Form $\mathbf{r}_{i-1} = \tilde{\mathbf{r}}_{i-1} - A_{i-1}\mathbf{e}_{i-1}$
- Restrict $\mathbf{r}_2$ to G_1 to give $\tilde{\mathbf{r}}_1$
- Solve exactly $A_1\mathbf{e} = \tilde{\mathbf{r}}_1$ to give $\tilde{\mathbf{e}}_1$
- For i=1, ..., m-2
  - Interpolate $\tilde{\mathbf{e}}_i$ to G_{i+1} and add to $\mathbf{e}_{i+1}$
  - Smooth on $A_{i+1}\mathbf{e} = \tilde{\mathbf{r}}_{i+1}$
- Interpolate $\tilde{\mathbf{e}}_{m-1}$ to get $\hat{\mathbf{e}}_m$
- Add $\hat{\mathbf{e}}_m$ to $\tilde{\mathbf{u}}_m$ to obtain $\mathbf{u}_m$ on G_m

The procedure we have outlined is called the *V-cycle* as the algorithm starts at the finest grid and we smooth our way down to the coarsest grid and then interpolate back up to our finest grid. The top of the V represents the fine grid while the lowest point on the V represents the coarsest grid as shown in figure 7.2

Another variant is to do two corection cycles at each level before returning to the next higher level. This can is known as the *W-cycle*. When one joins nested iteration with the *V-cycle* we obtain what is called the *full multigrid V-cycle* or (FMV).

7.0.3 Computational Considerations

Let's now step through the *V-cycle* and consider the compuational costs for the various steps.

We assume that $N = 2^{m-1}$ and will assume that our coarse grid corrections will move in increments of $2h$. We will denote the components of the fine grid, Ω^h as A_m, those on the next grid, Ω^{2h} as A_{m-1}, and so on.

The first step in the *V-cycle* is to do smoothing steps on the fine grid. We'll assume that the *smoother* is the Gauss-Seidel method, such that if we take one smoothing step starting with $\mathbf{u}_m$, the next approximate solution is,

$$\tilde{\mathbf{u}}_m = (D_m - L_m)^{-1}(U_m \mathbf{u}_m + \mathbf{b}) \tag{7.14}$$

The vector $\tilde{\mathbf{u}}_m$ replaces $\mathbf{u}_m$ in storage.

Next we compute the residual,

$$\mathbf{r}_m = \mathbf{b} - A_m \tilde{\mathbf{u}}_m \tag{7.15}$$

the residual is then restricted to G_{m-1}

$$\tilde{\mathbf{r}}_{m-1} = R_m \mathbf{r}_m \tag{7.16}$$

remembering that we have chosen to restrict by means of injection. Now we move to the $m-1$ level. We smooth,

$$A_{m-1}\mathbf{e} = \tilde{\mathbf{r}}_{m-1} \tag{7.17}$$

In practice, one usually starts with a zero guess. Using Gauss-Seidel, this yields (for one step),

$$\mathbf{e}_{m-1} = (D - L)^{-1}\tilde{\mathbf{r}}_{m-1}. \tag{7.18}$$

The residual is then computed as restriced by,

$$\tilde{\mathbf{r}}_{m-2} = R_{m-1}\tilde{\mathbf{r}}_{m-1} - R_{m-1}A_{m-1}\mathbf{e}_{m-1}. \tag{7.19}$$

The process now repeats at each level,

$$\mathbf{e}_i = (D - L)^{1}\tilde{\mathbf{r}}_i \quad i = m-2, \ldots, 2 \tag{7.20}$$

and

$$\tilde{\mathbf{r}}_{i-1} = R_i(\tilde{\mathbf{r}}_i - A_i \mathbf{e}_i) \quad i = m-2, \ldots, 2. \tag{7.21}$$

Once we reach the coarsest level, we solve the equation,

$$A_1 \mathbf{e} = \tilde{\mathbf{r}}_1 \tag{7.22}$$

exactly to give the value for $\tilde{\mathbf{e}}_1$.

At this point, the ascending portion of the V-cycle begins. At each level,$i = 2, \ldots, m-1$, we perform, first, an interpolation,

$$\hat{\mathbf{e}}_i = \mathbf{e}_i + I_i \tilde{\mathbf{e}}_{i-1} \tag{7.23}$$

This interpolates $\tilde{\mathbf{e}}_{i-1}$ to level i and then adds to the current correction vector, $\mathbf{e}_i$ to give $\hat{\mathbf{e}}_i$.

Now we take a smoothing step,

$$\tilde{\mathbf{e}}_i = (D - L)^{-1}(U\hat{\mathbf{e}}_i + \tilde{\mathbf{r}}_i) \tag{7.24}$$

Once we have finished this previous step, we will have $\tilde{\mathbf{e}}_{m-1}$ which is interpolated to $\hat{\mathbf{e}}_m$ and then added to $\tilde{\mathbf{u}}_m$ to obtain the new approximation on the fine grid,

$$\mathbf{u}_m^1 = \tilde{\mathbf{u}}_m + \tilde{\mathbf{e}}_m \tag{7.25}$$

This (finally) finishes our V-cycle.

7.0.4 Storage and Computational Costs

A word or two about storage and computational costs.

It can be shown [23] that there is an extra cost to store the corse grids and additional right hand sides due to the coarse grids. The extra storage costs can be expressed in terms of the storage cost for the original problem represented on the fine grid, S as,

$$S_c = S(2^{-d} + 2^{-2d} + \cdots + 2^{-(m-1)d}) = \frac{S(2^{-dm} - 1)}{2^{-d} - 1} < \frac{S}{1 - 2^{-d}} \tag{7.26}$$

where S_c is the storage of the coarse grid(s) and d is the dimensional of the problem. Thus for $d = 2$, $S_c < \frac{4}{3}S$ and for $d = 3$, we have $S_c < \frac{8}{7}S$. The same relationship holds for the right hand side.

A similar relationship occurs for the amount of computational work that is required. It is common to compare the amount of compuational effort in terms of a *work unit, W.* The work unit is defined to by the cost of a residual computation (or a relaxation sweep) on the fine grid. The work on each grid is roughly proportional to the number of unknowns so that for a d-dimensional problem, the work on each level is about 2_{-d} the amount of work on the next finer grid. This yields a relationship,

$$[2 + 3(2^{-d} + \cdots + 2^{-(m-1)d)}]W = [2 + \frac{3(2^{-dm} - 1}{2_{-d} - 1}]W \tag{7.27}$$

The factor of 3 accounts for the fact that on each coarse grid there are two relaxation (smoothing) sweeps plus one residual evaluation. For $d = 1$ this yields $8W$, for $d = 2$, we obtain $6W$, and for $d = 3$ we get $5W$.