Lectures 12-13: Introduction to Multigrid methods

The programs Decay.m and Damping.m investigate the behaviour of our 3 iterative methods on different grids and with different error wavenumbers. Essentially we have found:

(1) Jacobi and Gauss-Seidel quickly smoothed out the high wavenumbers (oscillations on a small scale). However, they were very slow to iron out the low wavenumbers (long waves). After a few iterations the error and residuals were dominated by these low wavenumbers. Thus even if the solution to the entire problem varies on a short length-scale, the errors after a few iterations vary on a large scale.

(2) These low wavenumber modes do not require a lot of points to resolve them well. Furthermore, on coarser grids they would be damped more quickly. This suggests that it might be worth swapping between more than one grid. This is the idea behind **multigrid methods.**

(3) The over-relaxation methods (SOR), although their overall convergence was much faster than Jacobi/GS, did not smooth the solution at all well. At all stages, the error vector, though small in magnitude, consisted of a mixture of wavenumbers. This does not lend itself well to the idea of swapping between grids, and so surprisingly, we will use our 2nd best method (Gauss-Seidel) on the various grids.

Two grid strategy

Suppose we have a coarse grid C and a fine grid F containing twice as many points in each direction. We wish to solve $A_F\mathbf{u}_F = \mathbf{b}_F$ on the fine grid. The corresponding approximation on C is $A_c\mathbf{u}_c = \mathbf{b}_c$.

(a) We begin with a few Gauss-Seidel sweeps on F to get rid of the short waves, obtaining an approximation $\mathbf{u}_F^*$ to $\mathbf{u}_F$.

(b) Next we calculate the residuals $\mathbf{r}_F = A\mathbf{u}_F^* - \mathbf{b}$.

(c) Now we transfer to the coarser grid C using a restriction operator $\mathcal{R}$: F $\rightarrow$ C, to find $\mathbf{r}_c = \mathcal{R}\mathbf{r}_F$.

(d) On the coarser grid we calculate the error $\mathbf{z_c}$ quickly, where $A_c\mathbf{z_c} = \mathbf{r_c}$.

(e) We now use an interpolation operator $\mathcal{P}$ to map the error $\mathbf{z_c}$ to $\mathbf{z_F} = \mathcal{P}\mathbf{z_c}$.

(f) We then modify our fine estimate by the calculated error, so that $\mathbf{u}_F^* + \mathbf{z_F}$ is the new solution estimate.

(g) We then take a few more sweeps using Gauss-Seidel, in case any short wave errors have crept in. This completes a cycle. If we wish we can repeat the process using our new estimate.

Before we can implement this scheme, we have to decide how to **interpolate** from a coarse mesh onto a finer one, and also, how best to **restrict** our solution on a finer mesh to a less accurate coarse mesh. If we do this linearly, we can represent these transformations by rectangular matrices, $\mathcal{P}$ and $\mathcal{R}$. Let these have dimensions $\mu \times \nu$ and $\nu \times \mu$ respectively. As each grid point on the coarse mesh is also a point on the fine mesh, the simplest idea would be to use the same value at the new point. But it might be a good idea to use a linear combination of all the neighbouring points. We want to minimize the errors introduced by transfer between grids. Ideally, we would like $\mathcal{PR}$ and $\mathcal{RP}$ to be identity matrices but that

would overconstrain the problem. It is a good idea to require the interpolation operator $\mathcal{P}$ and our restriction operator $\mathcal{R}$ to be adjoints with respect to the scalar product, $\{u, v\}$. (Think of the total "energy", $\int u^2\, dA$.) If we denote the coarse and fine meshes by C and F, and let u_c and v_F be any vectors in the respective meshes. Then

$$\{\mathbf{u}_c, \mathcal{R}\mathbf{v}_F\} = \{\mathcal{P}\mathbf{u}_c, \mathbf{v}_F\} \quad \Longrightarrow \quad \mathcal{R} = P^T/4, \tag{12.1}$$

allowing for a factor of 4 from the halving of the steplengths (the vector $\mathbf{v}_F$ is 4 times the length of $\mathbf{u}_c$.) Note effectively we are replacing the matrix A_c by $\mathcal{R}\mathrm{A}_F\mathcal{P}$ – this is known as the Galerkin condition.

The interpolation operator is easiest to choose, as we have less information to work with. The natural choice is to use the average of nearest neighbours:

$$\begin{aligned} u_F(2i, 2j) &= u_c(i, j) \\ u_F(2i+1, 2j) &= \tfrac{1}{2}\big(u_c(i, j) + u_c(i+1, j)\big) \\ u_F(2i, 2j+1) &= \tfrac{1}{2}\big(u_c(i, j) + u_c(i, j+1)\big) \\ u_F(2i+1, 2j+1) &= \tfrac{1}{4}\big(u_c(i, j) + u_c(i+1, j) + u_c(i, j+1) + u_c(i+1, j+1)\big) \end{aligned} \tag{12.2}$$

Using (12.1), the appropriate restriction operator is then $\mathbf{u}_c = \mathcal{R}\mathbf{u}_F$ where

$$\begin{aligned} u_c(i, j) = \tfrac{1}{4}u_F(2i, 2j) + \ldots \\ + \tfrac{1}{8}\left(u_F(2i+1, 2j) + u_F(2i, 2j+1) + u_F(2i-1, 2j) + u_F(2i, 2j-1)\right) + \ldots \\ + \tfrac{1}{16}\left(u_F(2i+1, 2j+1) + u_F(2i-1, 2j+1) + u_F(2i-1, 2j-1) + u_F(2i+1, 2j-1)\right) \end{aligned} \tag{12.3}$$

Although this is more complex than just $u_c(i, j) = u_F(2i, 2j)$, it keeps the errors under better control when flipping between grids.

More than 2 grids

The coarser grid has twice the steplength, but it may still be slow to solve for the errors as in step (d) above. But there is nothing to prevent us from performing the same algorithm again, transferring onto a still coarser grid, If our initial N is of the form $2^p + 1$, we may transfer all the way down to $p = 1$ and then move up the chain back to the finest grid.

The program MultigridV.m performs a cycle transferring down the grids to solve for the error on a coarse grid, and then interpolating the **error** back up to the finest grid. This is known as a 'V-cycle'. It is extremely efficient, but can still be improved upon!

Full Multigrid

The Multigrid algorithm starts on the fine grid with an arbitrary initial guess. Instead, we could solve the problem on a coarse grid and interpolate down to give a better initial estimate, and hence reduce the iterations required. The program FullMG.m starts on the coarsest grid and interpolates to the next grid. There it solves the problem with a 2 grid algorithm, and then interpolates again, each time using a V-cycle down to the coarsest grid. Not only is it faster, but full MG also obtains the solution on every grid as opposed to the final grid only.