6.3 Multigrid Methods

The Jacobi and Gauss-Seidel iterations produce smooth errors. The error vector e has its high frequencies nearly removed in a few iterations. But low frequencies are reduced very slowly. Convergence requires $O(N^2)$ iterations—which can be unacceptable. The extremely effective **multigrid idea** is to change to a coarser grid, on which "smooth becomes rough" and low frequencies act like higher frequencies.

On that coarser grid a big piece of the error is removable. *We iterate only a few times before changing from fine to coarse and coarse to fine.* The remarkable result is that multigrid can solve many sparse and realistic systems to high accuracy in a **fixed number of iterations**, not growing with n.

Multigrid is especially successful for symmetric systems. The key new ingredients are the (rectangular!) matrices R and I that change grids:

1. A ***restriction matrix*** $\boldsymbol{R}$ transfers vectors from the fine grid to the coarse grid.

2. The return step to the fine grid is by an ***interpolation matrix*** $\boldsymbol{I = I^h_{2h}}$.

3. The original matrix A_h on the fine grid is approximated by $\boldsymbol{A_{2h} = RA_hI}$ on the coarse grid. You will see how this A_{2h} is smaller and easier and faster than A_h. I will start with interpolation (a 7 by 3 matrix I that takes 3 v's to 7 u's):

Interpolation $\boldsymbol{Iv = u}$
$\boldsymbol{u}$ on the fine ($\boldsymbol{h}$) grid from
$\boldsymbol{v}$ on the coarse ($\boldsymbol{2h}$) grid
values are the $\boldsymbol{u}$'s.

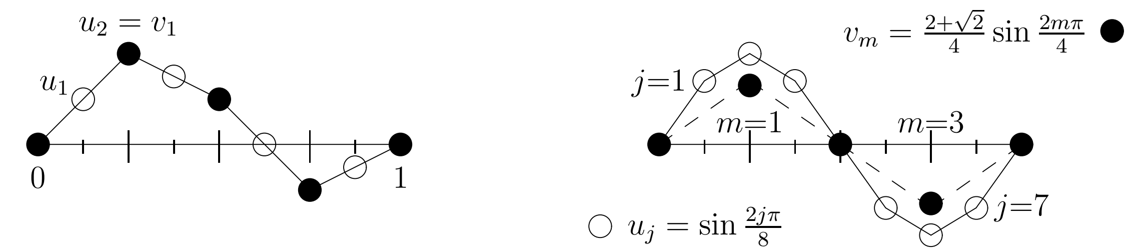
$$\frac{1}{2}\begin{bmatrix} 1 & & \\ 2 & & \\ 1 & 1 & \\ & 2 & \\ & 1 & 1 \\ & & 2 \\ & & 1 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} v_1/2 \\ v_1 \\ v_1/2+v_2/2 \\ v_2 \\ v_2/2+v_3/2 \\ v_3 \\ v_3/2 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} \qquad (1)$$

This example has $h = \frac{1}{8}$ on the interval $0 \le x \le 1$ with zero boundary conditions. The seven interior values are the u's. The grid with $2h = \frac{1}{4}$ has three interior v's.

Notice that u_2, u_4, u_6 from rows 2, 4, 6 are the same as v_1, v_2, v_3! Those coarse grid values v_j are just moved to the fine grid at the points $x = \frac{1}{4}, \frac{2}{4}, \frac{3}{4}$. The in-between values u_1, u_3, u_5, u_7 on the fine grid are coming from *linear interpolation* between $0, v_1, v_2, v_3, 0$:

Linear interpolation in rows 1, 3, 5, 7 $\qquad u_{2j+1} = \frac{1}{2}(v_j + v_{j+1})\,. \qquad (2)$

The odd-numbered rows of the interpolation matrix have entries $\frac{1}{2}$ and $\frac{1}{2}$. We almost always use grid spacings $h, 2h, 4h, \ldots$ with the convenient ratio 2. Other matrices I are possible, but linear interpolation is easy and effective. Figure 6.10a shows the new values u_{2j+1} (open circles) between the transferred values $u_{2j} = v_j$ (solid circles).



(a) Linear interpolation by $u = I_{2h}^h v$ (b) Restriction by $R_h^{2h} u = \frac{1}{2}(I_{2h}^h)^{\mathrm{T}} u$

Figure 6.10: Interpolation to the h grid (7 u's). Restriction to the $2h$ grid (3 v's).

When the v's represent smooth errors on the coarse grid (because Jacobi or Gauss-Seidel has been applied on that grid), interpolation gives a good approximation to the errors on the fine grid. A practical code can use 8 or 10 grids.

The second matrix we need is a **restriction matrix** $\boldsymbol{R_h^{2h}}$. It transfers u on a fine grid to v on a coarse grid. One possibility is the one-zero "injection matrix" that simply copies v from the values of u at the same points on the fine grid. This ignores the odd-numbered fine grid values u_{2j+1}. Another possibility (which we adopt) is the **full weighting operator $\boldsymbol{R}$** that comes from transposing I_{2h}^h.

Fine grid h to coarse grid 2h by a restriction matrix $\boldsymbol{R_h^{2h} = \frac{1}{2}(I_{2h}^h)^{\mathrm{T}}}$

Full weighting $\boldsymbol{Ru = v}$
Fine grid $\boldsymbol{u}$ to coarse grid $\boldsymbol{v}$

$$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & & \\ & & 1 & 2 & 1 & & \\ & & & & 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}. \quad (3)$$

The effect of this restriction matrix is shown in Figure 6.10b. We intentionally chose the special case in which $u_j = \sin(2j\pi/8)$ on the fine grid (*open circles*). Then v on the coarse grid (*dark circles*) is also a pure sine vector. *But the frequency is doubled* (a full cycle in 4 steps). So a smooth oscillation on the fine grid becomes "half as smooth" on the coarse grid, which is the effect we wanted.

Interpolation and Restriction in Two Dimensions

Coarse grid to fine grid in two dimensions from bilinear interpolation: Start with values $v_{i,j}$ on a square or rectangular coarse grid. Interpolate to fill in $u_{i,j}$ by a sweep (interpolation) in one direction followed by a sweep in the other direction. We could allow two spacings h_x and h_y, but one meshwidth h is easier to visualize. A horizontal sweep along row i of the coarse grid (which is row $2i$ of the fine grid) will fill in values of u at odd-numbered columns $2j+1$ of the fine grid:

$$\textbf{Horizontal sweep} \qquad u_{2i,2j} = v_{i,j} \quad \text{and} \quad u_{2i,2j+1} = \frac{1}{2}(v_{i,j} + v_{i,j+1}) \quad \text{as in 1D.} \qquad (4)$$

Now sweep vertically, up each column of the fine grid. Interpolation will keep those values (4) on even-numbered rows $2i$. It will average those values to find $\boldsymbol{u} = \boldsymbol{I2\mathrm{D}\,v}$ on the fine-grid odd-numbered rows $2i+1$:

$$\begin{array}{lrcl} \textbf{Vertical sweep} & u_{2i+1,2j} & = & (v_{i,j} + v_{i+1,j})/2 \\ \textbf{Averages of (4)} & u_{2i+1,2j+1} & = & (v_{i,j} + v_{i+1,j} + v_{i,j+1} + v_{i+1,j+1})/4\,. \end{array} \qquad (5)$$

The entries in the tall thin coarse-to-fine interpolation matrix $I2\mathrm{D}$ are $1, \frac{1}{2}$, and $\frac{1}{4}$.

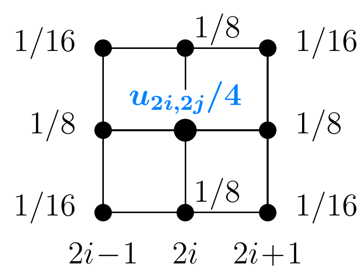
The full weighting fine-to-coarse restriction operator $R2\mathrm{D}$ is the *transpose* $I2\mathrm{D}^{\mathrm{T}}$, multiplied by $\frac{1}{4}$. That factor is needed (like $\frac{1}{2}$ in one dimension) so that a constant vector of 1's will be restricted to a constant vector of 1's. (The entries along each row of the wide matrix R add to 1.) This restriction matrix has entries $\frac{1}{4}, \frac{1}{8}$, and $\frac{1}{16}$ and *each coarse-grid value v is a weighted average of nine fine-grid values u*:

Restriction matrix $\boldsymbol{R = \frac{1}{4} I^{\mathrm{T}}}$

Row $\boldsymbol{i, j}$ of $\boldsymbol{R}$ produces $\boldsymbol{v_{i,j}}$

$\boldsymbol{v_{i,j}}$ uses $\boldsymbol{u_{2i,2j}}$ and 8 neighbors

The nine weights add to 1



You can see how a sweep along each row with weights $\frac{1}{4}, \frac{1}{2}, \frac{1}{4}$, followed by a sweep down each column, gives the nine coefficients in that "restriction molecule." Its matrix $R2\mathrm{D}$ is an example of a *tensor product* or *Kronecker product* `kron`(R, R). A 3 by 7 matrix R in one dimension becomes a 9 by 49 restriction matrix $R2\mathrm{D}$ in two dimensions.

Now we can transfer vectors between grids. We are ready for the ***geometric multigrid*** method, when the geometry is based on spacings h and $2h$ and $4h$. The idea extends to triangular elements (each triangle splits naturally into four similar triangles). The geometry can be more complicated than our model on a square.

When the geometry becomes too difficult, or we are just given a matrix, we turn (in the final paragraph) to ***algebraic multigrid***. This will imitate the multi-scale idea, but it works directly with $Au = b$ and not with any underlying geometric grid.

A Two-Grid V-Cycle (a v-cycle)

Our first multigrid method only involves two grids. The iterations on each grid can use Jacobi's $I - D^{-1}A$ (possibly weighted by $\omega = 2/3$ as in the previous section) or Gauss-Seidel. For the larger problem on the fine grid, iteration converges slowly to