MAT 228A Notes

Sam Fleischer

November 15, 2016

1 Multigrid

1.1 Transfer Operators

1.1.1 Restriction Operator

A map from a grid of spacing h to a grid of spacing $2h$. Injection (throwing out in-between points) is not good - we shouldn't throw away information. Instead, take a weighted average of the three points "above" you.. here are the stensils:

$$I_h^{2h} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

In 2D,

$$I_h^{2h} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Pseudocode for 1D:

```
• Loop over coarse mesh $j_c$
    – Compute corresponding fine grid point $j_f$.
    – Compute $u_c(j_c) = \frac{1}{4}(u_f(j_f - 1) + 2u_f(j_f) + u_f(j_f + 1))$.
```

Another stensil in 2D is

$$I_h^{2h} = \frac{1}{8}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

It is less desirable but cheaper.

1.1.2 Interpolation (Prolongation) Operator

In 1D, we could copy the data such that $(u_h)_{2j_c} = (u_{2h})_{j_c}$ and interpolate for the points in-between: $(u_h)_{2j_c-1} = \frac{1}{2}((u_{2h})_{j_c-1} + (u_{2h})_{j_c})$. Here is the stensil:

$$I_{2h}^{h} = \frac{1}{2}\left]\begin{matrix} 1 & 2 & 1 \end{matrix}\right[$$

In 2D,

$$I_{2h}^{h} = \frac{1}{4}\left]\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix}\right[$$

where the backwards brackets signify a function from the coarse grid to the fine grid, rather than from the fine grid to the coarse grid. Pseudocode for 1D linear interpolation:

```
• $u_f = 0$
```

- loop of coarse mesh j_c
  - * compute j_f
  - * $u_f(j_f - 1) = u_f(j_f - 1) + \frac{1}{2} u_c(j_c)$
  - * $u_f(j_f) = u_c(j_c)$
  - * $u_f(j_f + 1) = u_f(j_f + 1) + \frac{1}{2} u_c(j_c)$

1.1.3 1D example: 9 pt. fine mesh, 5 pt. coarse mesh

Fine mesh spacing is $1/8$, coarse mesh spacing is $1/4$. So for Dirichlet problem, $n_f = 7$, $n_c = 3$. Here is the matrix for full weighting. It is a 3×7 matrix since it is mapping grid-size h to grid-size $2h$.

$$I_h^{2h} = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 \end{bmatrix}$$

The interpolation operator is a 7×3 matrix

$$I_{2h}^h = \frac{1}{2} \begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

So $I_h^{2h} = \frac{1}{2} \left(I_{2h}^h\right)^T$. Choose the weighted inner product:

$$\langle u, v \rangle_h := h u^T v$$

So,

$$\begin{aligned} \langle u_h, I_{2h}^h v_{2h} \rangle_h &= h u_h^T I_{2h}^h v_{2h} \\ &= h \Big(\left(I_{2h}^h\right)^T u_h \Big)^T v_{2h} \\ &= h \left(2 I_h^{2h} u_h\right)^T v_{2h} \\ &= 2h \left(I_h^{2h} u_h\right)^T v_{2h} \\ &= \langle I_h^{2h} u_h, v_{2h} \rangle_{2h} \end{aligned}$$

So these two operators are each others' adjoints, when considering the proper inner product. Review of the definition of adjoint: A^* of A:

$$\langle x, Ay \rangle = \langle A^* x, y \rangle$$

1.2 Solving on the coarse grid

$$\cdots \to \underbrace{\text{smooth} \to \text{residual}}_{\text{fine mesh}} \to \text{residual} \underbrace{\to \text{solve} \to}_{\text{coarse mesh}} \text{interpolate} \to \underbrace{\text{correct} \to \text{smooth}}_{\text{fine mesh}} \to \dots$$

Solving requires

$$L_{2h} e_{2h} = f_{2h}$$

What is L_{2h}? One way to define L_{2h} is

$$L_{2h} I_h^{2h} L_h I_{2h}^h$$

This is called the Galerkin coarse grid operator. What does this look like? Here is the stensil.. if L_h is

$$L_h = \frac{1}{h^2}\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

then

$$L_{2h} = \frac{1}{(2h)^2}\begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

The problem with this is that L_{4h} is different, etc. so we get a series of operators for multigrid. A *much* simpler approach is to re-discretize the problem at each level, and just use the discrete Laplacian at that level:

$$L_{2h} = \frac{1}{(2h)^2}\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

This is geometric multigrid, whereas the former is algebraic multigrid. AMG is much more general, but is slower than GMG. GMG is useful since we know this is coming from a PDE.

1.3 How well does this work?

Let ν_1 be the number of presmooth steps and ν_2 the number of postsmooth stpes. Then define $\nu = \nu_1 + \nu_2$. The spectral radius of the two-grid, 2D model problem with full-weighting, bilinear interpolation, rediscretization.

Spectral radius

ν	ω-Jacobi	GS-RB
1	0.6	0.25
2	0.360	0.074
3	0.216	0.053
4	0.137	0.041

So we have mesh-independent convergence. We see $\rho < C < 1$, i.e. the spectral radius is bounded away from 1. So

$$\rho^k = \varepsilon$$
$$k = \frac{\ln \varepsilon}{\ln \rho}$$

We will discuss computational expense when we talk about *multi*grid later.

How much smoothing should we do? More smoothing is better, but there are diminishing returns the more smoothing we do. We need to balance the smoothing work with iteration work. Suppose the work per iteration is $\nu + w$, where ν is the smoothing work and w is everything else. So work per digit of accuracy is

$$\frac{\nu + w}{-\log_{10}(\rho)} = (\text{work})(\#\text{ iterations per digit})$$