Multi Grid

C. T. Kelley
NC State University
tim_kelley@ncsu.edu

Research Supported by NSF, DOE, ARO, USACE

DTU ITMAN, 2011

Outline

- This lecture
- Revised transport lecture
- Two papers
- MATLAB examples

Tuesday's directory has also been revised.

References I

- W. L. Briggs, V. E. Henson, and S. McCormick, A Multigrid Tutorial, 2nd edition, Society for Industrial and Applied Mathematics, Philadelphia, 2000.
- A. Brandt, Multilevel adaptive solutions to boundary value problems, Math. Comp., 31 (1977), pp. 333–390.
- W. Hackbusch, Multi-Grid Methods and Applications, vol. 4 of Springer Series in Computational Mathematics, Springer-Verlag, New York, 1985.
- S. McCormick, Multilevel Adaptive Methods for Partial Differential Equations, SIAM, Philadelphia, 1989.

References II

- K. E. Atkinson, Iterative variants of the Nyström method for the numerical solution of integral equations, Numer. Math., 22 (1973), pp. 17–31.
- C. T. Kelley, A fast multilevel algorithm for integral equations, SIAM J. Numer. Anal., 32 (1995), pp. 501–513.
- C. T. Kelley, Multilevel source iteration accelerators for the linear transport equation in slab geometry, Trans. Th. Stat. Phys., 24 (1995), pp. 679–708.

MG for Integral Equations

Fredholm 2nd kind integral equation on $C[0,1]$,

$$u(x) = (Ku)(x) + g(x) = \int_0^1 k(x,y)u(y)\,dy + g(x).$$

Here

- k and g are given continuous functions.
- $u \in C[0,1]$ is the unknown.

We assume that $I - K$ is nonsingular.

Atkinson (73) - Brakhage (60) Method

Notation:

- Sequence of quadrature rules: nodes $\{x_j^m\}_{j=1}^{N_m}$ and weights $\{w_j^m\}_{j=1}^{N_m}$.
- Operators: $K_m(u)(x) = \sum_{j=1}^{N_m} k(x, x_j^m) u(x_j^m) w_j^m$
- Note that K_m is defined on $C[0,1]$.

The sequence $\{K_m\}$ is collectively compact and converges strongly to K.

Defining all operators on $C[0,1]$ is easier than thinking of sequences of problems and operators.

Solving $u - K_m u = f$

First solve the finite-dimensional system for the function values at the nodes

$$u(x_i^m) - \sum_{j=1}^{N_m} k(x_i^m, x_j^m) u(x_j^m) w_j^m = f(x_i^m)$$

(with GMRES, for example).
Then recover u with the Nyström interpolation

$$u(x) = f(x) + \sum_{j=1}^{N_m} k(x, x_j^m) u(x_j^m) w_j^m.$$

Some Facts from Functional Analysis

Given $\rho > 0$ there is l_0 such that if $L \geq l \geq l_0$ the operator

$$B_l^L = I + (I - K_l)^{-1} K_L$$

satisfies

$$\|I - B_l^L (I - K_L)\| \leq \rho.$$

NOTE!! $(I - K_l)^{-1}$ won't work! We will fix that by changing K_l.

Atkinson-Brakhage iteration

Solve $u - K_L u = g$ on a fine mesh by using B_l^L as a preconditioner to fixed point iteration.

$$u_+ = u_c - B_l^L(u_c - K_L u_c - g).$$

Cost: Two fine-mesh calls to K_L.
$r_c = g - (I - K_L)u_c$ and $B_l^L r_c = r_c + (I - K_l)^{-1} K_L r_c$.
Evaluate $(I - K_l)^{-1} w$ via GMRES.

Implementation

You have seen how to compute K_L. The last part is the action of B_l^L on a function u.

- Evaluate $K_L u$.
- Solve $w - K_l w = K_L u$ as above.

A More Efficient Way

- Suppose you have a compact operator K and
- can compute operator-function products.
- Suppose you haev a projection P_l onto a finite dimensional subspace V_l and
- P_l converges strongly to I.
- Then KP_L converges to K in the operator norm and
- $(I - KP_L)^{-1} \to (I - K)^{-1}$ in norm.

Faster Two-Grid Method

$$u_+ = u_c - (I - KP_l)^{-1}((I - K)u_c - g)$$

How to evaluate $(I - KP_l)^{-1}w$.

- Solve the finite dimensional problem

$$w_l - P_l K P_l w_l = P_l g$$

- Let

$$w = g - KP_l w_l.$$

Nested Iteration: $K_i = KP_i$; $K = K_L$

Algorithm $\texttt{gridnest}(g, u, \{K_i\}, l, L)$

Solve $u^l - \mathcal{K}_l u^l = g$ on the coarse level.

Set $u = u^l$.

for $m = l + 1, \dots L$ **do**

$u = u - (I - KP_l)^{-1}((I - K)u - g)$

end for

Remarks

- Analysis shows that $\|K_l - K_L\|$ is small for sufficiently large l and all $L > l$ (including $L = \infty$, the continuous problem.
- The algorithm can be realized by using piecewise linear interpolation, evaluation at grid points, and full weighting for the coarse-to-fine transfer.

Easy example

$$u(x) - \frac{1}{2}\int_0^1 \sin(x-y)u(y)\,dy = \cos(x)$$

I have already discretized this into the form $(I-K)u = f$, so the hard job is the intergrid transfers.

- Coarse to fine: simple interpolation.
- Fine to coarse: repeated applications of full weighting.
- And here's some MATLAB ...

Nested Iteration

Opimal approach

- Start on coarse grid l; solve to high accuracy.
- Interpolate to fine grid. Solve with multilevel method.
- Keep the coarse level at l for the entire solve!k
- You'll need one iteration/level to maintain accuracy to truncation error if the coarse mesh is fine enough.

Example: Source Iteration in Transport Theory

- Let $\phi - K_L \phi = S$ be the fully discrete problem at level L.
- Map from level L to level $l < L$ by full weighting.
- That map has the same properties as a continuous projection.
- Map from l to L by interpolation.
- So $K_l = I_L^l K_L I_l^L$ will do the job.

Multigrid

We showed that a Jacobi iteration for the central difference discretization $A^h u^h = f^h$ of

$$-u'' = f, 0 < x < 1$$

with $u(0) = u(1) = 0$

- damped the error in the high frequencies fast, and
- the error in the low frequencies very slowly,
- so the iteration is a **smoother**,

which explains the methods poor convergence properties.
MG exploits this by attacking the smooth components on coarser grids.

Discrete Laplacian I

Recall that

$$A^h = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & 0 & \dots & 0, & 0 \\ -1 & 2 & -1 & ,0 & \dots & 0 \\ 0 & -1 & 2 & -1, & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots, & ,0, & -1 & 2 & -1 \\ 0 & \dots, & \dots,, & 0 & -1 & 2 \end{pmatrix}$$

and

$$f^h = (f(x_1), \dots, f(x_N))^T.$$

Discrete Laplacian II

The eigenvectors of A^h are $\{u_n\}_{n=1}^N$ where

$$u_n = (\xi_1^n, \dots, \xi_N^n)^T \text{ and } \xi_i^n = \sqrt{2/h}\sin(ni\pi h),$$

with eigenvalues

$$\lambda_n = h^{-2} 2\left(1 - \cos(\pi n h)\right) = \pi^2 n^2 + O(h^2) \text{ for small } n$$

We proved this with elementary trigonometry.

A better smoother

Consider the **damped Jacobi** iteration:

$$x_{n+1} = (1-\omega)x_n - \omega D^{-1}(L+U)x_n + D^{-1}b$$

with iteration matrix $M^h_{DJ} = (1-\omega)I + \omega M_{JAC}$, where

$$M^h_{JAC} = -D^{-1}(L+U).$$

So $\omega = 1$ is Jacobi and $\omega = 0$ is nothing.
What's the optimal ω if we want to damp the high-frequency terms?

Eigen-analysis

We showed that the eigenvalues of M_{JAC} were

$$\mu_n = 1 - (h^2/2)\lambda_n$$

with the same eigenvectors as A^h. Similarly

$$\begin{aligned} \mu_n &= (1-\omega) + \omega(1 - (h^2/2)\lambda_n) = 1 - (h^2/2)\omega\lambda_n \\ & 1 - \omega(1 - \cos(\pi n h)) \end{aligned}$$

Optimal ω

Now suppose $n \geq N/2$ (high frequency), then

$$\mu_n = 1 - \omega(1 - \cos(\pi nh)) = 1 - \omega + \omega \cos(\pi nh)$$

so

$$1 - 2\omega \leq \mu_n \leq 1 - \omega.$$

Minimize $|\mu_n|$ to see that the optimal value of ω is $2/3$. So

$$|\mu_n| \leq 1/3 \text{ for } N/2 \leq n \leq N - 1$$

Idealized Two-Grid Method: I

Notation:

- $h = 1/(N-1)$, $N-1$ even.
- Ω^h: space of grid functions with step size h so $u^h \in \Omega^h$
- $I_h^{h'}$: intergrid transfer from $\Omega^h \rightarrow \Omega^{h'}$
- P_H^h and P_L^h project onto low and high frequencies.

Idealized Two-Grid Method: II

For Now: We will define I_h^{2h} and I_{2h}^h by Fourier truncation.

$$\begin{aligned} u &= \sum_{n=1}^{N-1} \alpha_n u_n^h \in \Omega^h \\ I_h^{2h} u &= \sum_{n=1}^{(N-1)/2} \alpha_n u_n^{2h} \in \Omega^{2h} = I_h^{2h} P_L^h u \end{aligned}$$

and

$$\begin{aligned} w &= \sum_{n=1}^{(N-1)/2} \alpha_n u_n^{2h} \in \Omega^{2h} \\ I_{2h}^h w &= \sum_{n=1}^{(N-1)/2} \alpha_n u_n^h \in \Omega^h \end{aligned}$$

Note: $I_h^{2h} P_L A^u = A^{2h} I_h^{2h} P_L u$ and $I_{2h}^h I_h^{2h} = P_L$

Idealized Two-Grid Method: III

Simple method:

- Let $u_c \in \Omega^h$
- Take one weighted Jacobi iteration on Ω^h to obtain $u_{1/2}$
- Compute the residual $r^h = f^h - A^h u_{1/2}$
- Compute the **coarse grid correction** $w = (A^{2h})^{-1} I_h^{2h} r^h$
- $u_+ = u_{1/2} + I_{2h}^h w$

Idealized Two-Grid Method: IV

If we did everything on Ω^h, then

$$w = A^{-1} r = A^{-1}(f^h - A^h u) = A^{-1} f^h - u = u^h - u$$

so

$$u + w = u^h$$

solves the problem. We will show that the two grid method does pretty well.

Idealized Two-Grid Method: V

Apply weighted Jacobi and let $e = u^h - u$.

$$\|P_H e_{1/2}\| \leq (1/3)\|P_H e_0\|$$

so

$$\begin{aligned} r^h &= f^h - A^h u_{1/2} = P_H(f^h - A^h u_{1/2}) + P_L(f^h - A^h u_{1/2}) \\ &= A^h P_H e_{1/2} + P_L A^h e_{1/2}. \end{aligned}$$

We are just about done because ...

Idealized Two-Grid Method: VI

The coarse grid correction eliminates the low frequency errors.

- $I_h^{2h} P_L A^h e_{1/2} = A^{2h} P_L e_{1/2}$, so
- $(A^{2h})^{-1} I_h^{2h} P_L A^h e_{1/2} = I_h^{2h} P_L e_{1/2}$,
- and $w = I_{2h}^h I_h^{2h} P_L e_{1/2} = P_L e_{1/2}$.

Hence

$$e_+ = e_{1/2} - P_L e_{1/2} = P_H e_{1/2}$$

and hence $\|e_+\| \leq (1/3)\|e_c\|$

Algorithmic Description

$\texttt{two\_grid}(h, u_c^h, u_+^h, A, f)$

Smooth once to obtain $u_{1/2}^h$ from u_c^h

$r^h = f^h - A^h u_{1/2}^h$

Solve $A^{2h} w = I_h^{2h} r^h$ exactly.

$u_+^h = u_{1/2}^h + I_{2h}^h w$

Algorithmic Description: MG

Multigrid replaces the coarse mesh solve with two-grid iteration.
h: desired (finest) grid; H: coarsest grid.

$\texttt{multi\_grid}(h, H, u_c^h, u_+^h, A, f)$

if $h = H$ **then**

Solve $A^h u^h = f^h$ exactly.

else

Smooth once to obtain $u_{1/2}^h$ from u_c^h

$r^h = f^h - A^h u_{1/2}^h$

$\texttt{multi\_grid}(2h, H, 0, w, A, I_h^{2h} r^h)$

$u_+^h = u_{1/2}^h + I_{2h}^h w$

end if

This is a V-cycle. The convergence rate remains unchanged.

Observations

- MG is faster that two-grid because the exact solves live only on the coarsest mesh.
- This method only works for the simplest problem where
  - we can connect the eigenfunctions of A to those of M
  - the intergrid transfers have the same eigenfunctions
  - $I_{2h}^{h} I_{h}^{2h} = P_{L}^{h}$
- In general, we have none of those things.

Realistic Intergrid Transfers

- Nested grids: $x_i^h = ih$; $x_{2i}^h = x_i^{2h}$
- I_{2h}^h: linear interpolation;

$$(I_{2h}^h v^{2h})_{2i} = v_i^{2h} \text{ and } (I_{2h}^h v^{2h})_{2i-1} = (v_i^{2h} + v_{i-1}^{2h})/2.$$

- I_h^{2h}: full weighting

$$(I_h^{2h} v^h)_i = (v_{2i-i}^h + 2v_{2i}^h + v_{2i+1}^h)/4.$$

except on the boundary.

Matrix Representation: I

- $N = 2^p - 1$ internal grid points
- Internal grid points: $\{x_i^h\}_{i=1}^N$; $h = 1/(N+1)$
- Boundary grid points: $x_0^h = 0$, $x_{N+1}^h = 1$
- I_{2h}^h is $1 + (N-1)/2 \times N$
- Other option: restriction by injection: $(I_h^{2h} v)_i = v_{2i}$

Matrix Representation: II

$$I_{2h}^{h} = \frac{1}{2} \begin{pmatrix} 1 & 0 & \dots & 0 & 0 & 0 \\ 2 & 0 & \dots & 0 & 0 & 0 \\ 1 & 1 & \dots & 0 & 0 & 0 \\ 0 & 2 & \dots & 0 & 0 & 0 \\ 0 & 1 & 1 & \dots & 0 & 0 \\ 0 & 0 & 2 & \dots & 0 & 0 \\ 0 & 0 & 1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 1 \\ 0 & 0 & 0 & \dots & 0 & 2 \\ 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix}, \; I_{h}^{2h} = \frac{1}{2}(I_{2h}^{h})^{T}.$$

Comments

- Interpolatory intergrid transfers have aliasing effects.
- You have to smooth an extra time.
- But you get the convergence rate $1/9$ instead of $1/3$ you'd expect.

Intergrid Transfers for 1-D: I_{2h}^{h}

$w = I_{2h}^{h}(u)$

$N = h^{-1} - 1$; $h_2 = 2h$; $N_2 = h_2^{-1} - 1$

for $i = 0 : N_2$ **do**

$w_{2i} = u_i$

$w_{2i+1} = (u_i + u_{i+1})/2$

end for

Intergrid Transfers for 1-D: I_h^{2h}

$w = I_h^{2h}(u)$

$N = h^{-1} - 1$; $h_2 = 2h$; $N_2 = h_2^{-1} - 1$

$w_0 = u_0$; $w_{N_2+1} = u_{N+1}$

for $i = 1 : N_2$ **do**

$w_i = (u_{2i-1} + 2u_{2i} + u_{2i+1})/4$

end for

Smoother

Notation: Smooth ν times at level h with initial iterate u^h

$u^h \leftarrow S(u^h, f^h, \nu)$

for $is = 1 : \nu$ **do**
  for $i = 1 : N$ **do**
    $w_i = h^2(f_i^h + u_{i-1}^h + u_{i+1}^h)/2$
  end for
  for $i = 1 : N$ **do**
    $u_i^h = (1 - \omega)u_i^h + \omega w_i$
  end for
end for

The V-cycle

$u^h \leftarrow V^h(u^h, f^h)$

if $h = H$ **then**

Solve $A^h u^h = f^h$ to high accuracy.

else

$u^h \leftarrow S(u^j, f^h, \nu_1)$

Compute $f^{2h} = I_h^{2h}(f^h - A^h u^h)$

$u^{2h} = 0$

$u^{2h} \leftarrow V^{2h}(v^{2h}, f^{2h})$

$u^h \leftarrow u^h + I_{2h}^h u^{2h}$

$u^h \leftarrow S(u^h, f^h, \nu_2)$

end if

Comments

- Typical choice $\nu_1 = \nu_2 = 1$.
- V-cycle for high-order term is a good preconditioner for PDEs.
- V-cycle reduces the error by fixed amount independent of h.
- We consider very easy problems and simple grids
- Solve step for Ω^H can be iterative.
- There's more, much more
  - Other smoothers (Gauss-Seidel, ...)
  - Algebraic multigrid (AMG)
  - MG for Navier-Stokes, combustion ...
  - Non isotropic flows/grids

Matlab Example: Coarse Mesh Solve

You can avoid the exact solve on the coarse mesh with no loss. This example shows that

- Convergence rate is independent of h
- You can simply do a few (24) smoothing steps at the coarse level
- You are not restricted to Poisson's equation

Helmholtz in 1-D

$$u'' + \sigma u = f; u(0) = u(1) = 1.$$

Arrange things so the exact solution is

$$u(x) = e^x \sin(\pi x).$$

Comparisons from Coarse_Solve_Test directory

- Several fine levels: $h = 2^{-l}$, $5 \leq l \leq 10$
- Exact solve vs 24 smooths on Coarse mesh.

Nested Iteration or Full Multigrid (FMG): $h = 2^{-p}H$

Solve $A^H u^H = f^H$ on Ω^H; $h = H$.
for $il = 1 : p$ **do**
  $h = h/2$
  $u^h = I_{2h}^h u^{2h}$
  $u^h \leftarrow V^h(u^h, f^h)$
end for

Data Structures

You do not want to repeatedly allocate storage for the solutions on the various grids. You can avoid this by creating a large structure which preallcoates the storage.
Some of the examples in the **FMG_Test** do this.

Comments

- Optimal complexity (see next slide)
- Complexity leads to debugging tool
- FMG is a **solver**. If error is $O(h^2)$ then one V-cycle per level will suffice.
- A V-cycle can be a **preconditioner**

Complexity Example: I

- A^h discrete Laplacian in R, N interior grid points
- Cost of smoother, mat-vec, intergrid transfer $= 3N$
- Cost or residual, correction $= N$
- Cost of solve at level H is V_0
- Then, cost of V-cycle at level $h_j = 2^{-j} H$ is

$$V_j = 3(N_j(\nu_1 + \nu_2) + 2N_j) + 2N_j + V_{j-1}$$

where $N_j = (1/h_j) - 1 \approx 2N_{j-1}$.

Complexity Example: II

So, if $\nu_1 = \nu_2 = 1$, then

$$\begin{aligned} V_j &= 14N_j + V_{j-1} = 14\sum_{l=1}^{j} N_l + V_0 \\ &\leq 28N_j + V_0 \end{aligned}$$

Cost of FMG for $h = 2^{-p}H$; $N = N_p$

$$\begin{aligned} FMG_i &\leq \sum_{l=0}^{p} N_l + V_l \leq \sum_{l=0}^{p} 29N_l + pV_0 \\ &\leq 58N + pV_0 = O(N). \end{aligned}$$

Testing Complexity

In one dimension, you should see the computing time double if $h \rightarrow h/2$.

In practice, MATLAB-sized problems take so little time that you will find this hard to measure.

Exercises

- Solve the transport equation with mutligrid in space using the modified Atkinson-Brakhage method.
- Write a MG V-cycle for the 2-D Poisson Equation. Test it as both a solver and a preconditioner within the `klpde2ddemo.m`.
- How does the performance of your V-cycle preconditioner for the convection-diffusion problem in `klpde2ddemo.m` change if you use two V-cycles instead of one?