Lecture 20: Multigrid

David Bindel

7 Apr 2010

Logistics

- HW 3 due date: 4/19 (Monday)
  - Feel free to turn it in early
  - ... and work on your projects!
- HW 3 comments
  - In OpenMP, fork/join is not free!
  - Helps to associate data to processors (see prompt)
- Projects: please talk to me!
- Guest lecture on Weds, 4/21 (Charlie Van Loan)

Multilevel idea

Basic idea (many contexts)

- Coarsen
- Solve coarse problem
- Interpolate (and possibly refine)

May apply recursively.

Plan of attack

For today:

- Explain why it works in a couple ways for a simple case.
- Give some ideas for more complicated cases.
- Talk about parallelism.

Will introduce some fun ideas along the way.

Reminder: 1D Poisson

Continuous: $-u'' = f,\ u(0) = u(1) = 0$
Discrete: $Tu = h^2 f,\ u_0 = u_N = 0,$

$$T = \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix}$$

Simple iterative solvers move information one point per iteration. Therefore, convergence takes at least $O(N)$ iterations.

Jacobi on 1D Poisson

Jacobi for Poisson:

$$Mu^{(i+1)} + (T - M)u^{(i)} = h^2 f$$

where $M = 2I$ is the diagonal part of T.

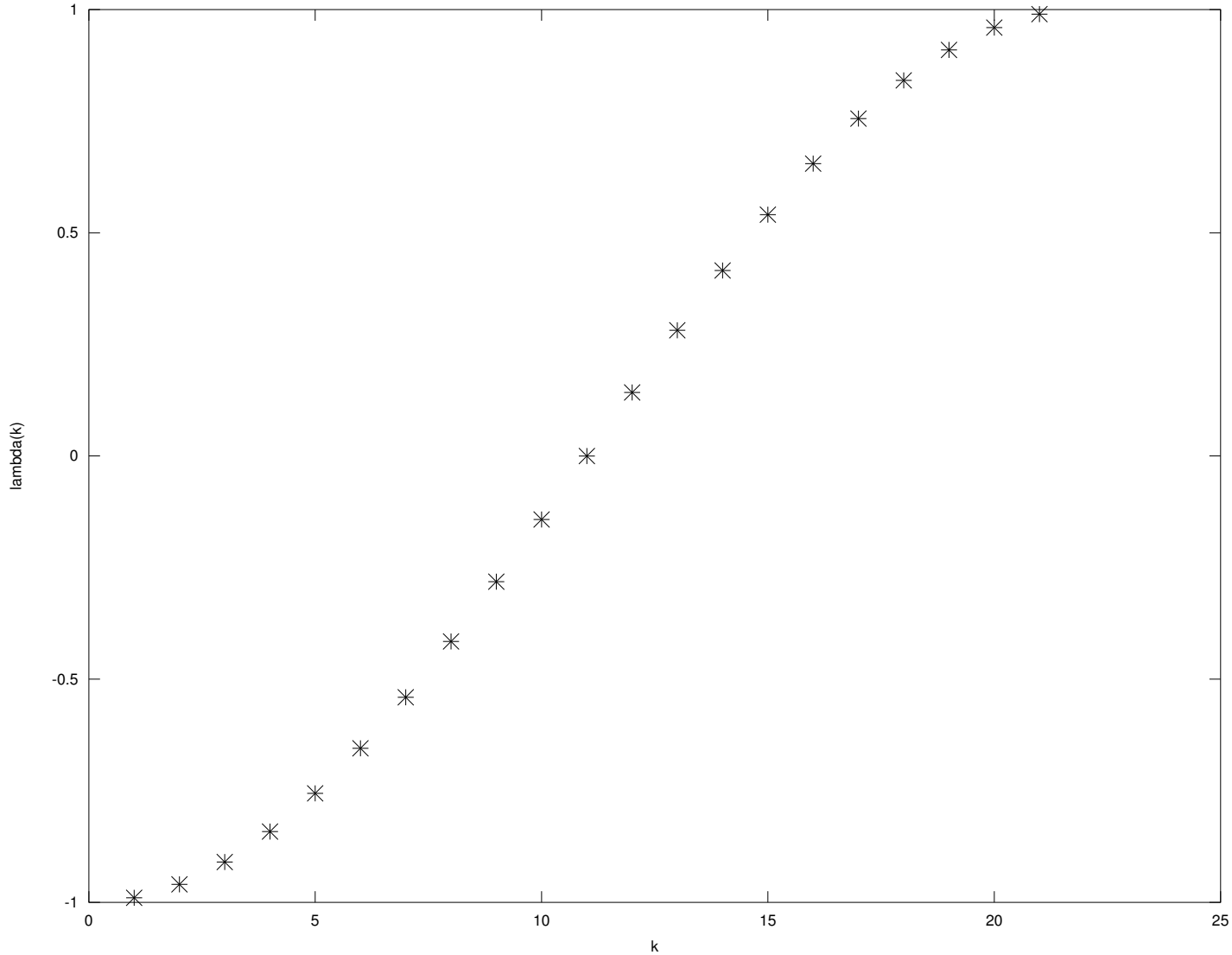
Let $e^{(i)} = u^{(i)} - u$ be the error at step i. Then

$$e^{(i+1)} = M^{-1}(T - M)e^{(i)} = \left(\frac{1}{2}T - I\right) e^{(i)}.$$

Or write $\hat{e}^{(i+1)} = \Lambda \hat{e}^{(i)}$ where $\hat{e}^{(i)} = Ze^{(i)}$ is discrete Fourier transform, Λ is eigenvalues of $T/2 - I$.

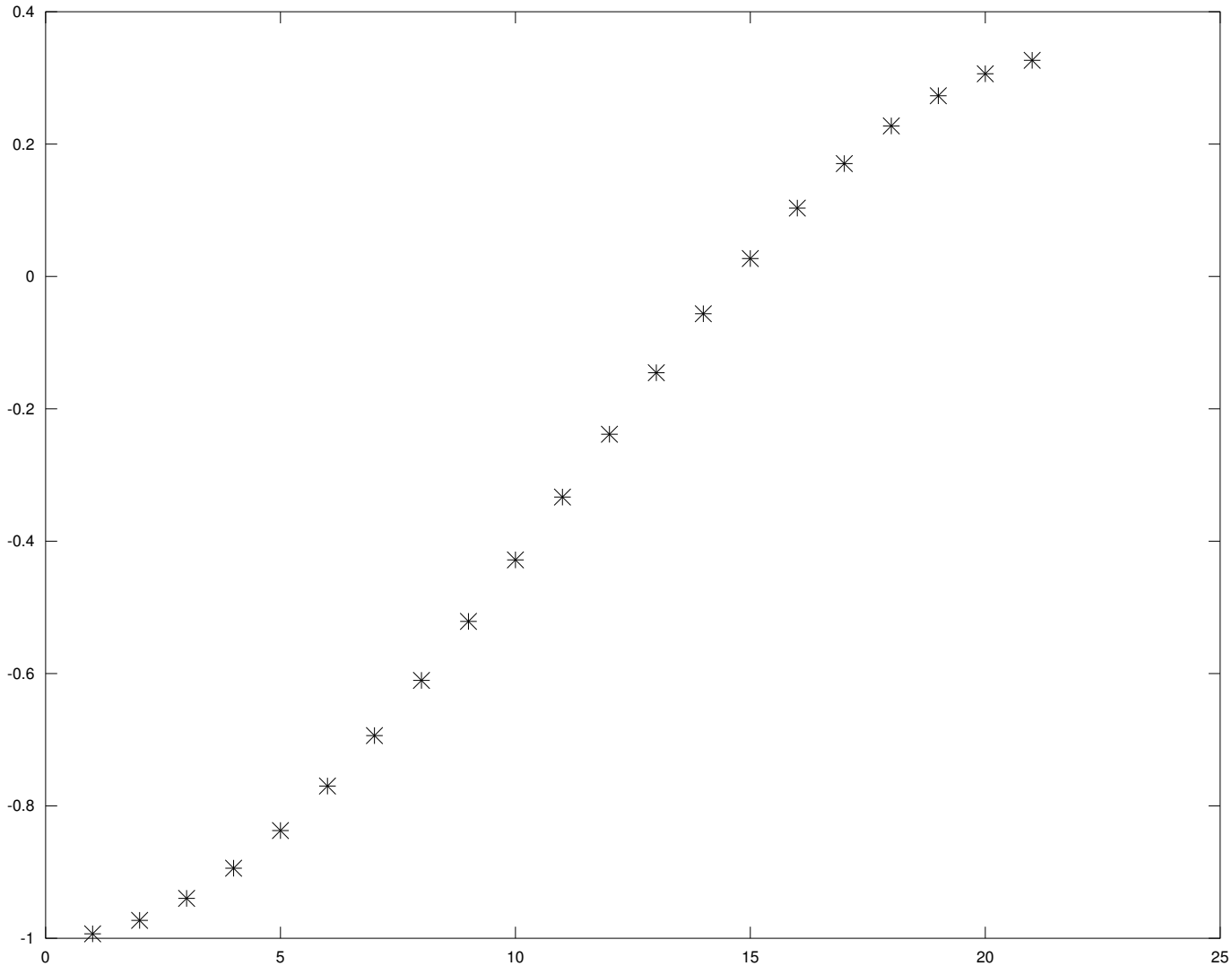
Spectral view

Eigenvalues of $M^{-1}(T - M)$, $M = 2I$



Spectral view of damped Jacobi

Eigenvalues of $M^{-1}(T - M)$, $M = 2\omega I$, $\omega = 2/3$



Jacobi and damped Jacobi

- Jacobi: largest and smallest eigenvalues are size $1 - O(h^2)$.
- ... so we don't damp high and low frequency error by much
- Damped Jacobi damps high frequencies, but not low
- How do we get the low frequencies?

Coarse grid approximation

Another idea: just use fewer discretization points!

- Take weighted average of nearby points to get coarse f_c
- Solve for coarse solution u_c
- Interpolate coarse u_c to original mesh ($\hat{u}$)

Mathematically, get

$$f_c = P^T f, \quad u_c = T_c^{-1}(h_c^2 f_c), \quad \hat{u} = P u_c$$

where

$$P = \begin{bmatrix} \frac{1}{2} & & \\ 1 & & \\ \frac{1}{2} & \frac{1}{2} & \\ & 1 & \\ & \frac{1}{2} & \frac{1}{2} \\ & & 1 \\ & & \ddots \end{bmatrix}$$

Coarse grid correction

Given an approximate solution $\tilde{u}$, compute the correction

$$\tilde{u}^{(1)} = \tilde{u} + PT_c^{-1}P^T(h^2 f - T\tilde{u})$$

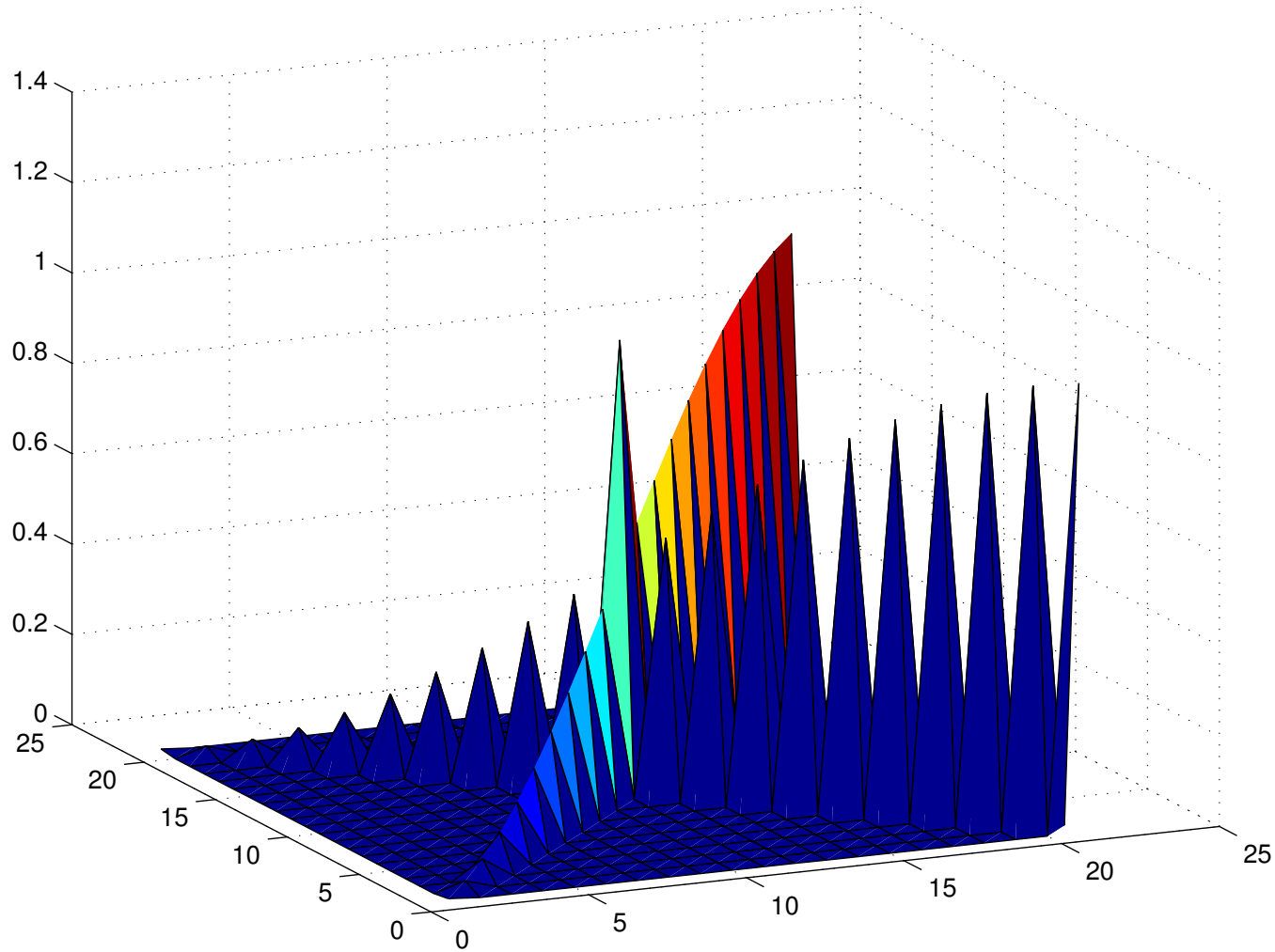
Corresponds to

$$\tilde{e}^{(1)} = (I - PT_c^{-1}P^T T)\tilde{e}$$

Let's consider what $(I - PT_c^{-1}P^T T)$ looks like in DFT basis...

Effect of coarse grid correction

Fourier error in coarse grid: $\hat{e} = Z(T - PT_c^{-1}P^T)Z^T f$.
Component magnitudes look like:



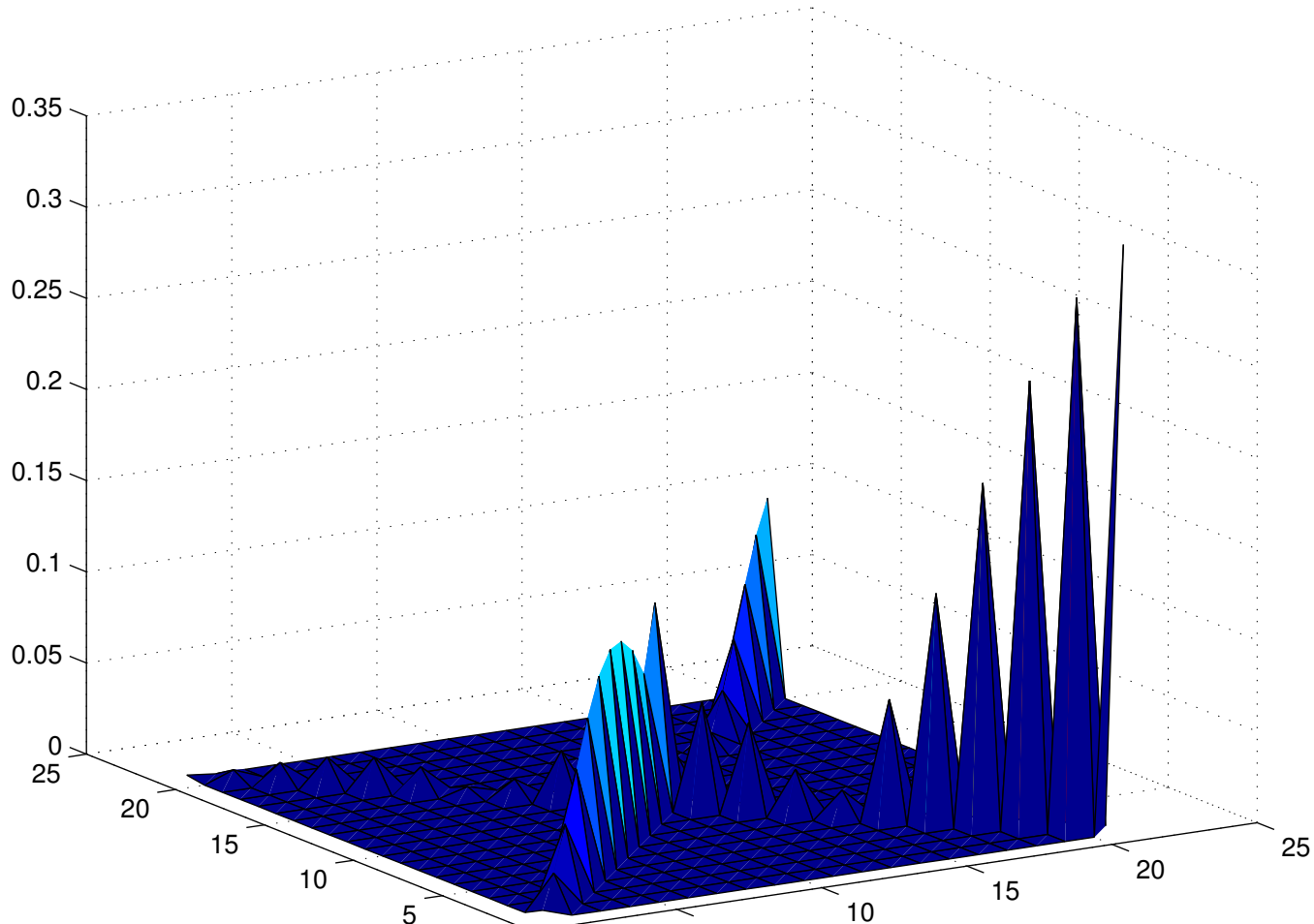
Putting it together

Two complimentary solvers:

- Coarse grid correction reduces *low* frequencies in error
- Weighted Jacobi damps out *high* frequencies in error

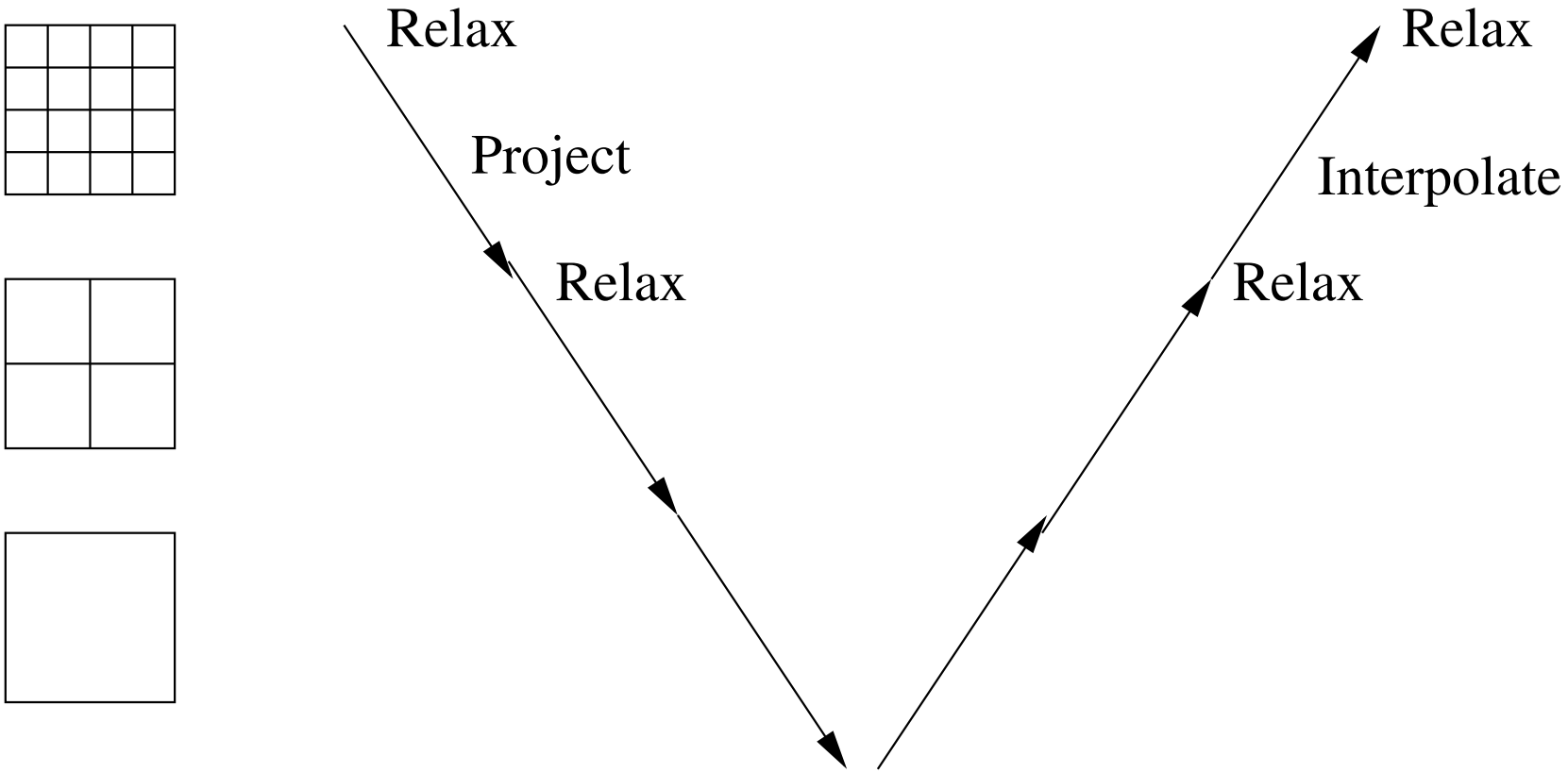
Suggests we alternate the two in order to reduce overall error.

Weighted Jacobi + coarse correction + weighted Jacobi yields:



Multigrid V-cycle

Need to do a solve for coarse-grid correction — recurse!
Each grid damps a part of the frequency error.



Reduces error by a factor *independent of n*.

Multigrid V

One V-cycle on 1D Jacobi:

$$\sum_{i=1}^{\log_2 N} O(2^i) = O(N)$$

And most work goes into (very parallel) operations on fine grids! Picture gets even better in 2D, 3D.

FFT-based methods may still be faster in practice, though, even if asymptotically slower.

Ingredients

To apply multigrid, we need:

- A *restriction* that maps from fine to coarse
- An *interpolation* that maps from coarse to fine
- A smoother for each grid

and usually a full solver for the coarsest grid. Also need a strategy (V cycles, W cycles, full multigrid, cascadic multigrid?)

Also, goes great with Krylov subspaces (MG as preconditioner or CG as smoother – probably not both).

Beyond Poisson on a brick

Same ingredients for more general problems!

- Adaptive meshes
- Irregular meshes
- More complicated PDEs
- Nonlinear problems (for Newton step or fully nonlinear)

Gotchas

Still requires thought:

- How do we choose the coarse meshes?
- How do we choose the prolongation/interpolation?
- How do we choose the smoother?
- Where do we stop the recursion (maybe with a direct solve)?
- What about features that a coarse solver might miss?
  - Anisotropic materials and meshes
  - Material discontinuities
  - Indefiniteness and pollution error

Attempted semi-automatic solution: algebraic multigrid.

Algebraic multigrid

Idea: automatically construct coarse meshes from matrix

- Use matching algorithm to figure out nodes to merge (as with multilevel graph partitioning!)
- May still want node positions (null space issues)

Example code: ML from Sandia.

Multilevel domain decomposition

Additive Schwarz preconditioner with n overlapping subdomains looks like

$$M = \sum_{j=1}^{n} P_j T_j^{-1} P_j^T.$$

If lots of subdomains, it again takes info a while to propagate...

... so add a coarse grid correction!

$$M_{ML} = P_C T_C^{-1} P_C^T + \sum_{j=1}^{n} P_j T_j^{-1} P_j^T.$$

This is another way to scalable algorithms.