CS 294-73
Software Engineering for Scientific Computing

Lecture 16: Multigrid
(structured grids revisited)

Guest lecturer: Hans Johansen, hjohansen@lbl.gov
Laplacian, Poisson’s equation, Heat Equation

Laplacian:

\[ \Delta \phi = \sum_{d=0}^{D-1} \frac{\partial^2 \phi}{\partial x_d^2}, \quad \phi(\mathbf{x}) = \phi : \Omega \to \mathbb{R} \]
\[ \mathbf{x} = (x_0, \ldots x_{D-1}) \]

Poisson’s equation:

\[ -\Delta \phi = f, \quad f : \Omega \to \mathbb{R}, \quad \phi = 0 \text{ on } \partial \Omega \]

Heat equation:

\[ \frac{\partial \phi}{\partial t} = \Delta \phi + f \]
\[ \phi(\mathbf{x}, t), f(\mathbf{x}, t) = \phi, f : \Omega \times [0, T] \to \mathbb{R} \]
\[ \phi(\cdot, t) = 0 \text{ on } \partial \Omega \]
Discretizing on a rectangle (or a union of rectangles)

\[
\frac{\partial^2 \phi}{\partial x_d^2} \bigg|_{ih} = \frac{1}{h^2} (\phi_{i+e^d} - 2\phi_i + \phi_{i-e^d}) + O(h^2)
\]

\(i = (i_0, \ldots, i_{D-1})\)

\[
\Delta \phi \bigg|_{ih} \approx (\Delta^h \phi^h)_i = \frac{1}{h^2} \left( -2D\phi_i + \sum_{d=0}^{D-1} (\phi_{i+e^d} + \phi_{i-e^d}) \right)
\]

Defined only on interior (blue) nodes. Values at black nodes are set to zero if we are using the boundary conditions given above.
Solving Poisson’s equation

Want to solve linear system of equations

\[-(\Delta^h \phi^h)_i = f_i\]

For \(\phi\) defined on the interior (blue) grid points \(i\). This makes sense, since the stencil for the operator requires only nearest neighbors.

\[
\Delta \phi \bigg|_{ih} \approx (\Delta^h \phi^h)_i = \frac{1}{h^2} \left( -2D\phi_i + \sum_{d=0}^{D-1} (\phi_{i+e^d} + \phi_{i-e^d}) \right)
\]

and we have values defined by the boundary conditions to be set to zero (black points).
Point Jacobi

As in the unstructured grid case, we can use point Jacobi to solve these equations.

\[ \phi_i^{h,l+1} = \phi_i^{h,l} + \lambda((\Delta^h \phi_i^{h,l}) + f_i) \]

where \( \lambda \) is the relaxation parameter for our iterative scheme.

Each iteration corrects \( k^{th} \) eigenmode of \( \Delta^h \phi^h \) at different rate:

\[ O(N^2) \] operations to get this far …
Smoothing properties of point Jacobi

Define the error

\[ \delta_i^l \equiv \phi_i^{h,l} - \phi_i^h \]

\[ (\Delta^h \delta^l)_i = R_i^l, \quad R^l = \Delta^h \phi^{h,l}_i + f^h \]

Even though we don’t know the error, we can compute the residual to provide a measure for the error: e.g. convergence if \( ||R^l|| \leq \epsilon ||f|| \).

We can also see how the error behaves under point Jacobi

\[ \phi_i^{h,l+1} = \phi_i^{h,l} + \lambda((\Delta^h \phi^{h,l})_i + f_i) \]

\[ \delta_i^{l+1} = \delta_i^l + \lambda(\Delta^h \delta^l) \]
Smoothing properties of point Jacobi

For example, choose $\lambda = \frac{h^2}{4D}$. Then

$$\delta^{l+1} = \delta^l + \lambda(\Delta^h \delta^l)$$

$$(\Delta^h \delta^h)_i = \frac{1}{h^2} \left( -2D\delta_i + \sum_{d=0}^{D-1} (\delta_{i+e^d} + \delta_{i-e^d}) \right)$$

$$\delta_i^{l+1} = \frac{1}{2} (\delta_i^l + \frac{1}{2D} \sum_{d=0}^{D-1} (\delta_i^{l+e^d} + \delta_i^{l-e^d}))$$
Smoothing properties of point Jacobi

\[ \delta_{i}^{l+1} = \frac{1}{2} (\delta_{i}^{l} + \frac{1}{2D} \sum_{d=0}^{D-1} (\delta_{i+e^{d}}^{l} + \delta_{i-e^{d}}^{l})) \]

- The value at the new iteration is an average of the values at the old iteration: weighted sum, positive weights that sum to 1.
- The max and min values are always strictly decreasing. Smoothes the local error very rapidly.
- Error equation looks like forward Euler for the heat equation – smooths higher wavenumbers faster.
Point Jacobi - smoothing example

N=128, k=1, for 128 iterations:
64x the work, but worse error!

What about other modes?

k=2,4,8 for 128 iterations:
each closer to final solution.

Point Jacobi for Poisson
requires \( O(N^{D+2}) \) work
for a specified level of error!
(and leaves “smooth” error)

Why?
Discrete Fourier analysis …
Assume periodic bc’s for Poisson, and recall discrete Fourier modes:

\[
f_j = \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \langle f, W^{(k)} \rangle W_j^{(k)} = \frac{1}{N} \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \mathcal{F}_N(f)_k \omega^{kj} = \frac{1}{N} \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \mathcal{F}_N(f)_k e^{2\pi ikjh}
\]

And finite difference operators are diagonalized: \( W_{j+1}^{k} = z^k W_j^{k} \)

\[
L(f)_j = \frac{f_{j+1} - 2f_j + f_{j-1}}{h^2}
\]

\[
= \frac{1}{N} \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \Lambda(k) \mathcal{F}_N(f)_k e^{2\pi ikjh} \quad \Lambda(k) = \frac{(z^k - 2 + z^{-k})}{h^2}
\]

\[
L^{-1}(f)_j = \frac{1}{N} \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \frac{1}{\Lambda(k)} \mathcal{F}_N(f)_k e^{2\pi ikjh}
\]
So what does Point Jacobi do in Fourier space?

$$\delta^{l+1} = \delta^l + \lambda \Delta^h \delta^l$$

$$\Rightarrow \mathcal{F}_k(\delta^{l+1}) = \mathcal{F}_k(\delta^l) + \lambda \Lambda(k) \mathcal{F}_k(\delta^l)$$

$$= (1 + \lambda \Lambda(k)) \mathcal{F}_k(\delta^l)$$

$$= \omega(k) \mathcal{F}_k(\delta^l)$$

Where each error mode’s damping factor $\omega(k)$ is:

$$\omega(k) = 1 + \lambda \Lambda(k)$$

$$= 1 + \frac{\lambda}{h^2} (z^k - 2 + z^{-k})$$

$$= 1 + \frac{2\lambda}{h^2} (\cos(2\pi kh) - 1)$$
Discrete Fourier Analysis of Point Jacobi

What does this look like for $\beta = 2\pi kh$, $k \in \left[0, \frac{N}{2}\right]$ (real spectrum)?

No choice helps lowest $k$ modes

$\lambda = 1/8$ (meh)

highest $k$ modes completely damped

$\lambda = 1/4$

$\omega(\beta) = 1 + 2\lambda \left(\cos(\beta) - 1\right)$

$\lambda = 1/3$

most high-$k$ modes damped

$\lambda = 1/2$

$+/-$ modes undamped!
Point Jacobi Fourier Analysis: Conclusions

• A few iterations of point Jacobi can be tuned to reduce the “high frequency” error with very just a few sweeps.

• These modes can be fixed *locally* – across a small number of points – while the “smooth” error persists *globally*.

• As resolution decreases, $N$ increases, and smooth errors ($j=1$) are less and less responsive to point Jacobi:

$$\omega(1) = 1 + \frac{2\lambda}{h^2} (\cos(2\pi h) - 1) \quad \text{max}(\lambda) = \frac{h^2}{2}$$

$$\approx 1 - C h^2$$

• Ideally, we’d use PJ for those modes separately with a coarse $N_c, h_c$ and $\lambda_c$ with good convergence rate and is stable on coarse grid …

• Then somehow blend the answers? $\implies$ **Multigrid!**
Multigrid

- Do a few iterations of point Jacobi on your grid to obtain $\tilde{\phi}^h$.
- Compute the residual $R^h = \Delta^h \tilde{\phi}^h + f^h$.
- Average the residual down onto a grid coarsened by a factor of 2: $R^{2h} = A(R^h)$.
- Apply point Jacobi to solve the residual equation for $\delta^{2h}$:
  $$\delta^{2h,l+1} = \delta^{2h,l} + \lambda (\Delta^{2h} \delta^{2h,l} + R^{2h})$$
- Interpolate correction back onto fine grid solution:
  $$\tilde{\phi}^+ = \mathcal{I}(\tilde{\delta}^{2h})$$
- Smooth again using point Jacobi.
**Multigrid**

- If the number of grid points is $2^M+1$, can apply this recursively.

```plaintext
multigrid(phi,f,h,level)
phi := phi + lambda*(L(phi) - f);
(p times)
if (level > 0)
   R = L(phi) - f;
   Rcoarse = Av(R);
   delta = 0.;
   call multigrid(delta,Rcoarse,2h, level-1);
   phi += Int(delta);
endif;
phi := phi + lambda*(L(phi) - f);
(p times)
```
Averaging and Interpolation

- Conservation of total charge.
  \[ \sum_{i_c} (2h)^D A(f)_{i_c} = \sum_i h^D f_i \]

- Adjoint & order conditions, other considerations (see Trottenberg, et al, *Multigrid*).

- For our second-order accurate discretization on a nodal-point grid, we can use the trapezoidal rule
  \[ A(f)_{i/2} = A^0 \circ A^1 \circ \ldots \circ A^{D-1}(f)_i \]
  \[ A^d(g)_i = \frac{1}{4}(g_{i+e^d} + 2g_i + g_{i-e^d}) \]
  (and bilinear interpolation).

Even if the grid is not a power of 2, can apply a direct solve at the bottom. 3 levels in 3D leads to a reduction by 512 in the number of unknowns.
Multigrid Convergence: 1D example

N=128, RHS step functions

N=128, exact solution
Multigrid Convergence: 1D example

\[ N=128, \text{ Point Jacobi only} \]

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Max Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N )</td>
<td>( \sim 0.9 )</td>
</tr>
<tr>
<td>( 8 \times N )</td>
<td>( \sim 0.45 )</td>
</tr>
<tr>
<td>( N^2 )</td>
<td>( \sim 2 \times 10^{-6} )</td>
</tr>
</tbody>
</table>

\[ \text{Coarse solution only} \]

<table>
<thead>
<tr>
<th>Depth</th>
<th>Max Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \sim 9 \times 10^{-4} )</td>
</tr>
<tr>
<td>2</td>
<td>( \sim 4 \times 10^{-3} )</td>
</tr>
<tr>
<td>3</td>
<td>( \sim 2 \times 10^{-2} )</td>
</tr>
<tr>
<td>4</td>
<td>( \sim 6 \times 10^{-2} )</td>
</tr>
<tr>
<td>5</td>
<td>( \sim 0.25 )</td>
</tr>
</tbody>
</table>
What’s the error from a coarse grid solution? *Look at Fourier modes* …

- Solution has lots of high wave-number content
What’s the error from a coarse grid solution? *Look at Fourier modes* …

- Solution has high-$k$ (wave number) content
- Solution on 1st coarse level matches low-$k$ well, but has high-$k$ error (piecewise linear interp.)
Multigrid Convergence: 1D example

What’s the error from a coarse grid solution? *Look at Fourier modes* …

- Solution has high-$k$ (wave number) content
- Solution on 1st coarse level matches low-$k$ well, but has high-$k$ error (piecewise linear interp.)
- Solution on 2nd coarse level has mid-$k$ error too (PJ less effective)
Multigrid Convergence: 1D example

What’s the error from a coarse grid solution? *Look at Fourier modes* …

- Solution has high-\(k\) (wave number) content
- Solution on 1\(^{\text{st}}\) coarse level matches low-\(k\) well, but has high-\(k\) error (piecewise linear interp.)
- Solution on 2\(^{\text{nd}}\) coarse level has mid-\(k\) error too (PJ less effective)
- Solution on 3\(^{\text{rd}}\) coarse level has errors in most \(k\) (solving wrong problem)
Multigrid Convergence: 1D example

What’s the error from a coarse grid solution? *Look at Fourier modes* …

- Solution has high-$k$ (wave number) content
- Solution on 1st coarse level matches low-$k$ well, but has high-$k$ error (piecewise linear interp.)
- Solution on 2nd coarse level has mid-$k$ error too (PJ less effective)
- Solution on 3rd coarse level has errors in most $k$ (solving wrong problem)
- *MG V-cycle gets all-$k$*
Multigrid Convergence: 1D example

How much work was it?
- \( O(N) \) PJ on each level, \( O(N) \) cost to average/interpolate \( \rightarrow O(N) \)
- Each coarser level is 1/2 the work \( \rightarrow O(N/2) + O(N/4) + O(N/8) \ldots \)
  \( \rightarrow \) Overall \( O(N) \) – *compared to \( O(N^3) \) for point Jacobi alone!*
- NOTE: ideal convergence for model problem, real world is harder!

\( N=128 * \text{and} * 2048 \)
- \( N=4 \) coarsest grid
- 2 Point Jacobi before \( \mathcal{A} \) (to reduce aliasing)
- 2 Point Jacobi after \( \mathcal{I} \) (to reduce high-\( k \) error)
  \( \rightarrow \) Each MG V-cycle reduces error by \( \sim 12x \)
Multigrid Analysis: Conclusions for 1D example

- Coarse grid correction is close to fine solution for low-$k$, but has high-$k$ content from interpolation operator.
- Coarsest point relaxation reduces low-$k$ wave number errors.
- The recursive algorithm reduces all wave numbers in $O(N)$ ops: (relax – coarsen – (do coarse) – interpolate – relax)

MG approach is problem-dependent … 1.2-20x reductions typical:

- Choice of point relaxation: some more effective than others, depending on the operator (discontinuous coefficients, etc.).
- Average and interpolation operators should represent smooth components of fine residuals/coarse solutions well (accuracy), with errors in the “near null space” of point relaxation scheme.
Results for real world examples
Finite-Volume Methods on Structured Grids

We write the Laplacian as the divergence of fluxes.

\[
\Delta \phi = \nabla \cdot (\nabla \phi) = \sum_{d=0}^{D-1} \frac{\partial F^d}{\partial x_d}
\]

\[
F^d = \frac{\partial \phi}{\partial x_d}
\]

Using the Divergence Theorem, this leads to a natural discretization of the Laplacian.

\[
-\frac{1}{h^D} \int_{V_i} \nabla \cdot \vec{F} \, d\vec{x} = -\frac{1}{h} \sum_d \langle F^d \rangle_{i+\frac{1}{2}e^d} - \langle F^d \rangle_{i-\frac{1}{2}e^d}
\]

\[
\langle F^d \rangle_{i+\frac{1}{2}e^d} \equiv \frac{1}{h^{D-1}} \int_{A_d^{\pm}} F^d(\vec{x}, t) \, dA
\]

This discretization satisfies a discrete analogue of the divergence theorem.

\[
\sum_{j \in \Omega} D(F)_j = \frac{1}{\Delta x} \sum_{j+\frac{1}{2}e^s \in \partial \Omega} \pm F^s_{j+\frac{1}{2}e^s}
\]
Finite-Volume Methods on Structured Grids

We use centered-differences and the midpoint rule to approximate fluxes:

\[ \langle F^d \rangle_{i + \frac{1}{2} e^d} = \left\langle \frac{\partial \phi}{\partial x_d} \right\rangle_{i + \frac{1}{2} e^d} \approx \frac{\phi_{i+e^d} - \phi_i}{h} \]

Away from boundaries, this leads to the same finite-difference approximation to the Laplacian.

\[ \Delta \phi|_{i h} \approx (\Delta^h \phi^h)_i = \frac{1}{h^2} \left( -2D \phi_i + \sum_{d=0}^{D-1} (\phi_{i+e^d} + \phi_{i-e^d}) \right) \]

At boundaries, it is different. It also interacts differently with multigrid.
Boundary conditions are expressed in terms of computing fluxes at boundary faces.

Coarsening is done by aggregating volumes, rather than by sampling gridpoints.

$$\left. \frac{\partial \phi}{\partial x_d} \right|_{i_b - \frac{1}{2} e_d} \approx \frac{2}{h} \phi_{i_b}$$

$$\phi = 0 \text{ on } \partial \Omega$$
Finite-Volume Methods on Structured Grids

Averaging and interpolation:

\[ A(f)_{i_c} = \frac{1}{2^D} \sum_{s \in \{0,1\}^D} f_{2i_c+s} \]

“Charge conservation” motivated by PDE

\[ I(f)_i = f_{i/2} \]

Piecewise constant interpolation!
Finite-Volume Methods on Structured Grids

Why would this work? 1D example …

Piecewise constant is linear with small +/- mode, which PJ removes
Discrete Fourier Analysis: 2D Poisson example

- $O(N)$ PJ + average/interpolate on each level
- Coarser levels $O(N/4) + O(N/16) + O(N/64) + \ldots$ work, better in 2D, even better in 3D
- Choosing $\mathcal{A}$, $\mathcal{I}$, and relaxation scheme are critical
- Note: need to shift FFT by $e^{2\pi i \frac{n}{N}}$ for FV
Results (Fourth-Order Finite-Volume)

Fig. 10. Problem 4: plot of the $\infty$-norm of the residual versus multigrid iteration.
The Heat Equation

\[
\frac{\partial \phi}{\partial t} = \Delta \phi + f
\]

\[\phi(x, t), f(x, t) = \phi, f : \Omega \times [0, T] \rightarrow \mathbb{R}\]

\[\phi(\cdot, t) = 0 \text{ on } \partial \Omega\]

Explicit discretization in time leads to time step restrictions of the form

\[
\frac{\Delta t}{h^2} \leq C
\]

So instead, we use implicit discretizations, e.g. Backward Euler:

\[
\phi^{n+1} = \phi^n + \Delta t \Delta^h \phi^{n+1} + \Delta t f^{n+1}
\]

\[
L(\phi^{n+1}) = \phi^n + \Delta t f^{n+1}, \quad L = (I - \Delta t \Delta^h)
\]

More generally, any implicit method for the heat equation will involve solving

\[
L(\phi) = f, \quad L = (I - \mu \Delta^h), \quad \mu > 0
\]
The Heat Equation – Point Jacobi

\[ \phi^{l+1} = \phi^l + \lambda \left( f - (I - \mu \Delta) \phi^l \right) \]

Can do the same residual / error analysis as in Poisson to obtain

\[ \delta^l_i = \phi^l_i - \phi_i \]

\[ (L\delta)_i^l = R_i^l, \ R^l = L\phi^l - f \]

\[ \delta^{l+1} = \delta^l - \lambda L\delta^l \]

Discrete Fourier analysis leads to:

\[ \omega(k) = 1 - \lambda + \frac{2\mu \lambda}{h^2} \left( \cos(2\pi k h) - 1 \right) \]

For \( \mu \) large (big \( \Delta t \) or strong diffusion) looks like Poisson, but for \( \mu \) small this looks more like identity operator (converges much faster for low-\( k \)):

\[ \omega(k) \approx 1 - \lambda \]
The Heat Equation – Point Jacobi

Relaxation parameter that eliminates highest-$k$ wave numbers:

\[ \omega\left(\frac{N}{2}\right) = 1 - \lambda + \frac{2\mu \lambda}{h^2} \left(\cos(\pi) - 1\right) \]

\[ \Rightarrow \lambda = \frac{1}{1 + \frac{4\mu}{h^2}} \]

Note as $h \to 0$ this becomes smaller (more Poisson-like), but $~1$ for $\mu \ll \frac{h^2}{4}$

More generally for any $D$:

\[ \lambda = \frac{1}{1 + \frac{4D\mu}{h^2}} \]

\[ \delta_{i+1} = \lambda \left(\frac{2D\mu}{h^2}\right) \left(\delta_i + \frac{1}{2D} \sum_d \delta_{i-e_d} + \delta_{i+e_d}\right) \]

Weighted sum, positive weights, but sum to less than 1, more so as $\mu$ gets smaller … heat equation converges faster than Poisson, sometimes very fast.