Multigrid for Poisson's Equation

Prof. Richard Vuduc

Georgia Institute of Technology

CSE/CS 8803 PNA, Spring 2008

[L.12] Thursday, February 14, 2008

Sources for today's material

- CS 267 (Yelick & Demmel, UCB)
- *Applied Numerical Linear Algebra*, by Demmel
- Heath (UIUC)

Review: Parallel FFT

Exploiting structure to obtain fast algorithms for 2-D Poisson

- **Dense LU**: Assume no structure ⇒ $O(n^6)$
- **Sparse LU**: Sparsity ⇒ $O(n^3)$, need extra memory
- **CG**: Symmetric positive definite ⇒ $O(n^3)$, a little extra memory
- **RB SOR**: Fixed sparsity pattern ⇒ $O(n^3)$, no extra memory
- ***FFT****: Eigendecomposition ⇒ $O(n^2 \log n)$*

Fast Fourier transform algorithm

$\mathrm{FFT}(x, \omega, m)$

if $m == 1$ **then**

return x_0

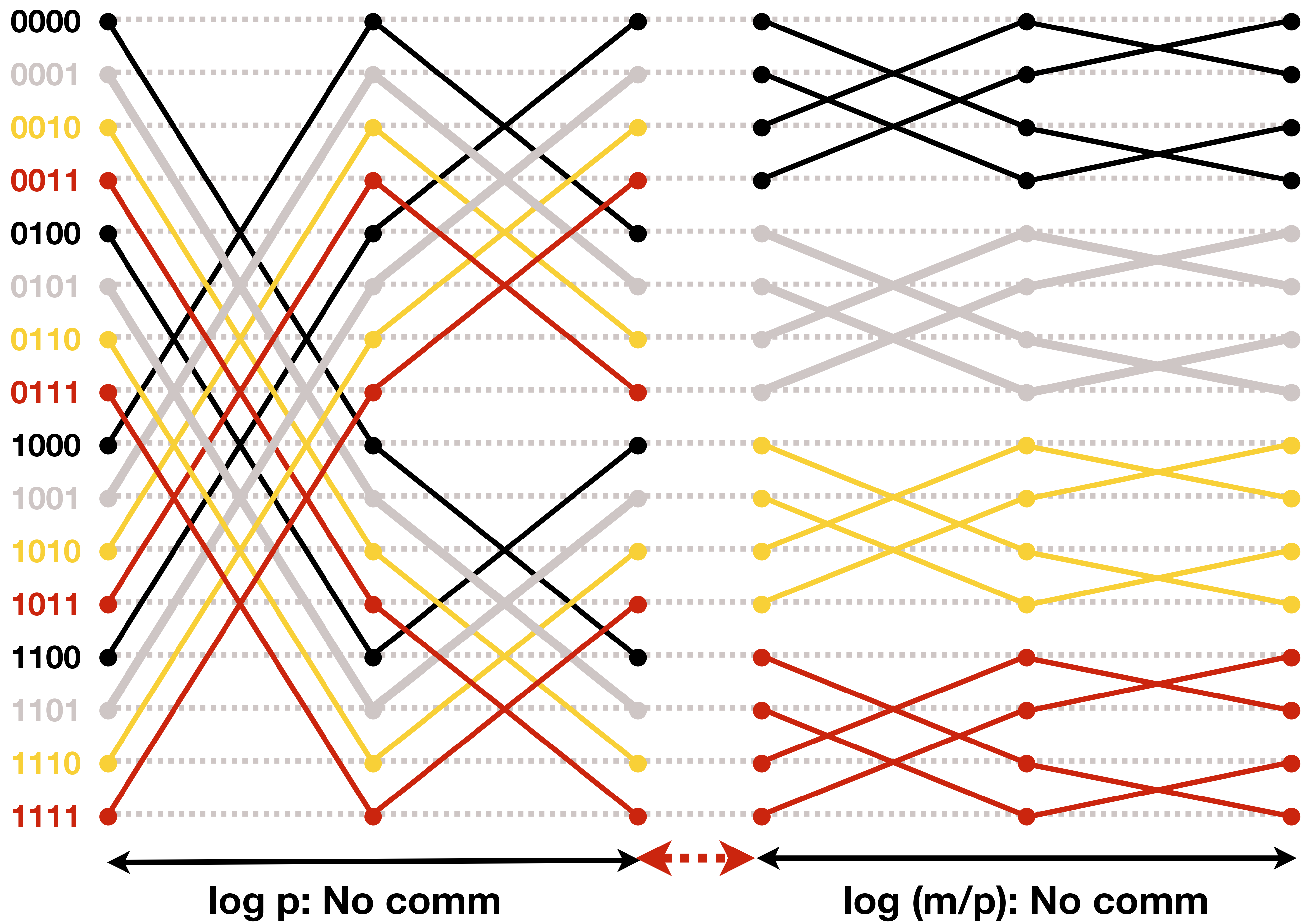
else

$x_{\text{even}} \leftarrow \mathrm{FFT}(x_{\text{even}}, \omega^2, \frac{m}{2})$

$x_{\text{odd}} \leftarrow \mathrm{FFT}(x_{\text{odd}}, \omega^2, \frac{m}{2})$

$w \leftarrow \left[w^0, w^1, \ldots, \omega^{\frac{m}{2}-1}\right]$ $\Longleftarrow$ precomputed

return $\left[x_{\text{even}} + (w. * x_{\text{odd}}),\ x_{\text{even}} - (w. * x_{\text{odd}})\right]$



FFT with transpose (p=4)

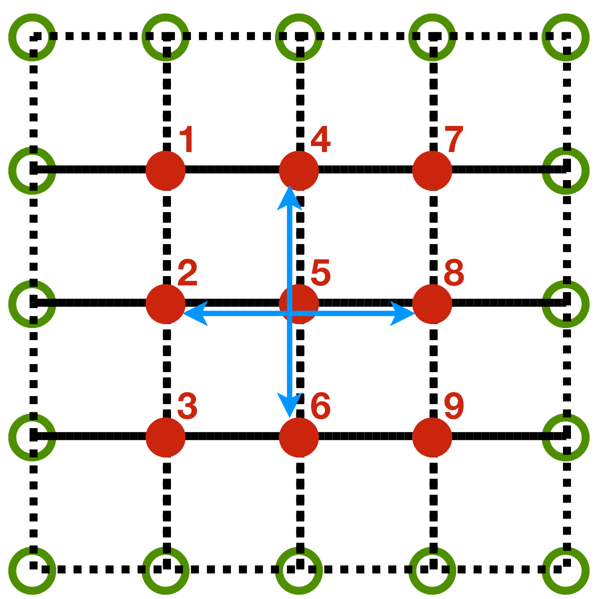
Algorithms for 2-D (3-D) Poisson, $N=n^2$ ($=n^3$)

Algorithm	Serial	PRAM	Memory	# procs
Dense LU	N^3	N	N^2	N^2
Band LU	N^2 ($N^{7/3}$)	N	$N^{3/2}$ ($N^{5/3}$)	N ($N^{4/3}$)
Jacobi	N^2 ($N^{5/3}$)	N ($N^{2/3}$)	N	N
Explicit inverse	N^2	$\log N$	N^2	N^2
Sparse LU	$N^{3/2}$ (N^2)	$N^{1/2}$	$N \log N$ ($N^{4/3}$)	N
Conj. grad.	$N^{3/2}$ ($N^{4/3}$)	$N^{1/2(1/3)} \log N$	N	N
RB SOR	$N^{3/2}$ ($N^{4/3}$)	$N^{1/2}$ ($N^{1/3}$)	N	N
FFT	$N \log N$	$\log N$	N	N
Multigrid	N	$\log^2 N$	N	N
Lower bound	N	$\log N$	N	

PRAM = idealized parallel model with zero communication cost.
Source: Demmel (1997)

Recall:
Discretizing 2-D Poisson

Graph and stencil



$$
\begin{array}{c}
\begin{array}{ccccccccc}
 & 2 & & 4 & 5 & 6 & & 8 &
\end{array} \\
\left(
\begin{array}{ccc|ccc|ccc}
4 & -1 & & -1 & & & & & \\
-1 & 4 & -1 & & -1 & & & & \\
 & -1 & 4 & & & -1 & & & \\
\hline
-1 & & & 4 & -1 & & -1 & & \\
 & -1 & & -1 & 4 & -1 & & -1 & \\
 & & -1 & & -1 & 4 & & & -1 \\
\hline
 & & & -1 & & & 4 & -1 & \\
 & & & & -1 & & -1 & 4 & -1 \\
 & & & & & -1 & & -1 & 4
\end{array}
\right)
\end{array}
$$

$$4u_{ij} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} = h^2 f_{i,j}$$

Recall: Jacobi's method

- Rearrange terms in (2-D) Poisson:

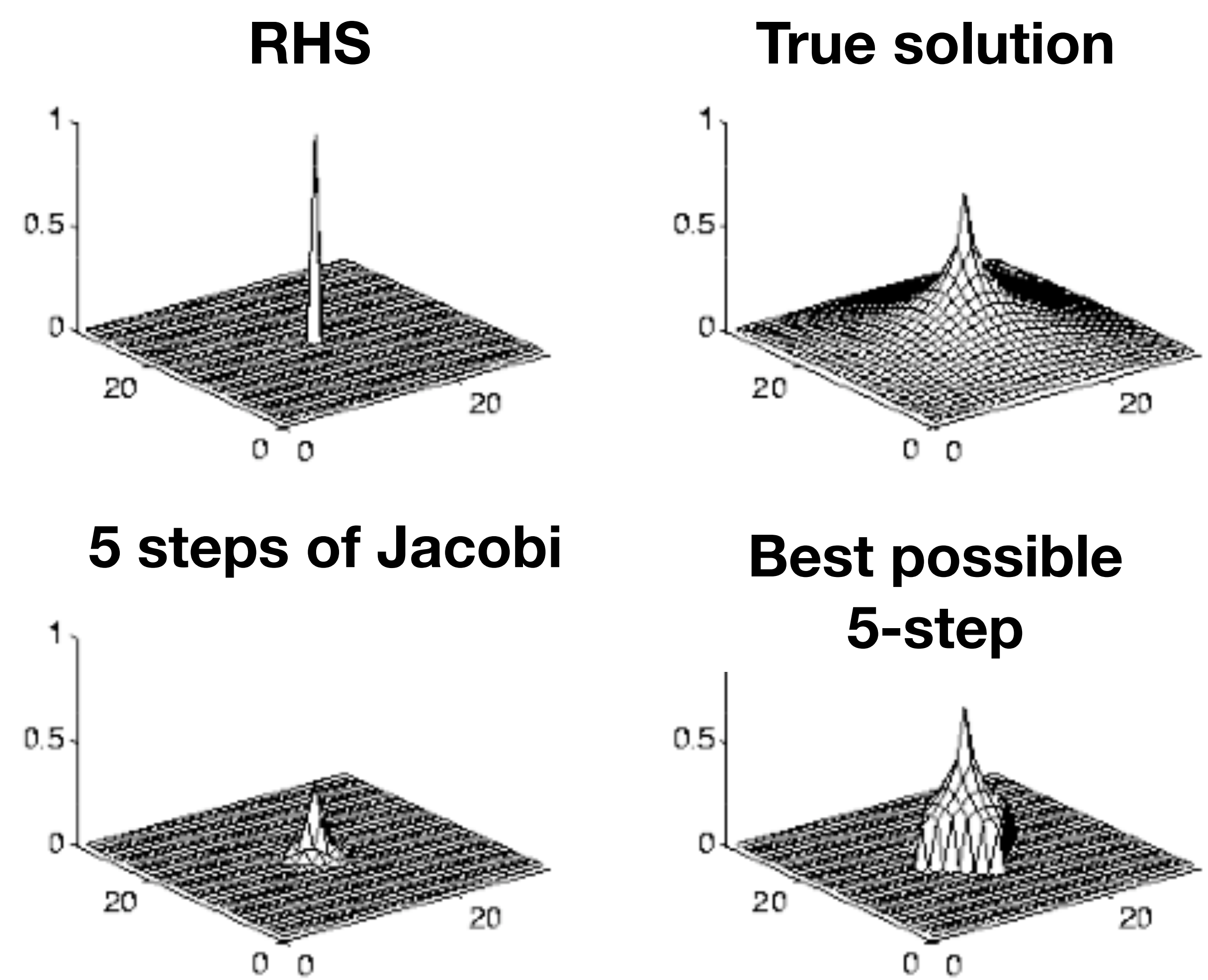
$$u_{i,j} = \frac{1}{4}\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} + h^2 f_{i,j}\right)$$

- For each (i, j), iteratively update (weighted averaging):

$$u_{i,j}^{(t+1)} = \frac{1}{4}\left(u_{i-1,j}^{(t)} + u_{i+1,j}^{(t)} + u_{i,j-1}^{(t)} + u_{i,j+1}^{(t)} + h^2 f_{i,j}\right)$$

- Motivation: Match solution to discrete Poisson exactly at nodes.

Problem: Slow convergence



Recall: Convergence of Jacobi's method

- Converges in $O(N=n^2)$ steps, so serial complexity is $O(N^2)$.
- Define error at each step as:

$$\epsilon_t \triangleq \sqrt{\sum_{i,j} (u_{i,j}^t - u_{i,j})^2}$$

- For Jacobi, can show:

$$\epsilon_t \leq \left(\cos \frac{\pi}{n+1}\right)^t \epsilon_0 \quad \overset{n \to \infty}{\approx} \quad \left(1 - \frac{\pi^2}{4} \cdot \frac{t}{n^2}\right) \epsilon_0$$

Consider Jacobi in 1-D

$$
\begin{aligned}
h^2 f_i &= -u_{i-1} + 2u_i - u_{i+1} \\
u_i^{(t+1)} &= \frac{1}{2}\left(u_{i-1}^{(t)} + u_{i+1}^{(t)}\right) + \frac{h^2}{2} f_i \\
&\Downarrow \\
u^{(t+1)} &= \underbrace{\left(I - \frac{1}{2} T_n\right)}_{\equiv R} \cdot u^{(t)} + \frac{h^2}{2} f \\
&\equiv R \cdot u^{(t)} + c
\end{aligned}
$$

Closer look at 1-D Jacobi's convergence

- Consider total error at each step

$$
\begin{aligned}
\epsilon^{(t)} &\equiv u^{(t)} - u \\
&= R \cdot \epsilon^{(t-1)} \\
&= R^t \cdot e_0
\end{aligned}
$$

Closer look at 1-D Jacobi's convergence

- Consider slightly broader class of *weighted* Jacobi methods

$$
\begin{aligned}
T_n u &= h^2 f \equiv c \\
R_w &\equiv I - \frac{w}{2} T_n \\
u^{(t+1)} &= R_w u^{(t)} + \frac{w}{2} c \\
&\Downarrow \\
\epsilon^{(t)} &= R_w^t \epsilon^{(0)}
\end{aligned}
$$

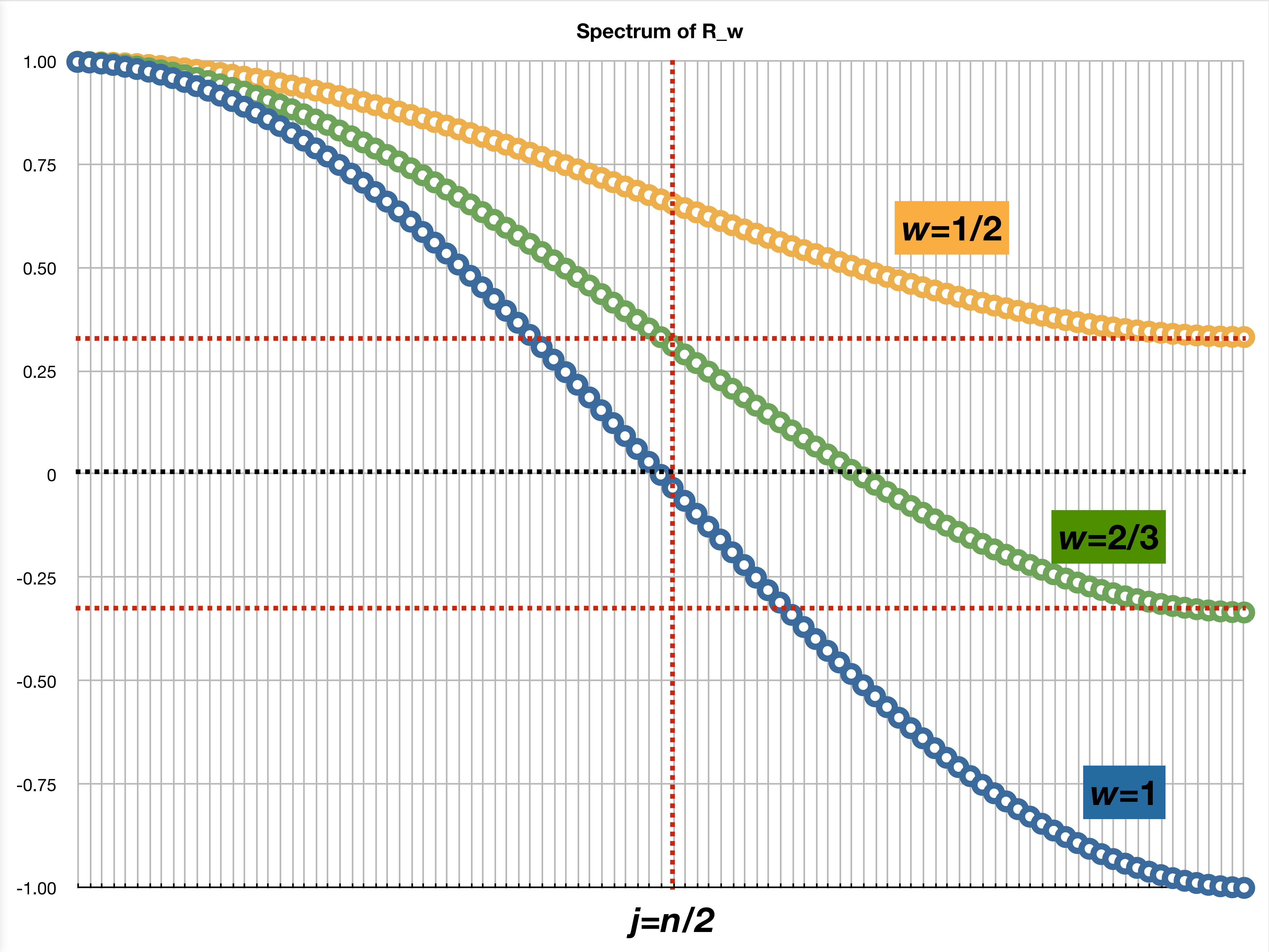
1-D Poisson: Eigendecomposition of T_n

- *Lemma*: Eigenvalues and eigenvectors of T_n are

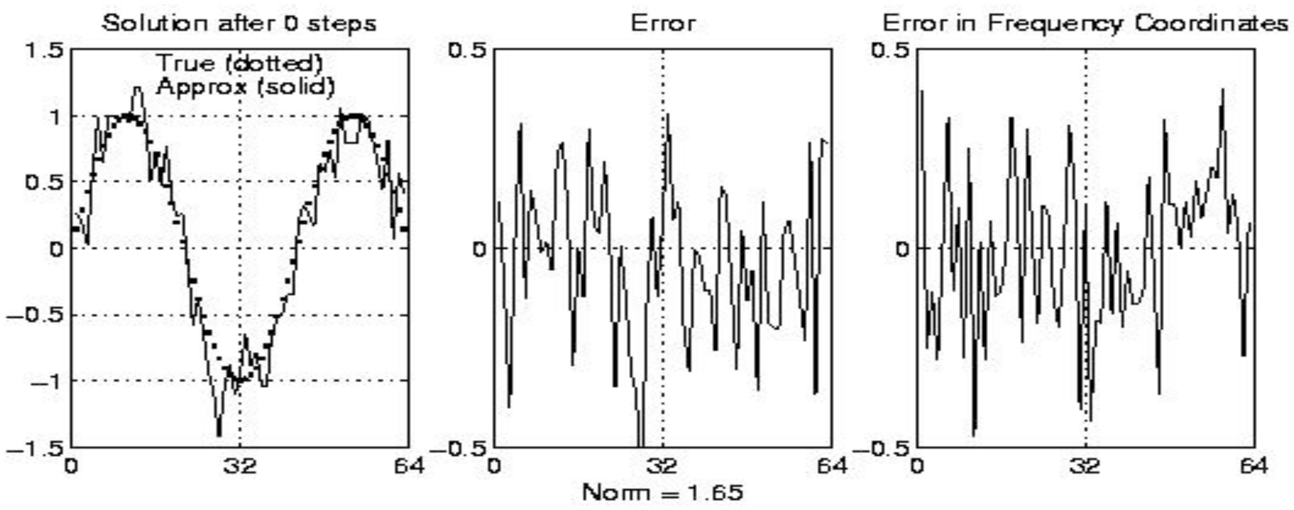
$$
\begin{aligned}
T_n z_j &= \lambda_j z_j \\
\lambda_j &= 2\left(1 - \cos\frac{\pi j}{n+1}\right) \\
z_j(k) &= \sqrt{\frac{2}{n+1}}\sin\frac{\pi j k}{n+1}
\end{aligned}
\qquad \Longrightarrow \qquad
\begin{aligned}
T_n &= Z \Lambda Z^T \\
Z Z^T &= I
\end{aligned}
$$

Error “frequencies”

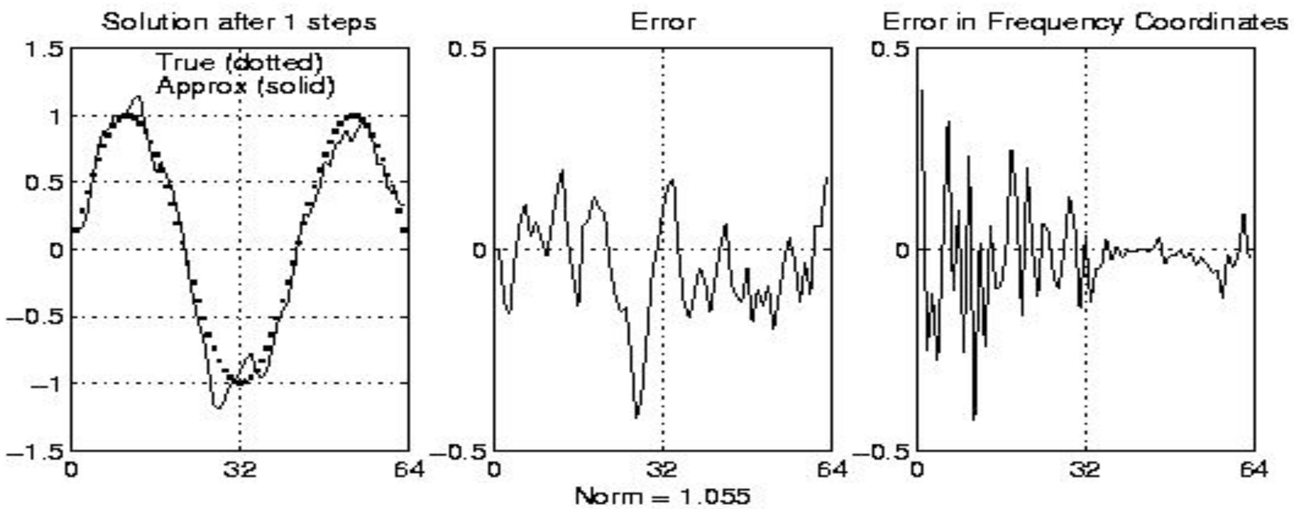
$$
\begin{aligned}
\epsilon^{(t)} = R_w^t \cdot \epsilon^{(0)} \quad &= \quad (I - \frac{w}{2} Z \Lambda Z^T)^t \cdot \epsilon^{(0)} \\
&= \quad Z \left(I - \frac{w}{2} \Lambda \right)^t Z^T \cdot \epsilon^{(0)} \\
&\Downarrow \\
Z^T \cdot \epsilon^{(t)} \quad &= \quad \left(I - \frac{w}{2} \Lambda \right)^t Z^T \cdot \epsilon^{(0)} \\
\left(Z^T \cdot \epsilon^{(t)} \right)_j \quad &= \quad \left(I - \frac{w}{2} \Lambda \right)^t_{jj} \left(Z^T \cdot \epsilon^{(0)} \right)_j
\end{aligned}
$$



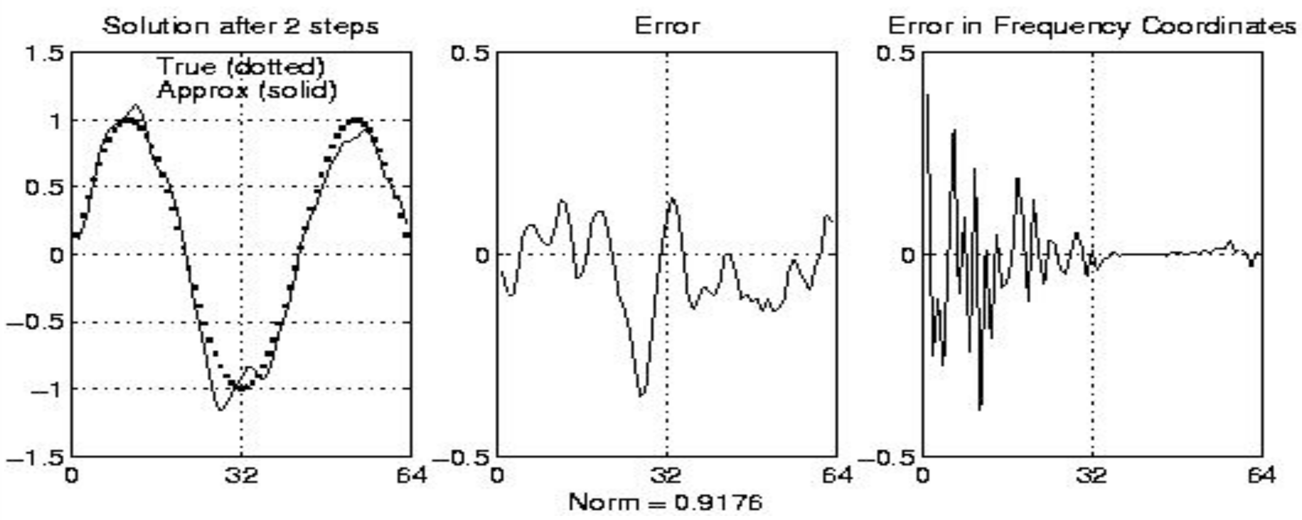
For 1-D discrete Poisson, with n = 99



Initial error
"Rough"
Lots of high frequency components
Norm = 1.65



Error after 1 weighted Jacobi step
"Smoother"
Less high frequency component
Norm = 1.06



Error after 2 weighted Jacobi steps
"Smooth"
Little high frequency component
Norm = .92,
won't decrease much more

Faster information propagation?

- "Multigrid" idea
  - Approximate problem on fine grid by a coarser grid
  - Solve the coarse grid problem approximately
  - Interpolate coarse grid solution onto fine grid, as an initial guess
  - Recursively solve coarse grid problems
- To work, coarse grid solution must be a good approximation to fine grid

Same idea applies elsewhere

- Multilevel graph partitioning (METIS)
  - Coarsen graph via maximal independent set problem
  - Partition coarse graph, refine using Kernighan-Lin
- Barnes-Hut and fast multipole method for, say, gravity problems
  - Approximate particles in a region by total mass, center of gravity
  - Divide regions recursively

Divide-and-conquer in multigrid

- Spatial domain
  - Get an initial solution for an $n \times n$ grid by solving approximately on $n/2 \times n/2$ grid
  - Recurse
- Frequency domain
  - Think of error as sum of eigenvectors, *e.g.*, sine-curves of different frequencies
  - Solving on a particular grid “smooths” (dampens) high-frequency error

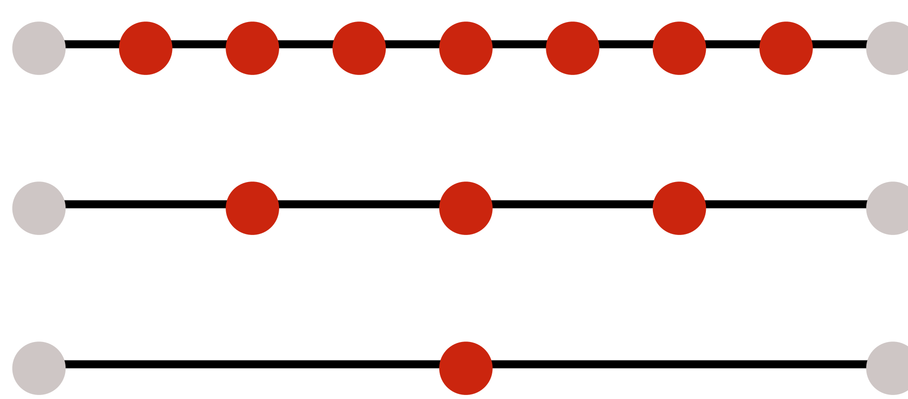
“Multigrids” in 1-D

$$
\begin{aligned}
P^{(i)} &= \text{Problem on } 2^i + 1 \text{ grid} \\
T^{(i)} x^{(i)} &= b^{(i)}
\end{aligned}
$$

$P^{(3)}$

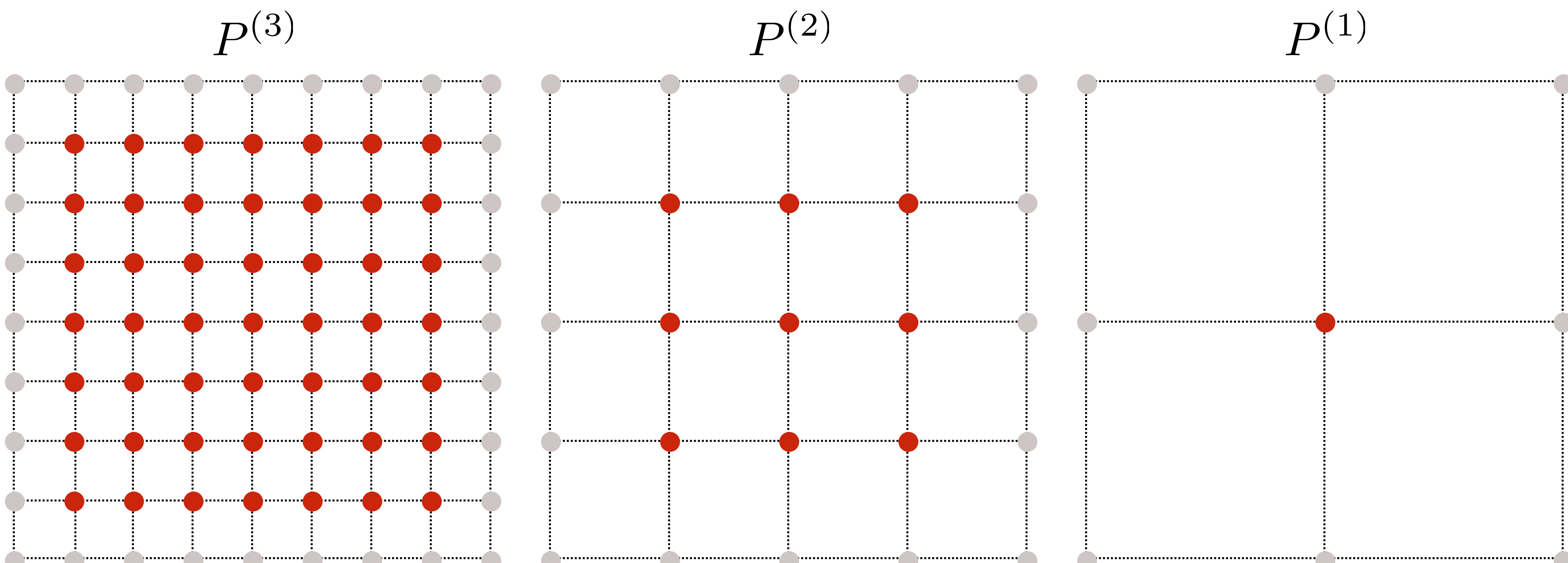
$P^{(2)}$

$P^{(1)}$

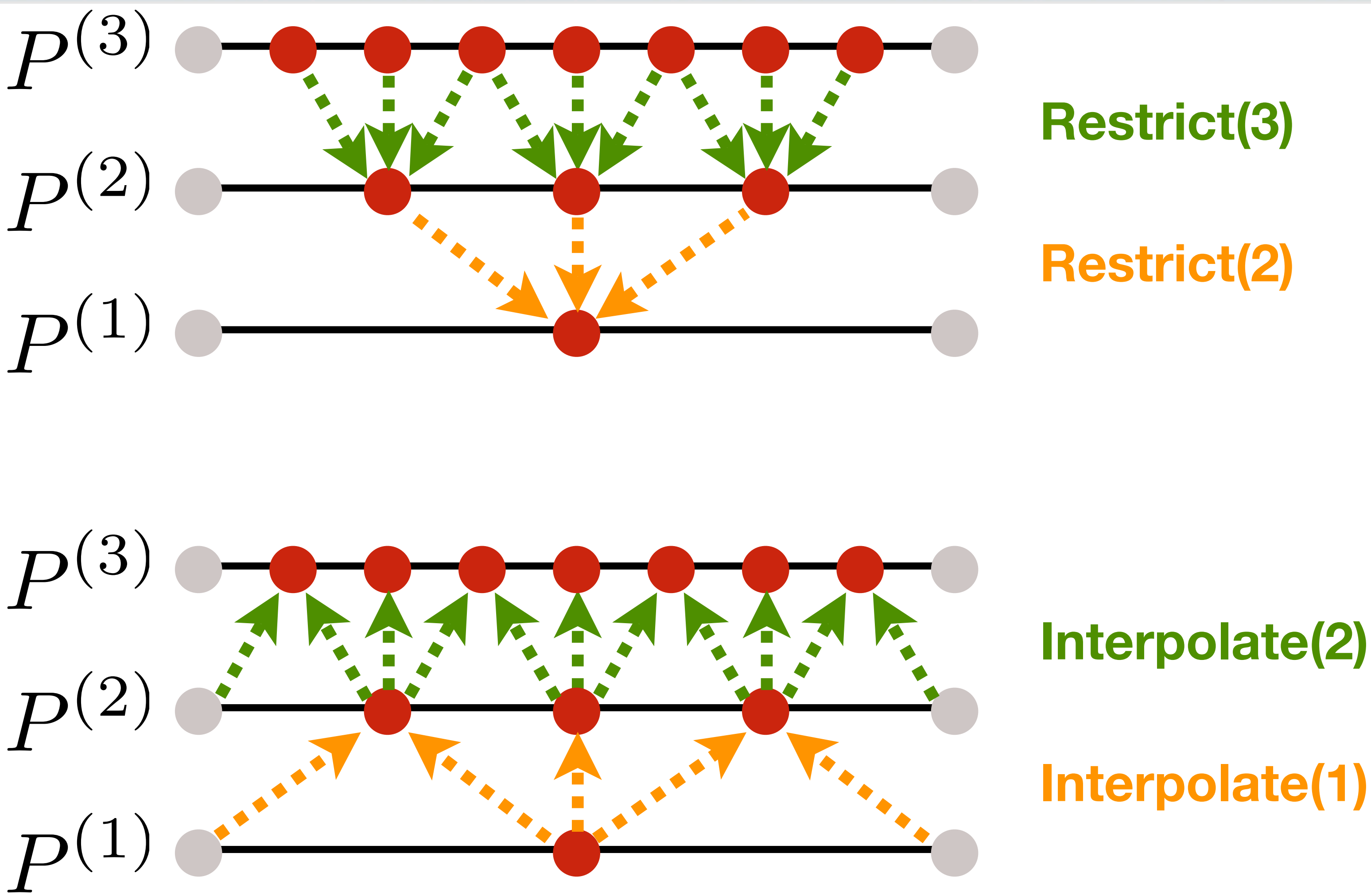


"Multigrids" in 2-D

$$\begin{aligned} P^{(i)} &= \text{Problem on } \left(2^i+1\right) \times \left(2^i+1\right) \text{ grid} \\ T^{(i)} x^{(i)} &= b^{(i)} \end{aligned}$$



Multigrid operators



Multigrid operators

$$P^{(i)} : x^{(i)} \equiv$$ **Current estimated solution**

$$b^{(i)} \equiv$$ **Right-hand side**

$$R^{(i)} \quad : \quad b^{(i-1)} \leftarrow R^{(i)}\left(b^{(i)}\right)$$ ⇐ **Restriction**

$$L^{(i)} \quad : \quad x^{(i)} \leftarrow L^{(i-1)}\left(x^{(i-1)}\right)$$ ⇐ **Interpolation**

$$S^{(i)} \quad : \quad x^{(i)}_{\text{improved}} \leftarrow S^{(i)}\left(b^{(i)}, x^{(i)}\right)$$

⇑ **Solution operator (smoother)**

Multigrid "V-cycle": Building block for the full multigrid algorithm

$\text{MGV}\left(b^{(i)}, x^{(i)}\right)$ ⇐ **Returns improved $x^{(i)}$ for $T^{(i)}x^{(i)} = b^{(i)}$**

if $i == 1$ **then**

  return exact solution of $P^{(1)}$

else

  $x^{(i)} \leftarrow S^{(i)}\left(b^{(i)}, x^{(i)}\right)$

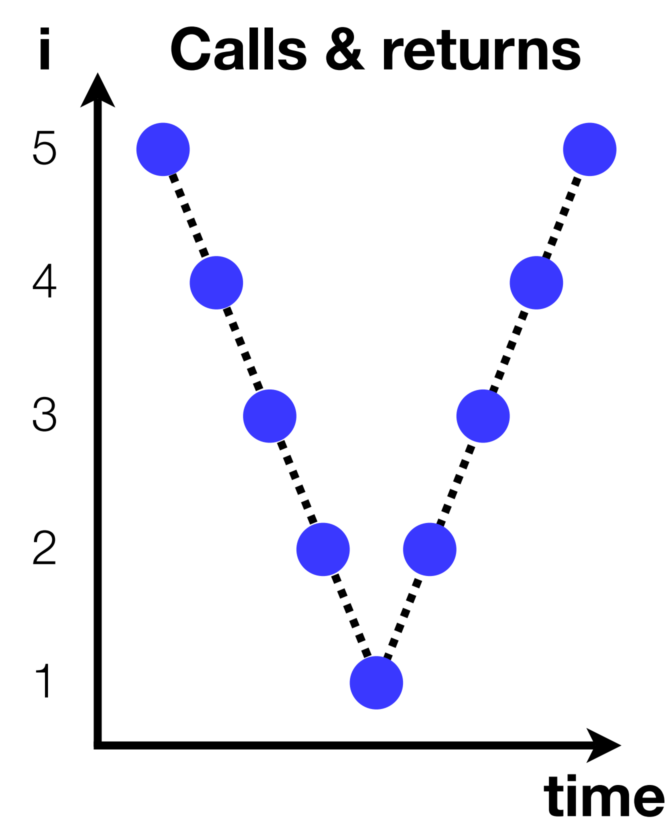
  $r^{(i)} \leftarrow T^{(i)} \cdot x^{(i)} - b^{(i)}$

  $d^{(i)} \leftarrow L^{(i-1)}\left(\text{MGV}\left(R^{(i)}\left(r^{(i)}\right), 0\right)\right)$

  $x^{(i)} \leftarrow x^{(i)} - d^{(i)}$

  $x^{(i)} \leftarrow S^{(i)}\left(b^{(i)}, x^{(i)}\right)$

  return $x^{(i)}$



Multigrid V-cycle

$\mathrm{MGV}\left(b^{(i)}, x^{(i)}\right)$ ⇐ Returns improved $x^{(i)}$ for $T^{(i)}x^{(i)} = b^{(i)}$

if $i == 1$ **then**

  return exact solution of $P^{(1)}$ **Base case: 1 unknown**

else

  $x^{(i)} \leftarrow S^{(i)}\left(b^{(i)}, x^{(i)}\right)$

  $r^{(i)} \leftarrow T^{(i)} \cdot x^{(i)} - b^{(i)}$

  $d^{(i)} \leftarrow L^{(i-1)}\left(\mathrm{MGV}\left(R^{(i)}\left(r^{(i)}\right), 0\right)\right)$

  $x^{(i)} \leftarrow x^{(i)} - d^{(i)}$

  $x^{(i)} \leftarrow S^{(i)}\left(b^{(i)}, x^{(i)}\right)$

  return $x^{(i)}$

Multigrid V-cycle

$\mathrm{MGV}\left(b^{(i)}, x^{(i)}\right)$ ⇐ Returns improved $x^{(i)}$ for $T^{(i)}x^{(i)} = b^{(i)}$

if $i == 1$ **then**

  return exact solution of $P^{(1)}$ Base case: 1 unknown

else

  $x^{(i)} \leftarrow S^{(i)}\left(b^{(i)}, x^{(i)}\right)$ Smooth (damps high-freq error)

  $r^{(i)} \leftarrow T^{(i)} \cdot x^{(i)} - b^{(i)}$

  $d^{(i)} \leftarrow L^{(i-1)}\left(\mathrm{MGV}\left(R^{(i)}\left(r^{(i)}\right), 0\right)\right)$

  $x^{(i)} \leftarrow x^{(i)} - d^{(i)}$

  $x^{(i)} \leftarrow S^{(i)}\left(b^{(i)}, x^{(i)}\right)$

  return $x^{(i)}$

Multigrid V-cycle

$\text{MGV}\left(b^{(i)}, x^{(i)}\right)$ ⇐ Returns improved $x^{(i)}$ for $T^{(i)}x^{(i)} = b^{(i)}$

if $i == 1$ **then**

return exact solution of $P^{(1)}$ Base case: 1 unknown

else

$x^{(i)} \leftarrow S^{(i)}\left(b^{(i)}, x^{(i)}\right)$ Smooths (damps high-freq error)

$r^{(i)} \leftarrow T^{(i)} \cdot x^{(i)} - b^{(i)}$ Computes residual

$d^{(i)} \leftarrow L^{(i-1)}\left(\text{MGV}\left(R^{(i)}\left(r^{(i)}\right), 0\right)\right)$ Recursively solves

$x^{(i)} \leftarrow x^{(i)} - d^{(i)}$ Corrects fine-grid solution

$x^{(i)} \leftarrow S^{(i)}\left(b^{(i)}, x^{(i)}\right)$

return $x^{(i)}$

Multigrid V-cycle

$\text{MGV}\left(b^{(i)}, x^{(i)}\right)$ ⇐ Returns improved $x^{(i)}$ for $T^{(i)}x^{(i)} = b^{(i)}$

if $i == 1$ **then**

  return exact solution of $P^{(1)}$ Base case: 1 unknown

else

  $x^{(i)} \leftarrow S^{(i)}\left(b^{(i)}, x^{(i)}\right)$ Smooths (damps high-freq error)

  $r^{(i)} \leftarrow T^{(i)} \cdot x^{(i)} - b^{(i)}$ Computes residual

  $d^{(i)} \leftarrow L^{(i-1)}\left(\text{MGV}\left(R^{(i)}\left(r^{(i)}\right), 0\right)\right)$ Recursively solves

  $x^{(i)} \leftarrow x^{(i)} - d^{(i)}$ Corrects fine-grid solution

  $x^{(i)} \leftarrow S^{(i)}\left(b^{(i)}, x^{(i)}\right)$ **Smooth again**

  return $x^{(i)}$

Multigrid V-cycle

$\color{blue}{\mathrm{MGV}}\left(b^{(i)}, x^{(i)}\right)$ ⇐ **Returns improved $x^{(i)}$ for $T^{(i)}x^{(i)} = b^{(i)}$**

if $i == 1$ **then**

  return exact solution of $P^{(1)}$

else

  $x^{(i)} \leftarrow S^{(i)}\left(b^{(i)}, x^{(i)}\right)$

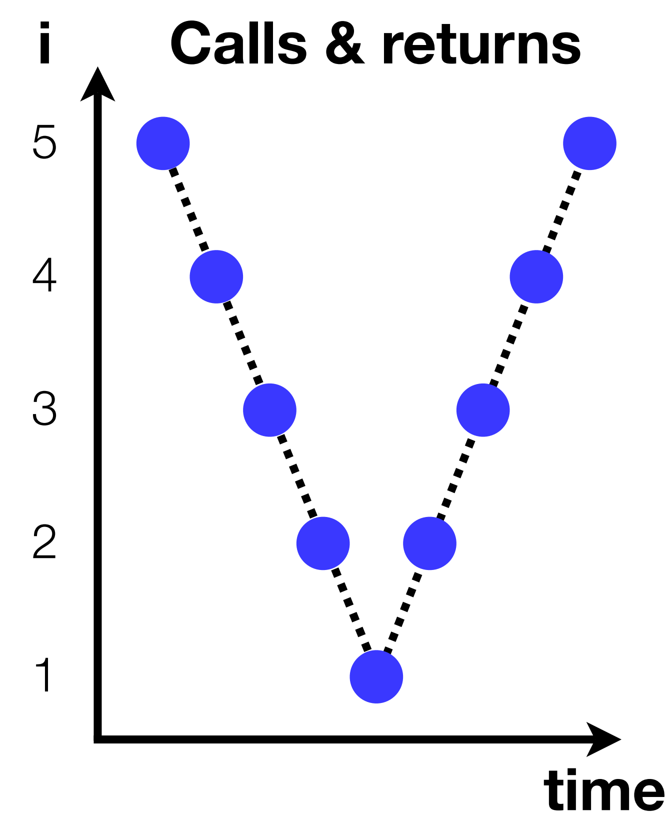
  $r^{(i)} \leftarrow T^{(i)} \cdot x^{(i)} - b^{(i)}$

  $d^{(i)} \leftarrow L^{(i-1)}\left(\color{blue}{\mathrm{MGV}\left(R^{(i)}\left(r^{(i)}\right), 0\right)}\right)$

  $x^{(i)} \leftarrow x^{(i)} - d^{(i)}$

  $x^{(i)} \leftarrow S^{(i)}\left(b^{(i)}, x^{(i)}\right)$

  return $x^{(i)}$



Multigrid V-cycle: Correction step

$\vdots$

$$r^{(i)} \leftarrow T^{(i)} \cdot x^{(i)} - b^{(i)}$$
$$d^{(i)} \leftarrow L^{(i-1)} \left(\mathrm{MGV} \left(R^{(i)} \left(r^{(i)} \right), 0 \right) \right)$$
$$x^{(i)} \leftarrow x^{(i)} - d^{(i)}$$

$\vdots$

Justification:

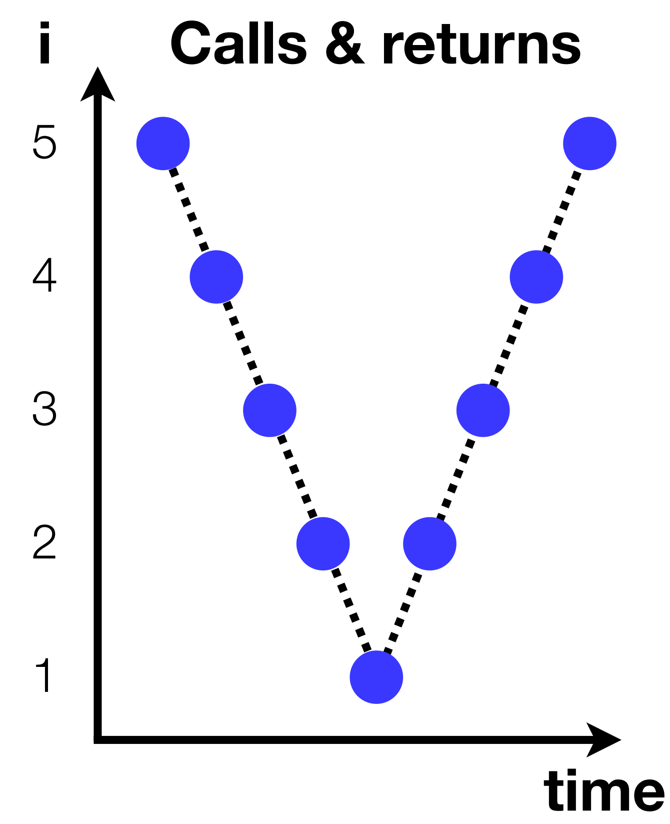
$$\begin{aligned} T^{(i)} d^{(i)} &= r^{(i)} \\ &= T^{(i)} x^{(i)} - b^{(i)} \\ \Longrightarrow b^{(i)} &= T^{(i)} \left(x^{(i)} - d^{(i)} \right) \end{aligned}$$

Multigrid V-cycle: Complexity

Serial complexity (2-D Poisson):

$$C(i) = \left\{ \begin{array}{cc} O\left(4^i\right) + C(i-1) & i > 1 \\ O(1) & i = 1 \end{array} \right\}$$

$$\begin{aligned} C(m) &= \sum_{i=1}^{m} O(4^i) = O(4^m) \\ &= O(\text{no. of unknowns}) \end{aligned}$$



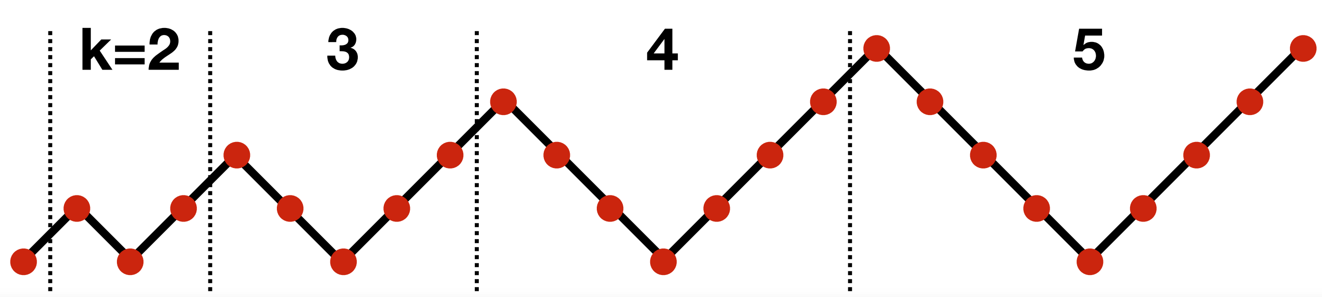
Full multigrid algorithm

$$\text{FMG}\left(b^{(k)}, x^{(k)}\right)$$

$$x^{(1)} \leftarrow \text{Exact solution to } P^{(1)}$$

$$\textbf{for } i = 2 \textbf{ to } k \textbf{ do}$$

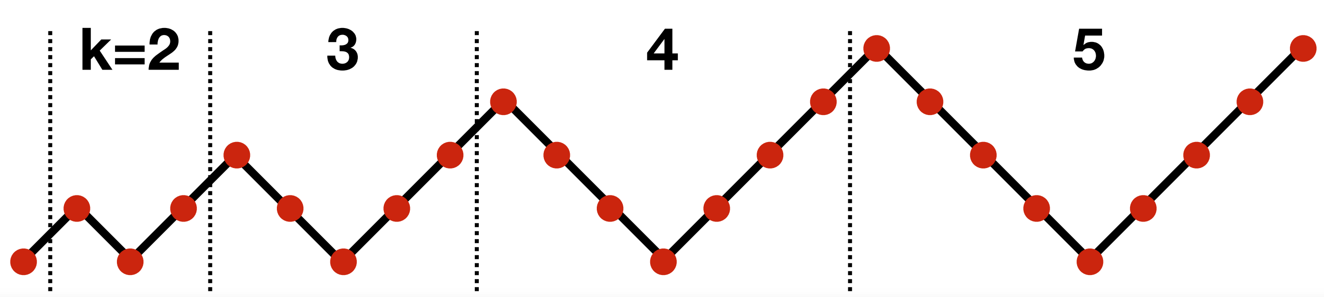
$$x^{(i)} \leftarrow \text{MGV}\left(b^{(i)}, L^{(i-1)}\left(x^{(i-1)}\right)\right)$$



Full multigrid algorithm: Complexity

$$\begin{aligned} C(m) &= \sum_{k=1}^{m} O(4^k) \\ &= O(4^m) = O(\text{no. of unknowns}) \end{aligned}$$

$$\begin{aligned} \text{PRAM} &= \sum_{k=1}^{m} O(k) \\ &= O(m^2) = O(\log^2(\text{unknowns})) \end{aligned}$$



Full multigrid: Convergence

- **Theorem**: After one FMG call,
  - error after <= .5 * (error before)
  - Independent of no. of unknowns (!)

- **Corollary**: Can make error < any fixed tolerance in a fixed no. of steps, independent of grid size (!)

Administrivia

Two joint classes with CS 8803 SC

- **Tues 2/19**: Floating-point issues in parallel computing by me
- **Tues 2/26**: GPGPUs by Prof. Hyesoon Kim
- **Both classes meet in Klaus 1116E**

Administrative stuff

- **New room** (dumpier, but cozier?): College of Computing Building **(CCB) 101**
- **Accounts**: Apparently, you already have them
- Front-end login node: **ccil.cc.gatech.edu** (CoC Unix account)
  - We "own" **warp43—warp56**
  - Some docs (**MPI**): http://www-static.cc.gatech.edu/projects/ihpcl/mpi.html
  - **Sign-up** for mailing list: https://mailman.cc.gatech.edu/mailman/listinfo/ihpc-lab

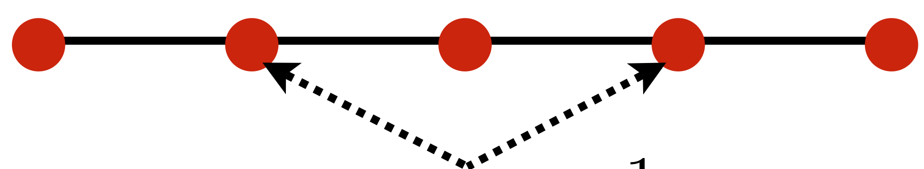
Homework 1: Parallel conjugate gradients

- Implement a parallel solver for Ax = b (serial C version provided)
  - Evaluate on three matrices: 27-pt stencil, and two application matrices
  - “Simplified:” No preconditioning
  - **Bonus**: Reorder, precondition
- Performance models to understand scalability of your implementation
  - Make measurements
  - Build predictive models
- Collaboration encouraged: Compare programming models or platforms

Some details on multigrid operators

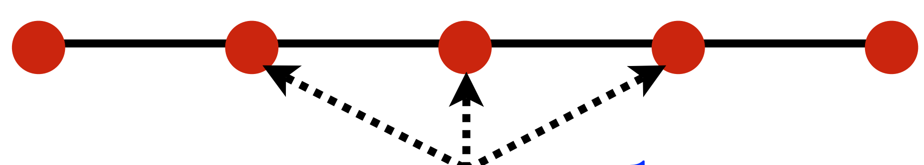
Solution operator (smoother), $S^{(i)}$: Weighted Jacobi

Recall: Jacobi

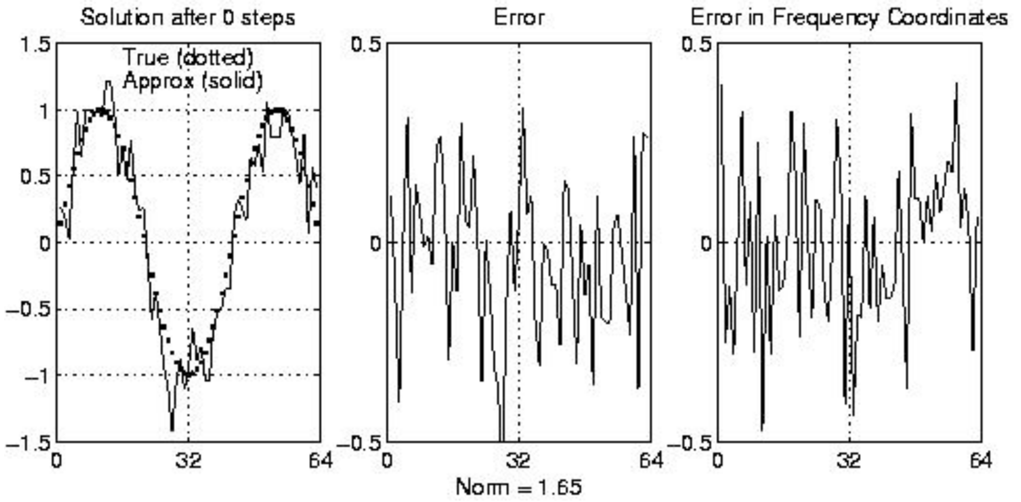


$$x_j \leftarrow \frac{1}{2}\left(x_{j-1} + x_{j+1} + b_j\right)$$

Weighted Jacobi



$$x_j \leftarrow \frac{1}{3}\left(x_{j-1} + x_j + x_{j+1} + b_j\right)$$

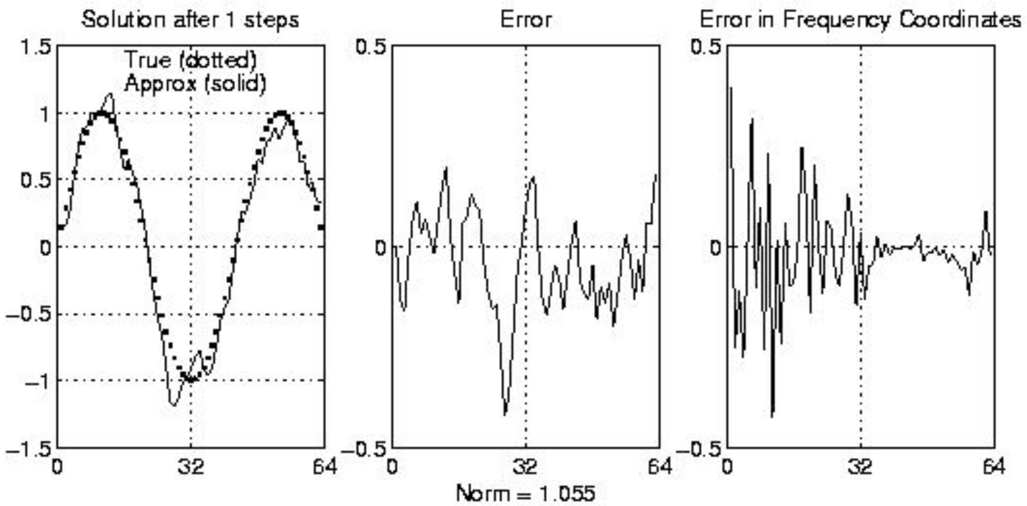


Initial error

"Rough"

Lots of high frequency components

Norm = 1.65

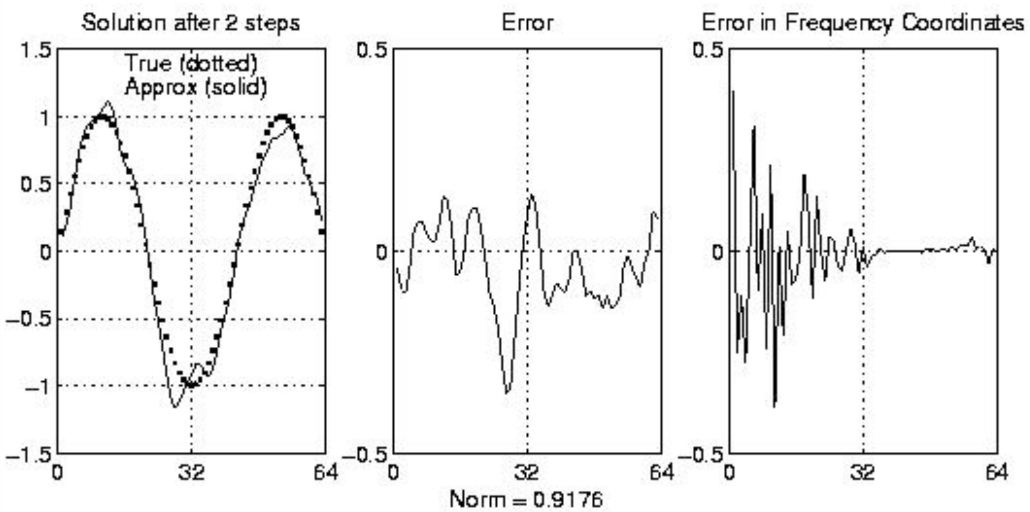


Error after 1 weighted Jacobi step

"Smoother"

Less high frequency component

Norm = 1.06



Error after 2 weighted Jacobi steps

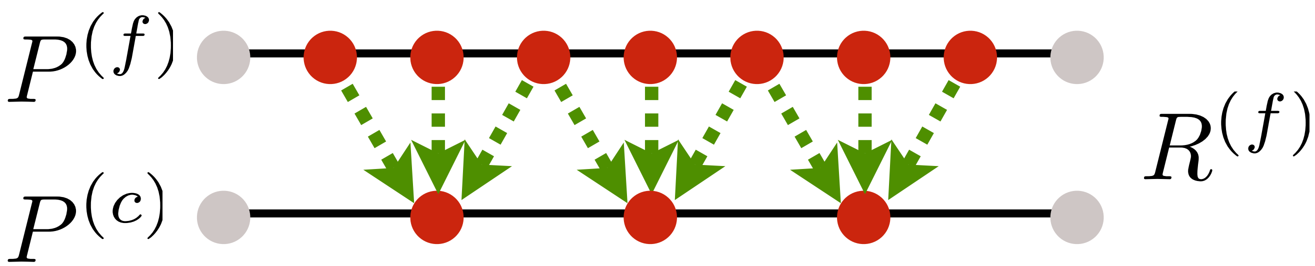
"Smooth"

Little high frequency component

Norm = .92,

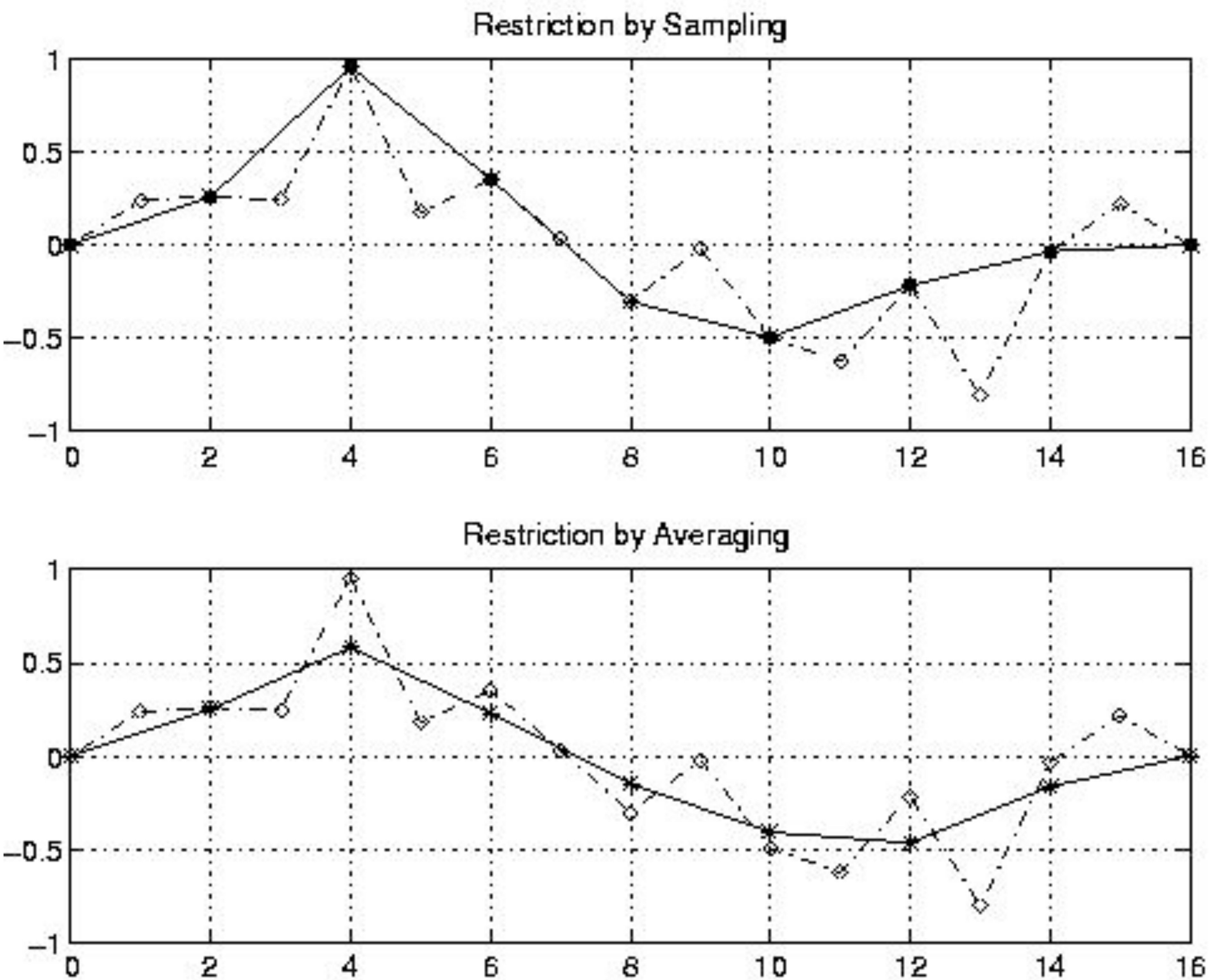
won't decrease much more

Restriction operator, $R^{(i)}$

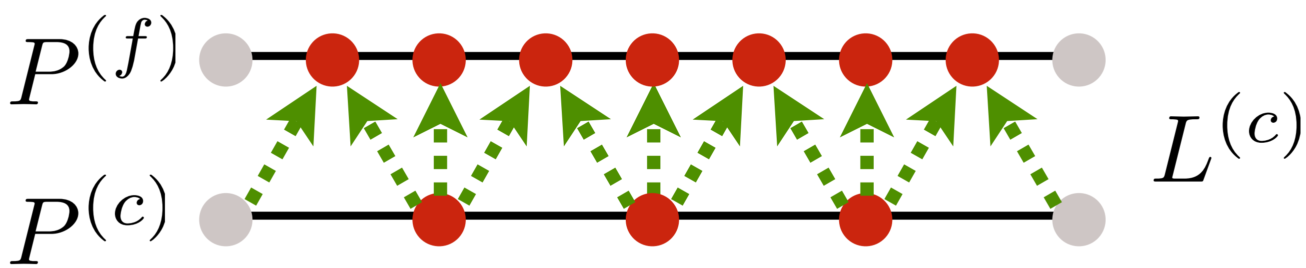


$$x_j^{(\mathrm{c})} \leftarrow \frac{1}{4}x_{j-1}^{(\mathrm{f})} + \frac{1}{2}x_j^{(\mathrm{f})} + \frac{1}{4}x_{j+1}^{(\mathrm{f})}$$

In 2-D: Average with all 8 neighbors

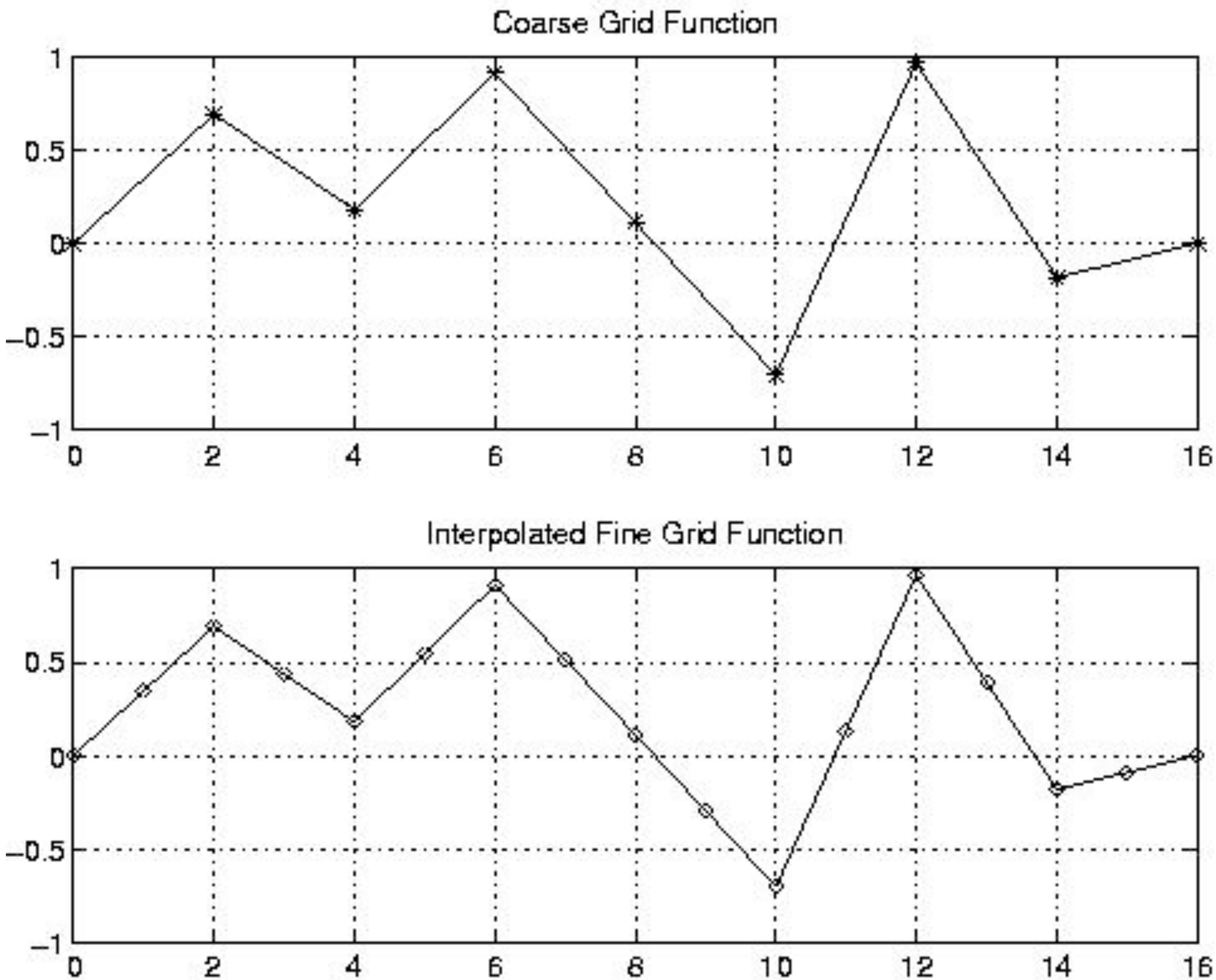

Restriction by Sampling
1
0.5
0
−0.5
−1
0
2
4
6
8
10
12
14
16
Restriction by Averaging
1
0.5
0
−0.5
−1
0
2
4
6
8
10
12
14
16

Interpolation operator, $L^{(i)}$

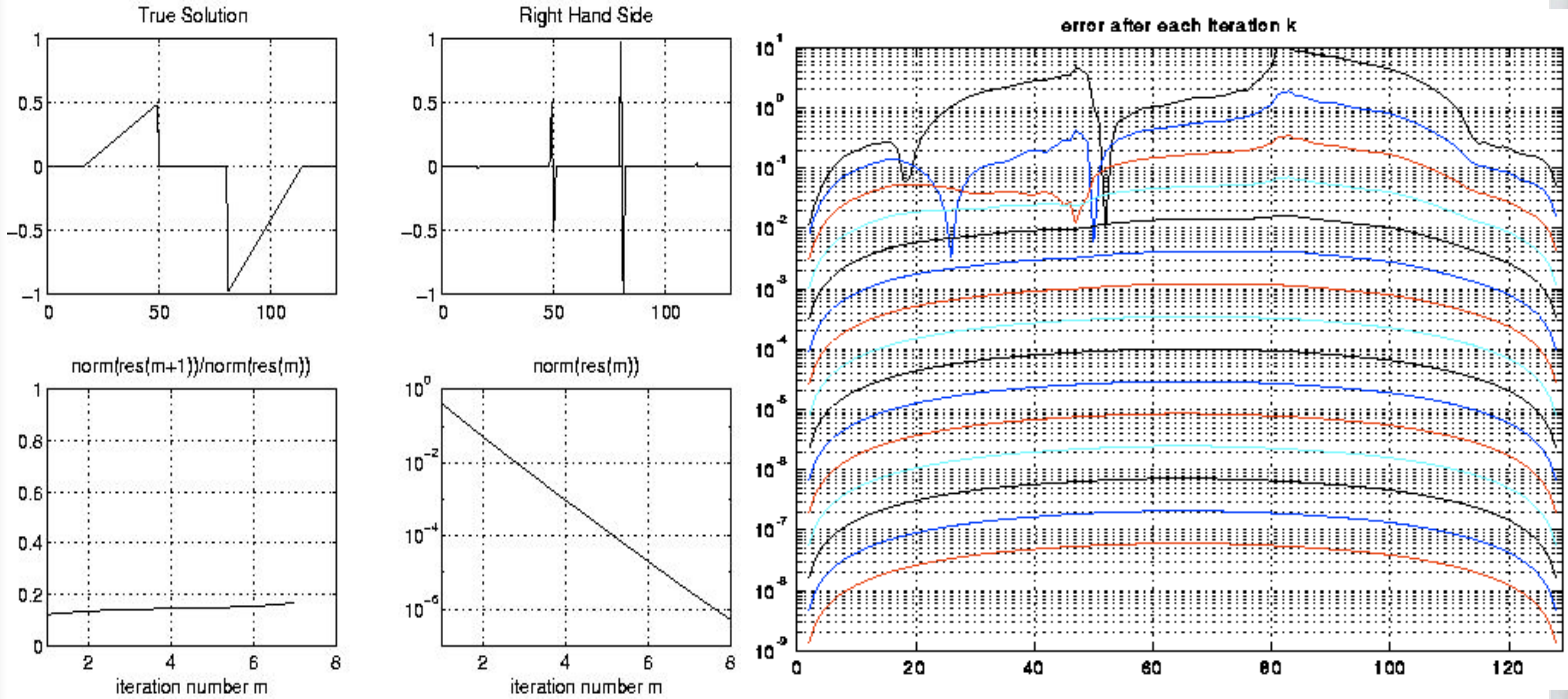


$$x_j^{(f)} = \left\{ \begin{array}{cc} x_j^{(c)} & \text{if } \exists\ x_j^{(c)} \\ \frac{1}{2}\left(x_{\text{left}(j)}^{(c)} + x_{\text{right}(j)}^{(c)}\right) & \text{otherwise} \end{array} \right\}$$

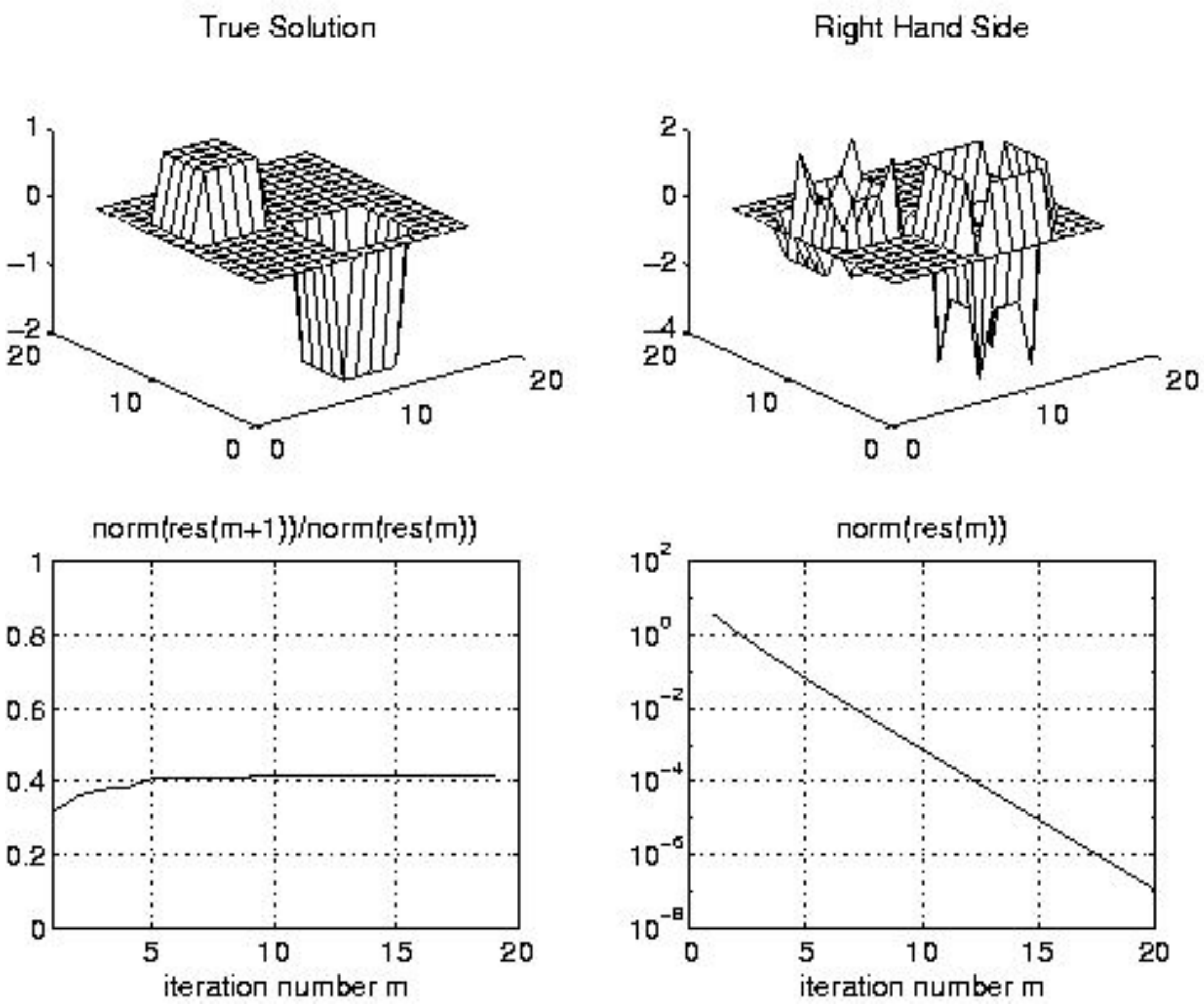
In 2-D: Average with all 8 neighbors


Coarse Grid Function
Interpolated Fine Grid Function
1
0.5
0
−0.5
−1
0
2
4
6
8
10
12
14
16

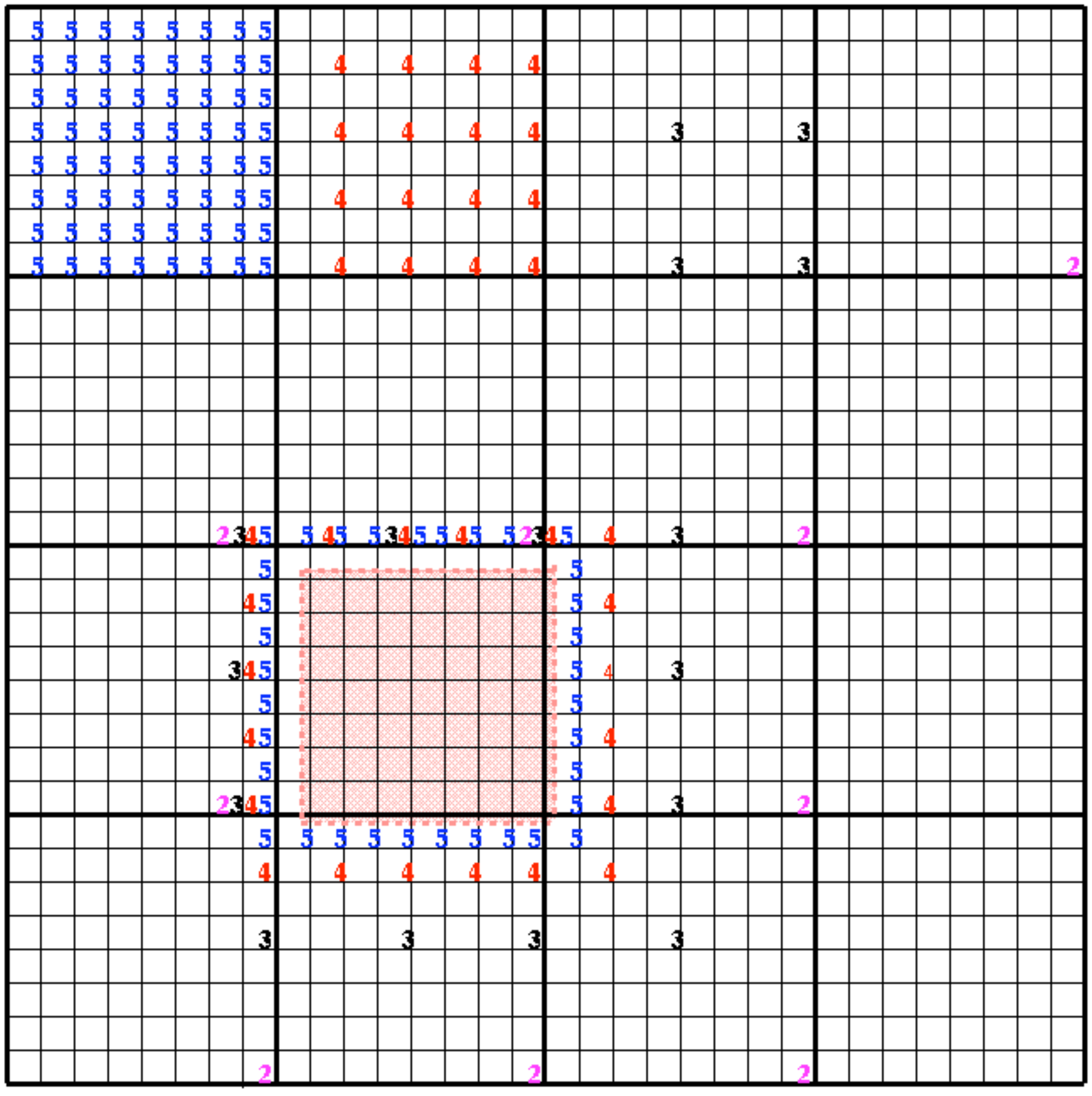
Convergence (1-D)



Convergence (2-D)



Parallelizing multigrid



Performance model (2-D)

- **Domain**: $2^m+1 \times 2^m+1$ grid
- **Processors**: $p = 4^k$, in a $2^k \times 2^k$ grid
- Each processor owns $2^{m-k} \times 2^{m-k}$ points
- Cost of one level *j* of V-cycle
  - If $j \geq k$: $O(4^{j-k})$ flops + $O(1) \cdot \alpha$ + $O(2^{j-k}) \cdot \beta^{-1}$
  - If $j < k$: $O(1)$ flops + $O(1) \cdot \alpha$ + $O(1) \cdot \beta^{-1}$
- Sum over *j* to get total V-cycle cost

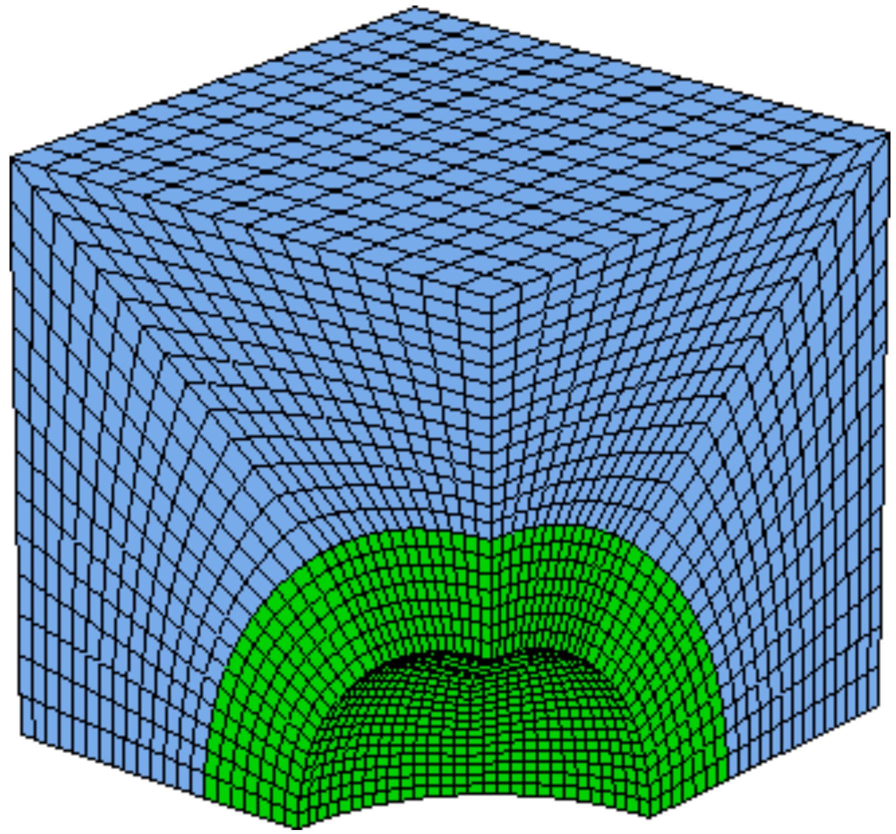
Parallel Poisson solvers compared

	Flops	Messages	Words
MG	$\frac{N}{p} + \log p \log N$	$\log^2 N$	$\sqrt{\frac{N}{p}} + \log p \log N$
FFT	$\frac{N \log N}{p}$	$\sqrt{p}$	$\frac{N}{p}$
SOR	$\frac{N^{\frac{3}{2}}}{p}$	$\sqrt{N}$	$\frac{N}{p}$

Multigrid on unstructured meshes

Multigrid on an unstructured mesh

- How to coarsen?
  - Can't just pick every other point
  - Maximal independent sets
- How to restrict? Interpolate?
- How to "smooth?"
- How to handle coarsest meshes?
  - Switch to fewer processors?
  - Switch to another solver?



“In conclusion…”