Multigrid Overview

James Demmel, Dragica Vasileska and Marco Saraniti

History of Multi-Grid Method

- 1964: first paper, Fedorenko, Russia
  - large constants: ~40,000 work units, no implementation?
- 1977: Achi Brandt, Israel, made it practical, wrote seminal paper
- late 70's: Nicolaides, Hackbusch, and others proved convergence for certain PDE's; Brandt proved fast convergence
- interest took off around 1981
- but there was (and still is) much skepticism from some because there was little theory
- today used to solve PDE's in many disciplines
- current research: a drive to achieve "textbook efficiency" for general flow simulations (all Mach numbers and Reynolds numbers)
- somewhat superseded by wavelet methods?

Literature

Brandt, 1988, The Weizmann Insitute Research in Multilevel Computation: 1988 Report, Proc. Copper Mtn. Conf. on Multigrid Methods, 1989 (53 pp.) *Survey of recent applications. I found this quite thought-provoking.*

Brandt, 1982, Guide to Multigrid Development, in Hackbusch & Trottenberg, eds., Multigrid Methods, pp. 220-312. *Guidelines for multigrid implementers. Long.*

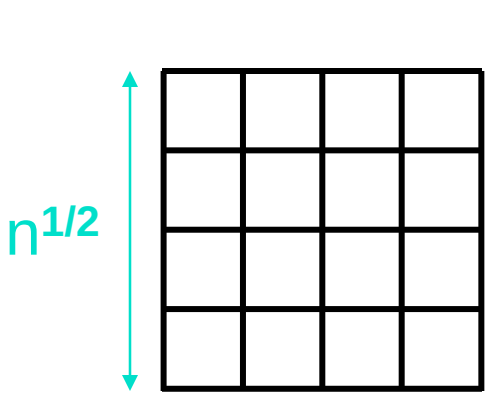
Brandt, 1997, The Gauss Center Research in Multiscale Scientific Computation, Proc. Copper Mtn. Conf. on Multigrid Methods, on web (50 pp.) http://www.wisdom.weizmann.ac.il/research.html *More esoteric than 1988 report above.*

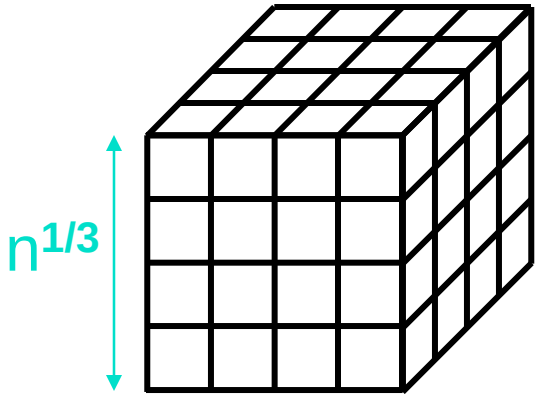
Brandt, 1980, Multilevel Adaptive Computations in Fluid Dynamics, AIAA J., vol. 18, pp. 1165-1172. *Short, fairly readable.*

Brandt, 1977, Multi-Level Adaptive Solutions to Boundary-Value Problems, Mathematics of Computation, pp. 333-390. *The seminal paper on multigrid.*

Complexity of linear solvers

Time to solve model problem (Poisson's equation) on regular mesh





	2D	3D
Sparse Cholesky:	$O(n^{1.5})$	$O(n^2)$
CG, exact arithmetic:	$O(n^2)$	$O(n^2)$
CG, no precond:	$O(n^{1.5})$	$O(n^{1.33})$
CG, modified IC:	$O(n^{1.25})$	$O(n^{1.17})$
CG, support trees:	$O(n^{1.20})$ -> $O(n^{1+})$	$O(n^{1.75})$ -> $O(n^{1.31})$
Multigrid:	$O(n)$	$O(n)$

Outline and Review

- Review Poisson equation
- Overview of Methods for Poisson Equation
- Jacobi's method
- Red-Black SOR method
- Conjugate Gradients
- FFT

Reduce to sparse-matrix-vector multiply
Need them to understand Multigrid

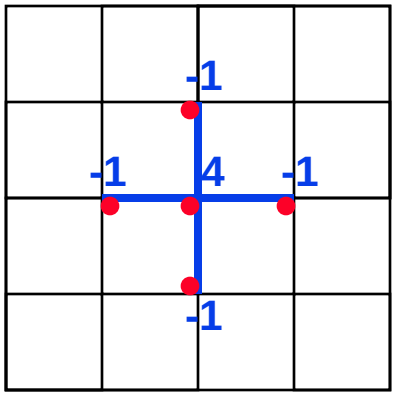
- Multigrid

2D Poisson's equation

° Similar to the 1D case, but the matrix T is now

$$T = \left(\begin{array}{ccc:ccc:ccc}
4 & -1 & & -1 & & & & & \\
-1 & 4 & -1 & & -1 & & & & \\
 & -1 & 4 & & & -1 & & & \\
\hdashline
-1 & & & 4 & -1 & & -1 & & \\
 & -1 & & -1 & 4 & -1 & & -1 & \\
 & & -1 & & -1 & 4 & & & -1 \\
\hdashline
 & & & -1 & & & 4 & -1 & \\
 & & & & -1 & & -1 & 4 & -1 \\
 & & & & & -1 & & -1 & 4
\end{array}\right)$$

Graph and "stencil"



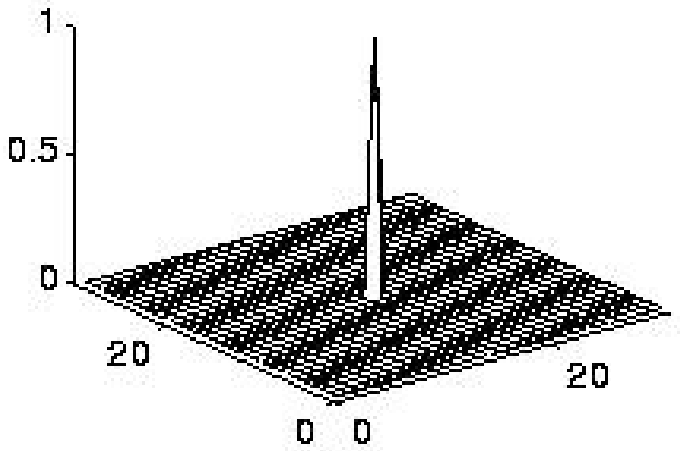
° 3D is analogous

Multigrid Motivation

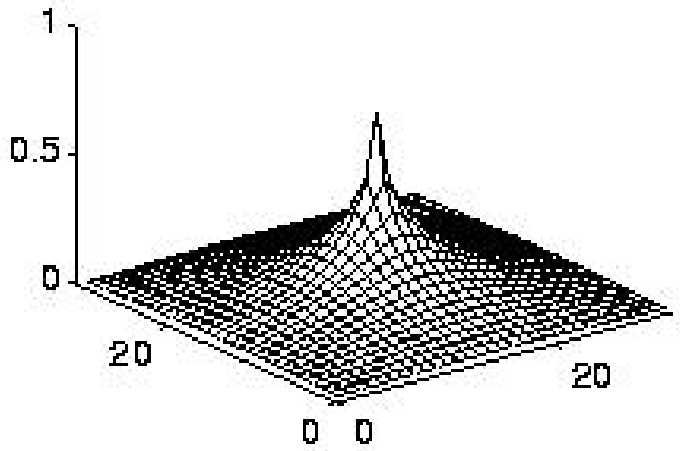
- Jacobi, SOR, CG, or any other sparse-matrix-vector-multiply-based algorithm can only move information one grid call at a time for Poisson
  - New value at each grid point depends only on neighbors
- Can show that decreasing error by fixed factor c<1 takes at least $O(n^{1/2})$ steps
  - See next slide: true solution for point source like log 1/r
- Therefore, converging in O(1) steps requires moving information across grid faster than just to neighboring grid cell per step

Multigrid Motivation

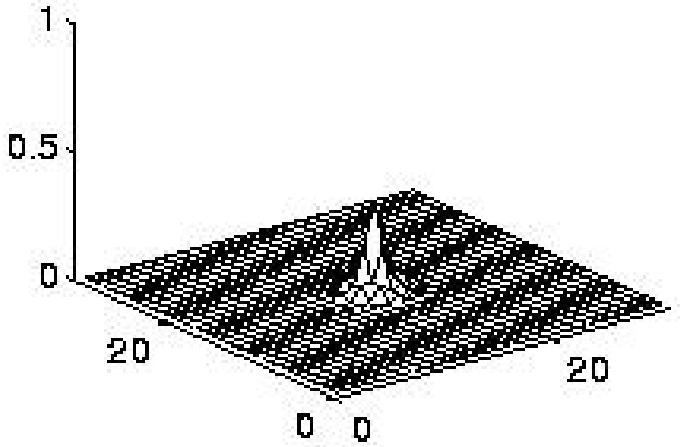
Right Hand Side



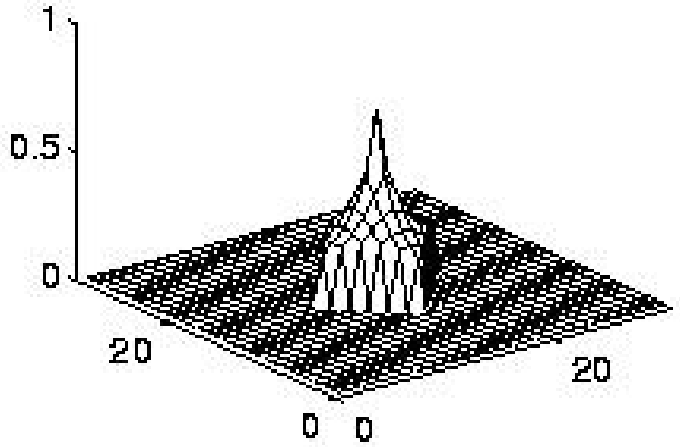
True Solution



5 steps of Jacobi



Best 5 step solution



Multigrid Overview

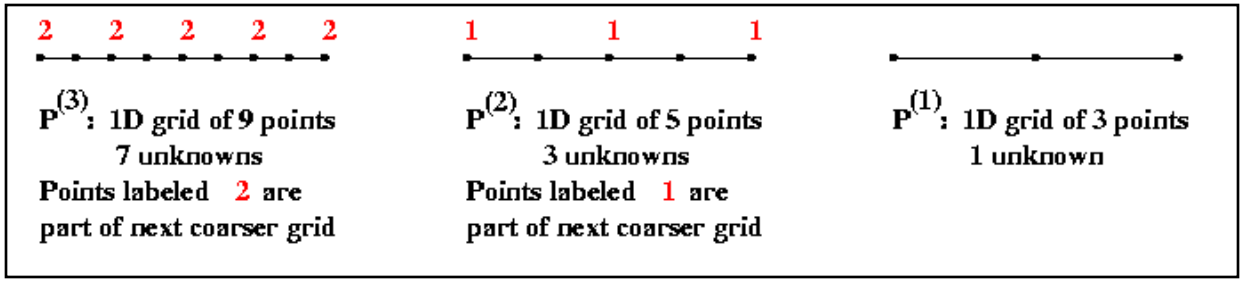
- **Basic Algorithm:**
  - **Replace problem on fine grid by an approximation on a coarser grid**
  - **Solve the coarse grid problem approximately, and use the solution as a starting guess for the fine-grid problem, which is then iteratively updated**
  - **Solve the coarse grid problem recursively, i.e. by using a still coarser grid approximation, etc.**
- **Success depends on coarse grid solution being a good approximation to the fine grid**

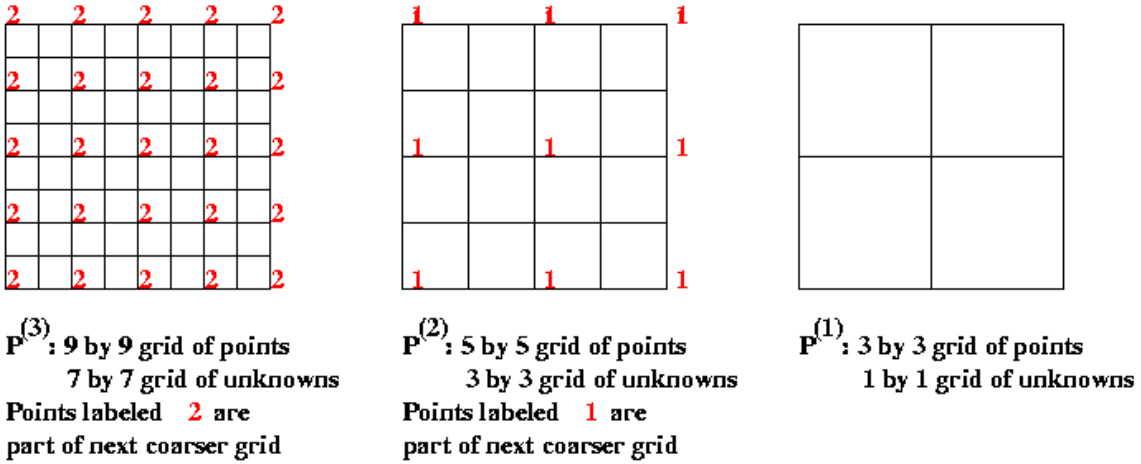
Multigrid uses Divide-and-Conquer in 2 Ways

- **First way:**
  - **Solve problem on a given grid by calling Multigrid on a coarse approximation to get a good guess to refine**
- **Second way:**
  - **Think of error as a sum of sine curves of different frequencies**
  - **Same idea as FFT solution, but not explicit in algorithm**
  - **Each call to Multgrid responsible for suppressing coefficients of sine curves of the lower half of the frequencies in the error (pictures later)**

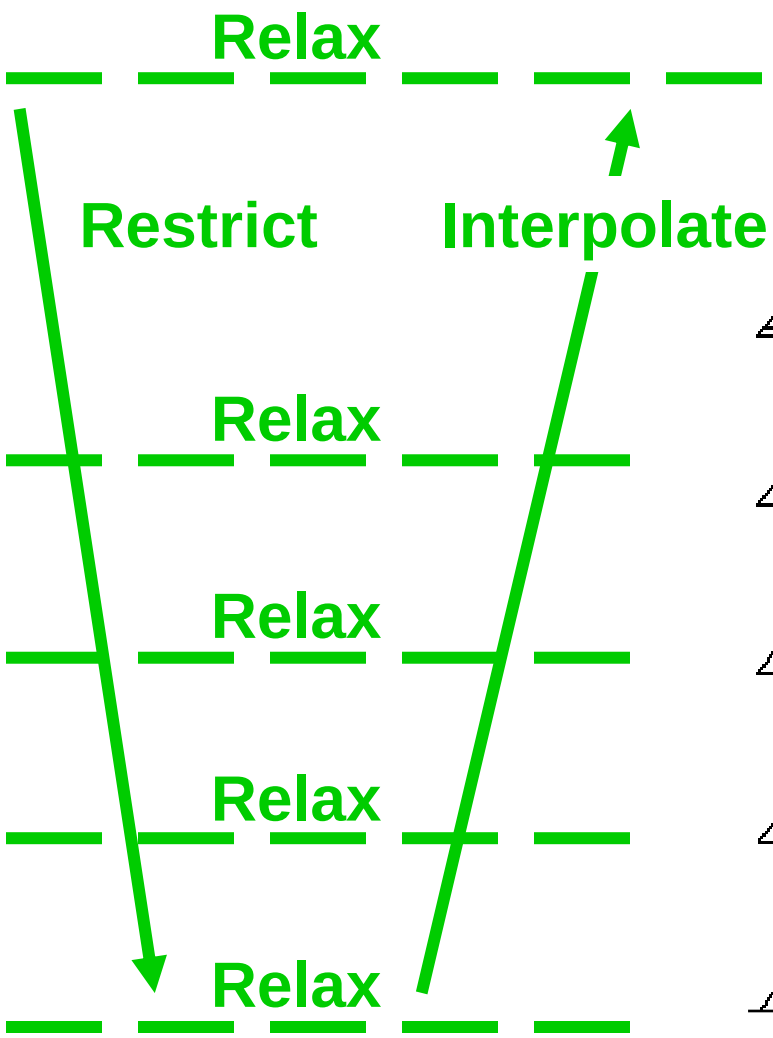
Multigrid Sketch (1D and 2D)

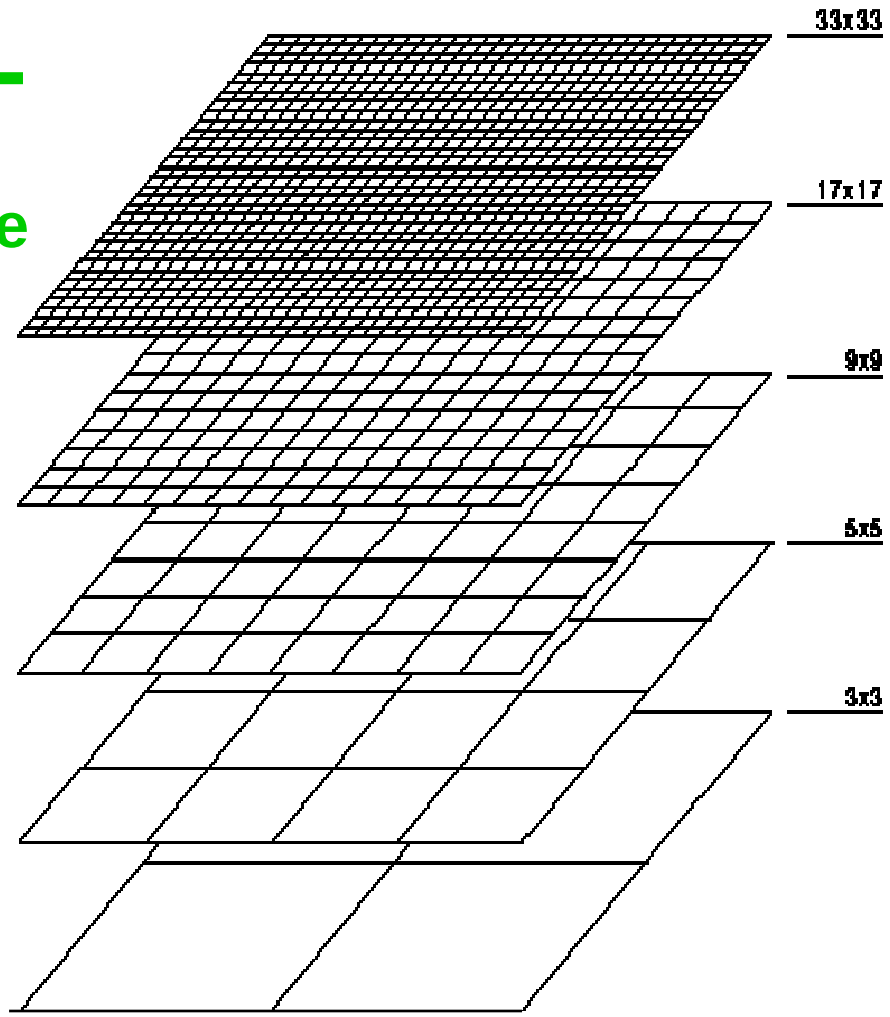
- Consider a 2^m+1 grid in 1D (2^m+1 by 2^m+1 grid in 2D) for simplicity
- Let $P^{(i)}$ be the problem of solving the discrete Poisson equation on a 2^i+1 grid in 1D (2^i+1 by 2^i+1 grid in 2D)
  - Write linear system as T(i) * x(i) = b(i)
- $P^{(m)}$, $P^{(m-1)}$, … , $P^{(1)}$ is sequence of problems from finest to coarsest





Multigrid Hierarchy





Slide source: Geoffrey Fox

Multigrid Operators

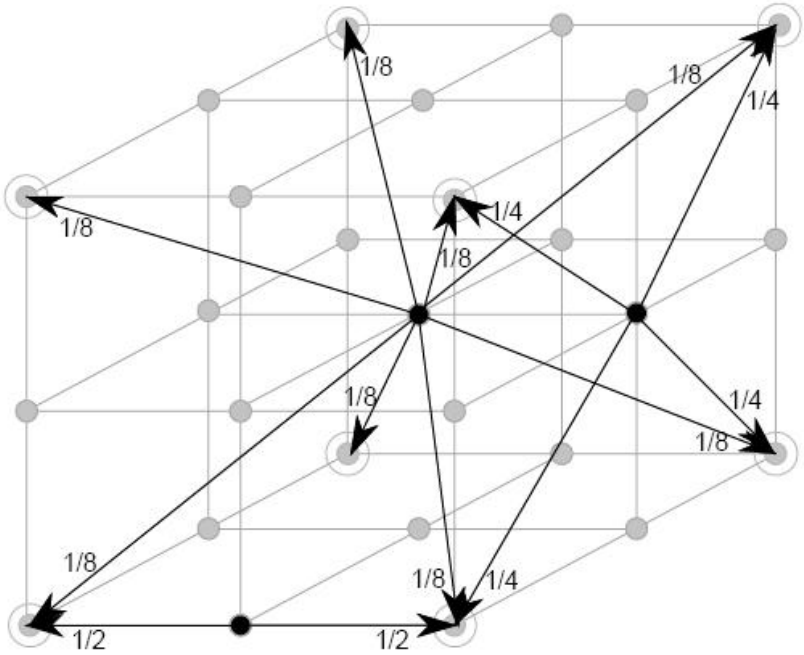
- For problem $P^{(i)}$:
  - b(i) is the RHS and
  - x(i) is the current estimated solution
  - (T(i) is implicit in the operators below.)

  } both live on grids of size 2^i-1

- All the following operators just average values on neighboring grid points
  - Neighboring grid points on coarse problems are far away in fine problems, so information moves quickly on coarse problems
- The restriction operator R(i) maps $P^{(i)}$ to $P^{(i-1)}$
  - Restricts problem on fine grid $P^{(i)}$ to coarse grid $P^{(i-1)}$ by sampling or averaging
  - b(i-1)= R(i) (b(i))
- The interpolation operator In(i-1) maps an approximate solution x(i-1) to an x(i)
  - Interpolates solution on coarse grid $P^{(i-1)}$ to fine grid $P^{(i)}$
  - x(i) = In(i-1)(x(i-1))
- The solution operator S(i) takes $P^{(i)}$ and computes an improved solution x(i) on same grid
  - Uses "weighted" Jacobi or SOR
  - $x_{improved}(i) = S(i)(b(i), x(i))$
- Details of these operators after describing overall algorithm

Trilinear interpolation between the grids

3-D Prolongation



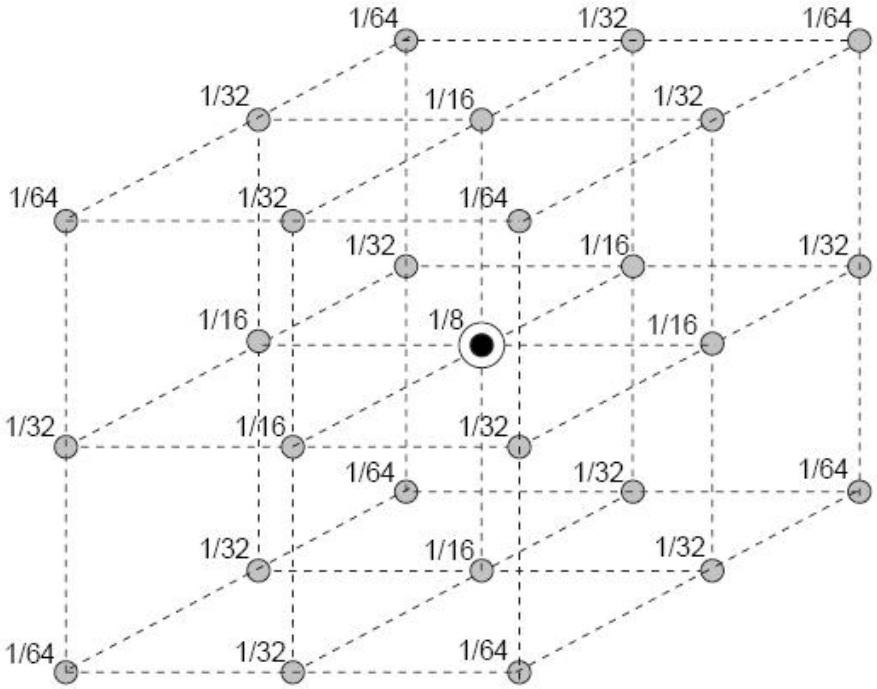
2-D Prolongation

$$\begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$$

The arrows denote the coarse grid points to be used for interpolating the dense grid point. The numbers attached to the arrows denote the contribution of the specific coarse grid point.

Ref: Can K. Sandalci, Ç. K. Coç, S. M. Goodnick: Three Dimensional Monte Carlo Device Simulation with Parallel Multigrid Solver. PPSC 1997

3-D Restriction



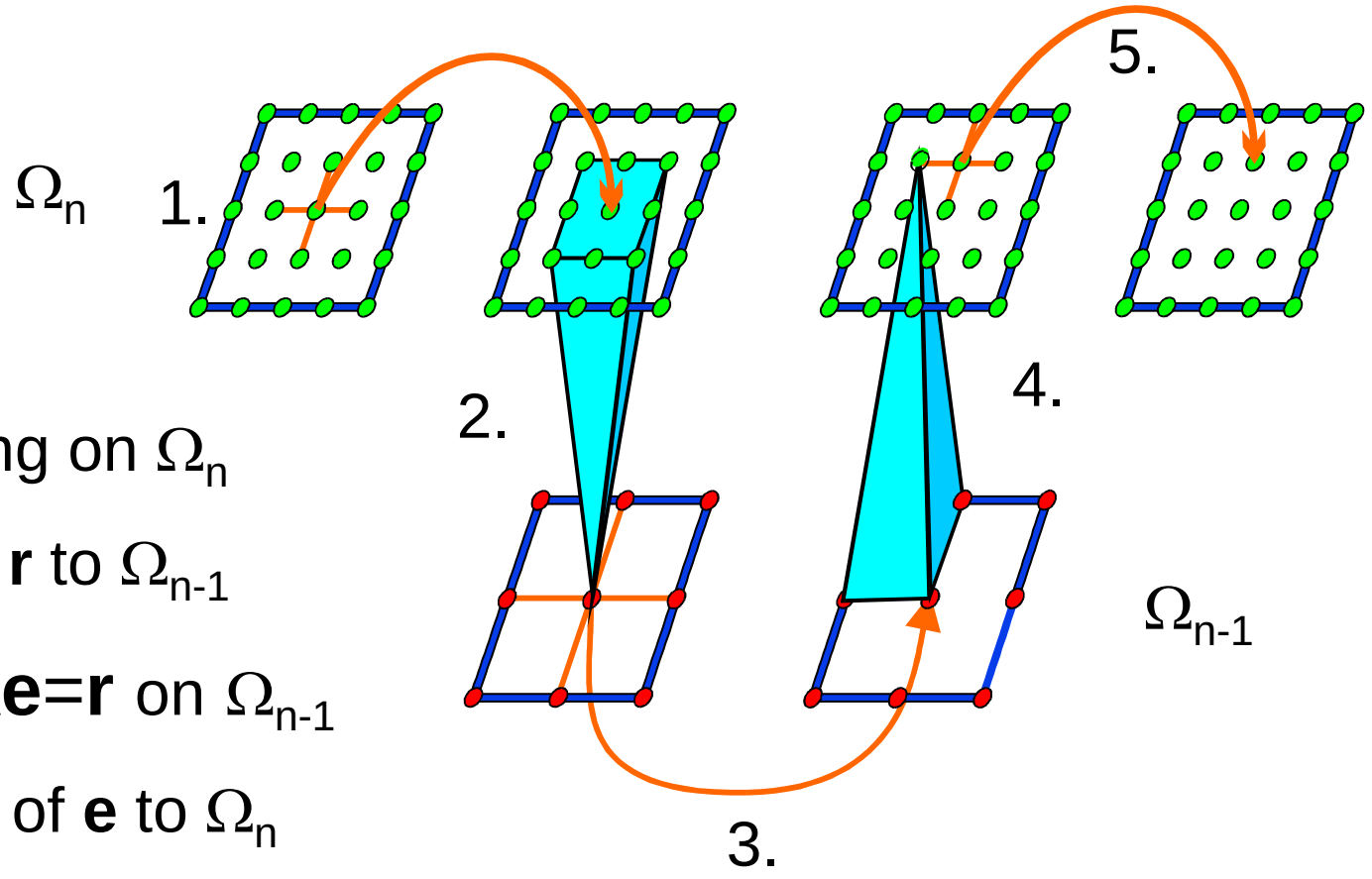
2-D Restriction

$$\begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix}$$

A 27-point full weighting scheme is used. The number in front of each grid point denotes its weight in this operation.

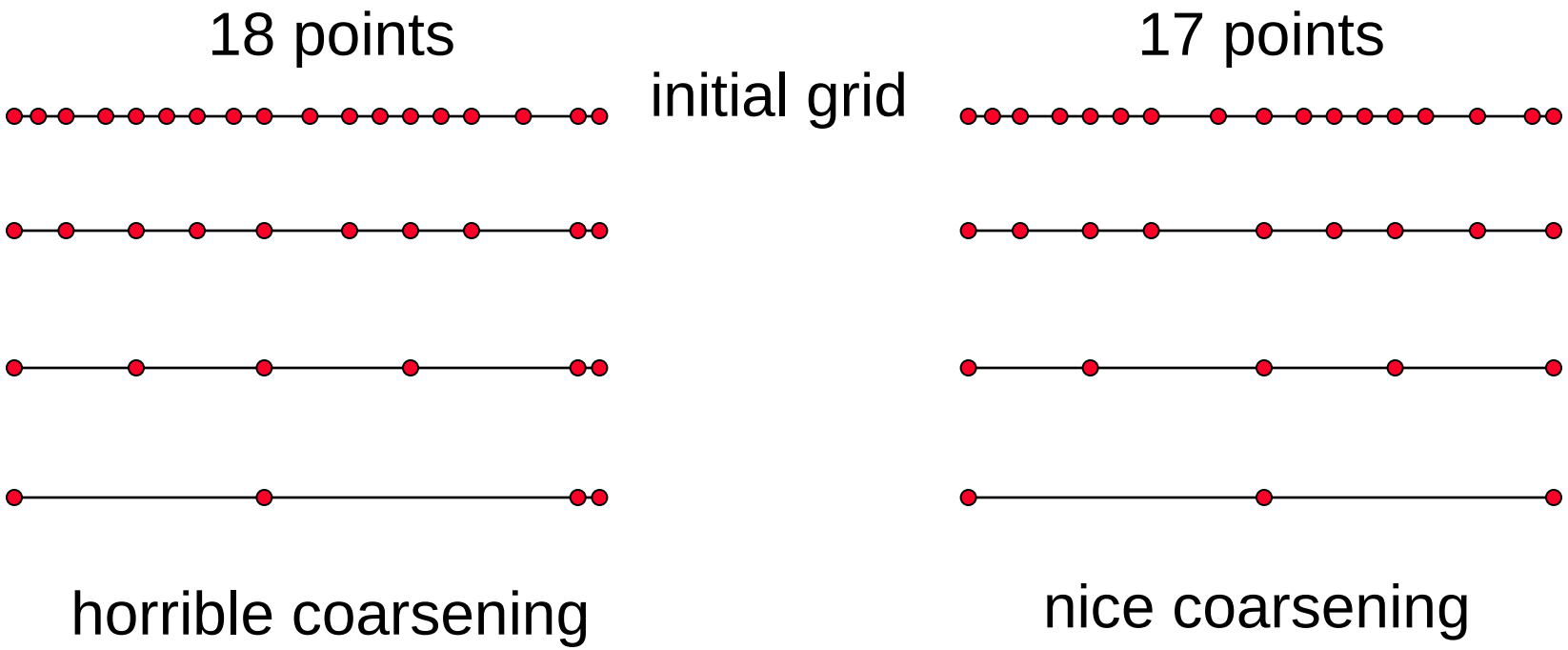
Multigrid method:

two-grid iteration to solve A**v**=**f** *on grid* Ω_n



1. pre-smoothing on Ω_n
2. restriction of **r** to Ω_{n-1}
3. solution of A**e**=**r** on Ω_{n-1}
4. prolongation of **e** to Ω_n
5. post-smoothing on Ω_n

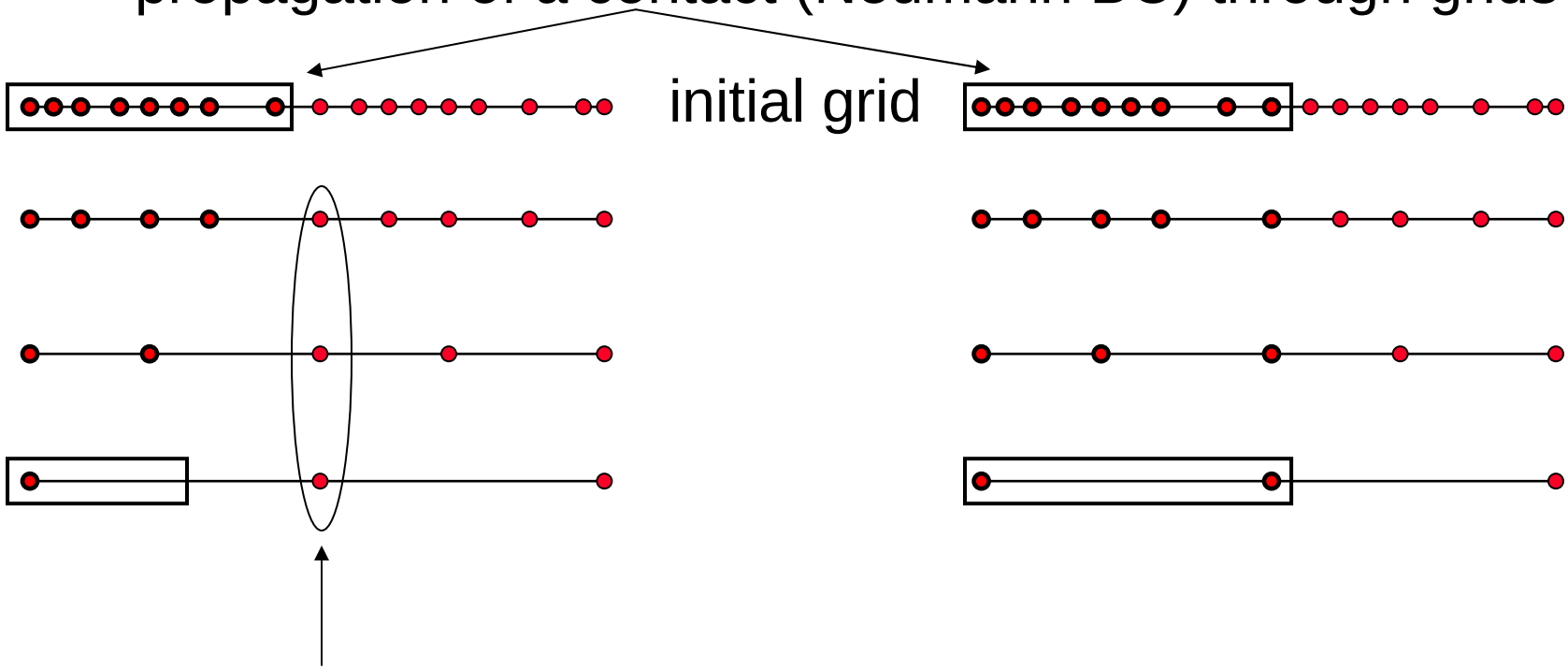
Coarsening techniques: importance of boundaries I



When the number of points in one dimension is 2^N+2 (N being a natural number), a geometric mismatch is generated in the coarser grids, which show pronounced in-homogeneity. The convergence of the method is severely slowed down in these cases.

Coarsening techniques: importance of boundaries II

propagation of a contact (Neumann BC) through grids

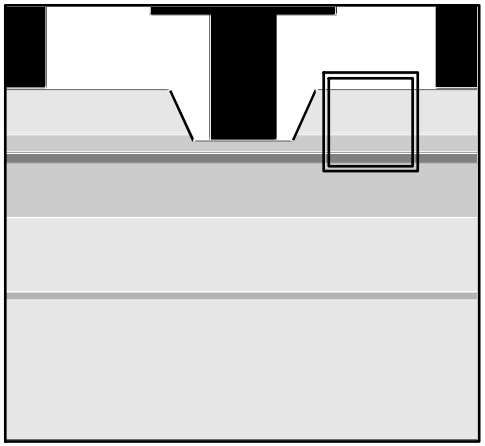


post-smoothing often <u>required</u> on these points

no post-smoothing is required

Coarsening techniques: simple application

Grid	points	X-points	Y-points	coarsening
Ω_7	22962	258	89	X,Y
Ω_6	5850	130	45	X,Y
Ω_5	1518	66	23	X,Y
Ω_4	408	34	12	X,Y
Ω_3	126	18	7	X,Y
Ω_2	40	10	4	X
Ω_1	24	6	4	X
Ω_0	16	4	4	-

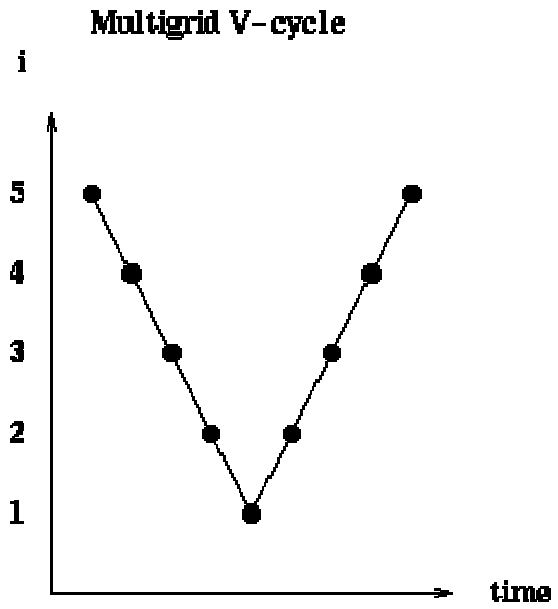


Multigrid V-Cycle Algorithm

```
Function MGV ( b(i), x(i) )
  … Solve T(i)*x(i) = b(i) given b(i) and an initial  guess for x(i)
  … return an improved x(i)
  if (i = 1)
      compute exact solution x(1) of P(1)        only 1 unknown
      return x(1)
  else
      x(i) = S(i) (b(i), x(i))                   improve solution by
                                                 damping high frequency error
      r(i)  = T(i)*x(i) - b(i)                   compute residual
      d(i) = In(i-1) ( MGV( R(i) ( r(i) ), 0 ) ) solve T(i)*d(i) = r(i) recursively
      x(i) = x(i) - d(i)                         correct fine grid solution
      x(i) = S(i) ( b(i), x(i) )                 improve solution again
      return x(i)
```

Why is this called a V-Cycle?

- Just a picture of the call graph
- In time a V-cycle looks like the following



Complexity of a V-Cycle on a Serial Machine

- Work at each point in a V-cycle is O(# unknowns)
- Cost of Level i is $(2^i-1)^2 = O(4^i)$
- If finest grid level is m, total time is:

$$\sum_{i=1}^{m} O(4^i) = O(4^m) = O(\text{\# unknowns})$$

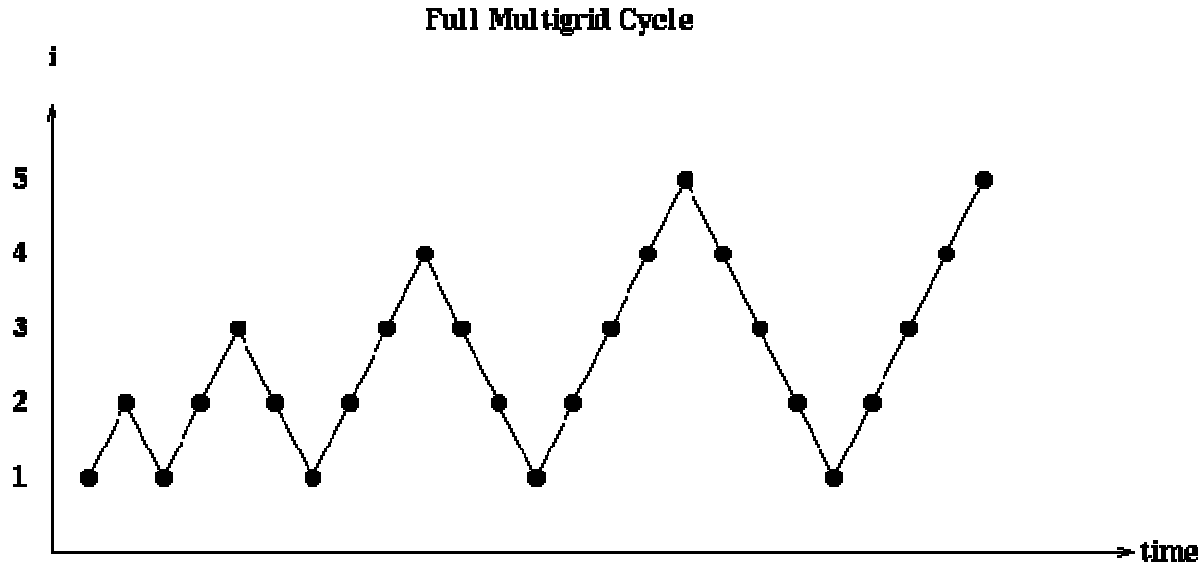
Full Multigrid (FMG)

- **Intuition:**
  - improve solution by doing multiple V-cycles
  - avoid expensive fine-grid (high frequency) cycles

```
Function FMG (b(m), x(m))
    … return improved x(m) given initial guess
    compute the exact solution x(1) of P(1)
    for i=2 to m
        x(i) = MGV ( b(i), In (i-1) (x(i-1) ) )
```

- **In other words:**
  - Solve the problem with 1 unknown
  - Given a solution to the coarser problem, $P^{(i-1)}$, map it to starting guess for $P^{(i)}$
  - Solve the finer problem using the Multigrid V-cycle

Full Multigrid Cost Analysis



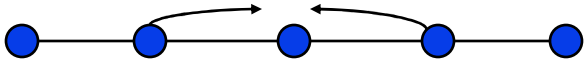
° One V for each call to FMG

- people also use Ws and other compositions

° Serial time: $\sum_{i=1}^{m} O(4^i) = O(4^m) = O(\#\text{ unknowns})$

The Solution Operator S(i) - Details

- The solution operator, S(i), is a weighted Jacobi
- Consider the 1D problem



- At level i, pure Jacobi replaces:

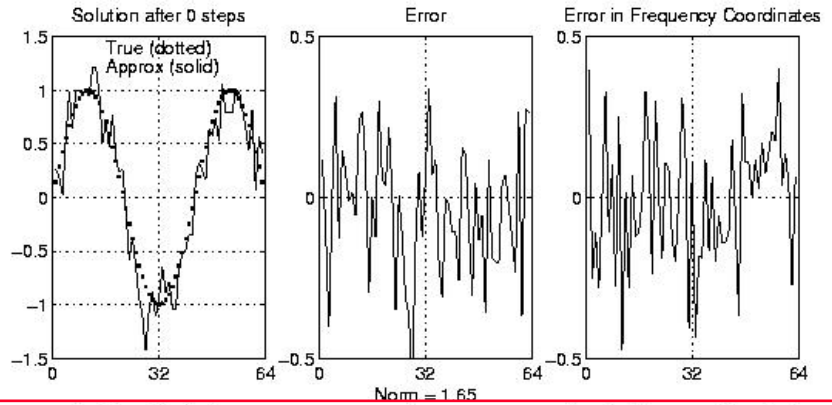
  x(j) := 1/2 (x(j-1) + x(j+1) + b(j))

- Weighted Jacobi uses:

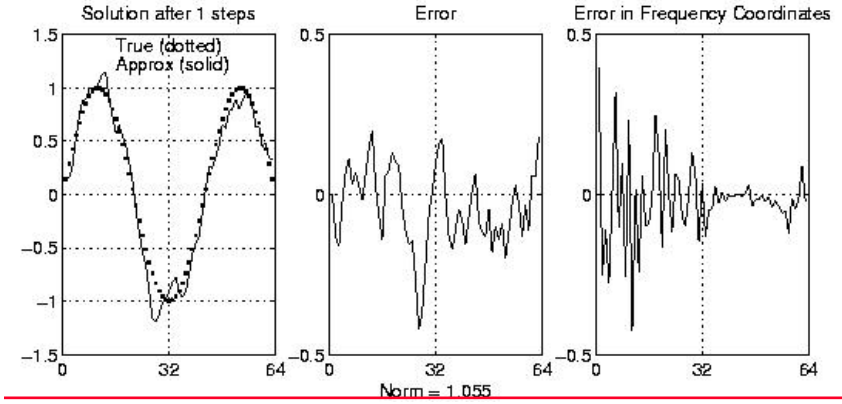
  x(j) := 1/3 (x(j-1) + x(j) + x(j+1) + b(j))

- In 2D, similar average of nearest neighbors

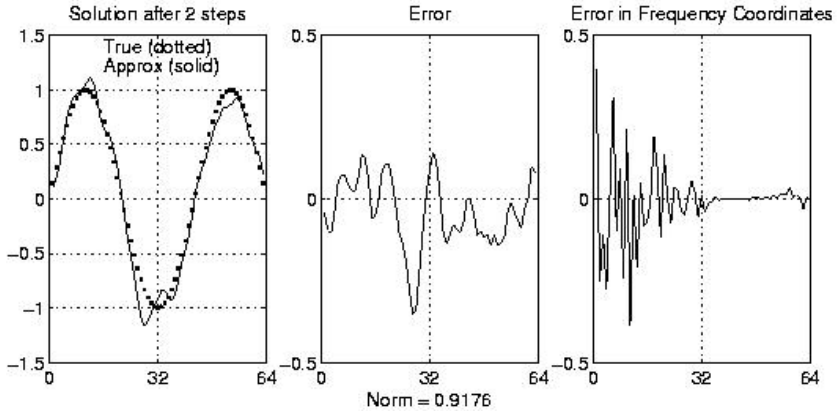
Weighted Jacobi chosen to damp high frequency error



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



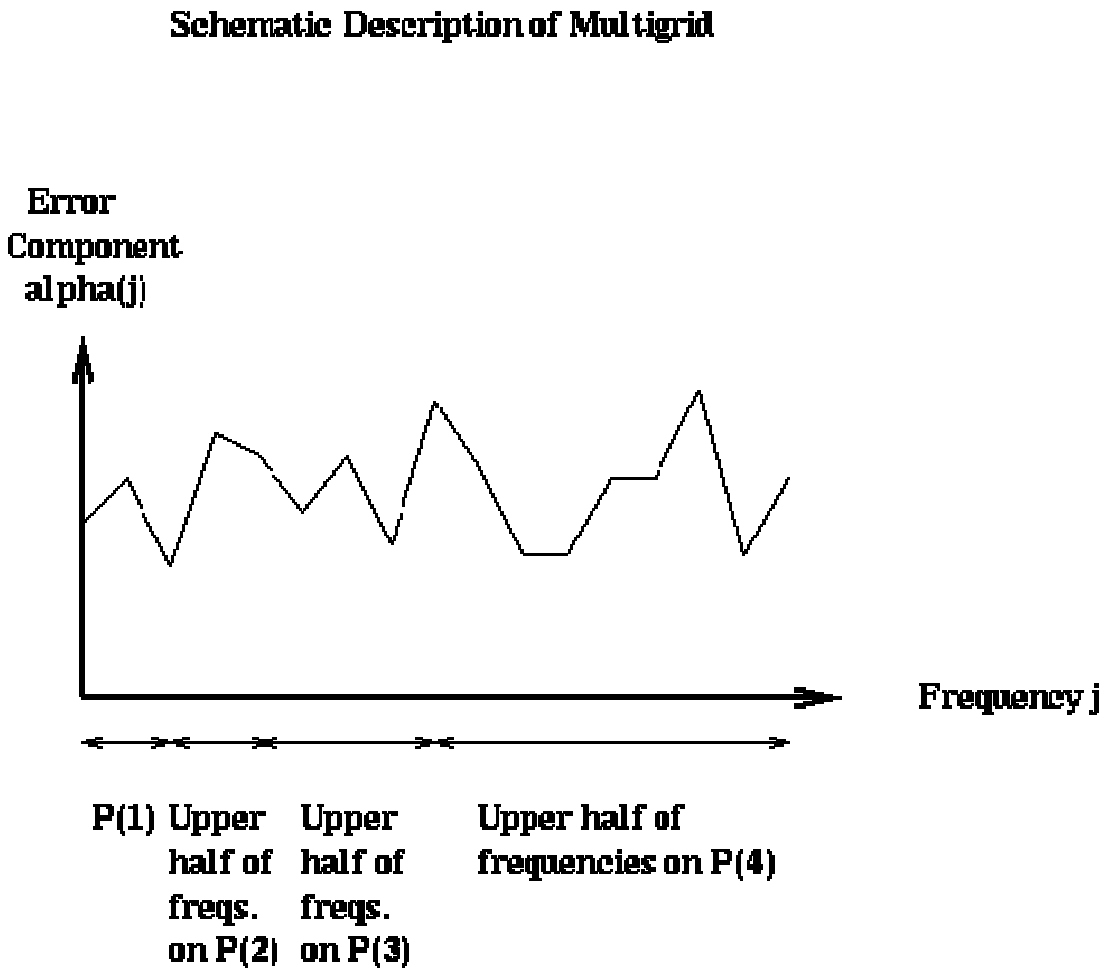
Error after 1 Jacobi step
- **"Smoother"**
- **Less high frequency component**
- **Norm = 1.055**



Error after 2 Jacobi steps
- **"Smooth"**
- **Little high frequency component**
- **Norm = .9176,**
  won't decrease much more

Multigrid as Divide and Conquer Algorithm

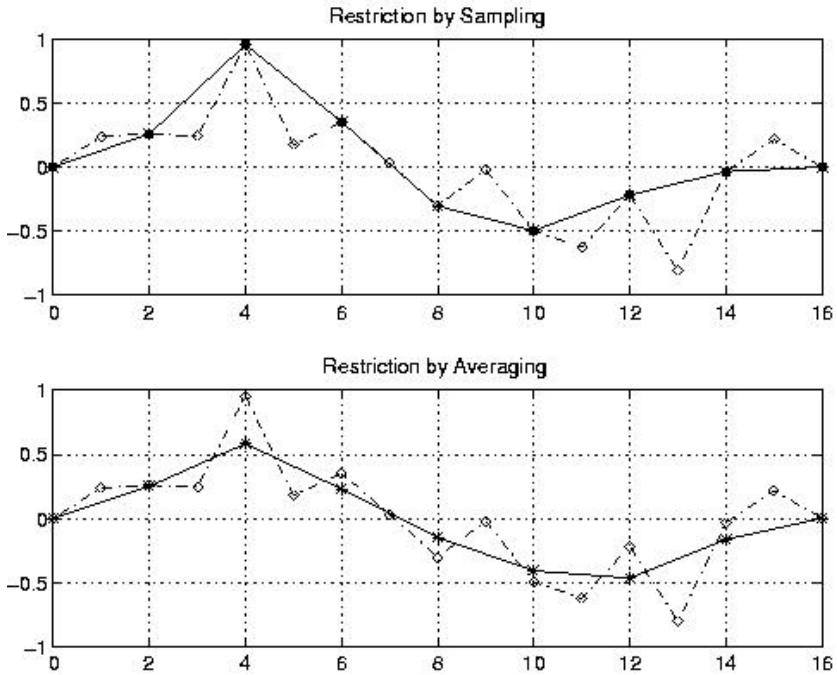
- Each level in a V-Cycle reduces the error in one part of the frequency domain



The Restriction Operator R(i) - Details

- The restriction operator, R(i), takes
  - a problem $P^{(i)}$ with RHS b(i) and
  - maps it to a coarser problem $P^{(i-1)}$ with RHS b(i-1)

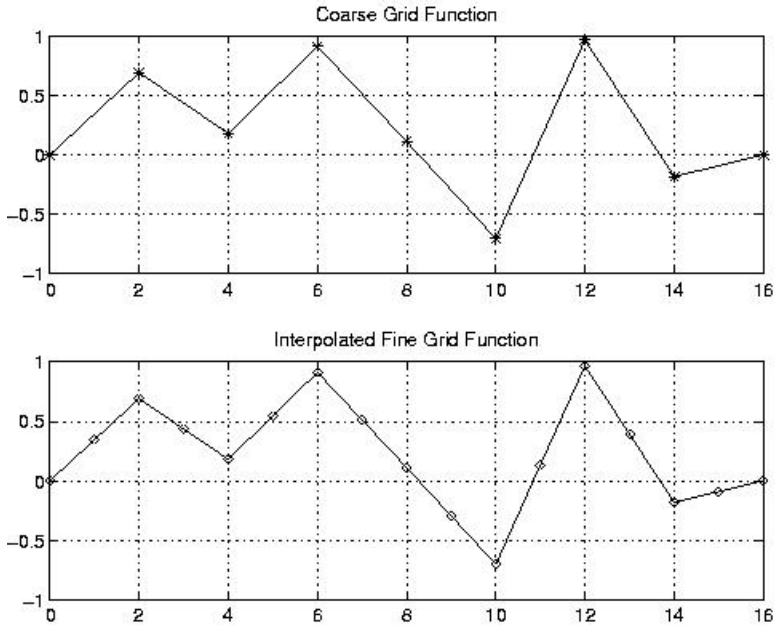
- In 1D, average values of neighbors
  - $x_{coarse}(i) = 1/4 * x_{fine}(i-1) + 1/2 * x_{fine}(i) + 1/4 * x_{fine}(i+1)$



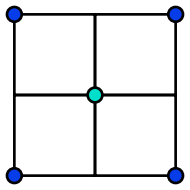
- In 2D, average with all 8 neighbors (N,S,E,W,NE,NW,SE,SW)

Interpolation Operator

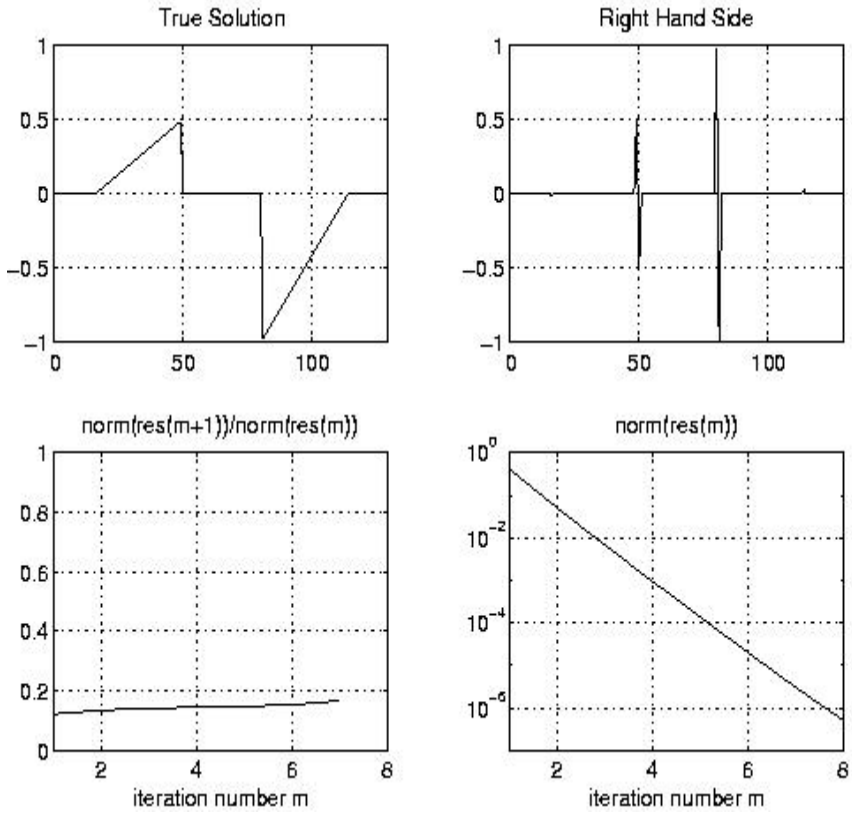
- The interpolation operator In(i-1), takes a function on a coarse grid $P^{(i-1)}$, and produces a function on a fine grid $P^{(i)}$
- In 1D, linearly interpolate nearest coarse neighbors
  - $x_{fine}(i) = x_{coarse}(i)$ if the fine grid point i is also a coarse one, else
  - $x_{fine}(i) = 1/2 * x_{coarse}$(left of i) + $1/2 * x_{coarse}$(right of i)

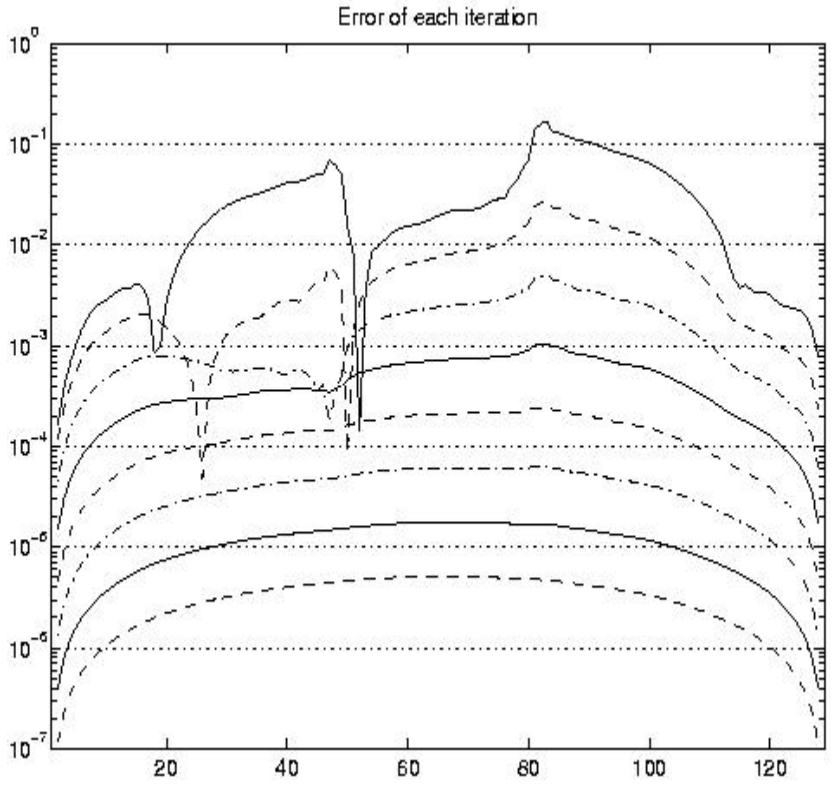


- In 2D, interpolation requires averaging with 2 or 4 nearest neighbors (NW,SW,NE,SE)

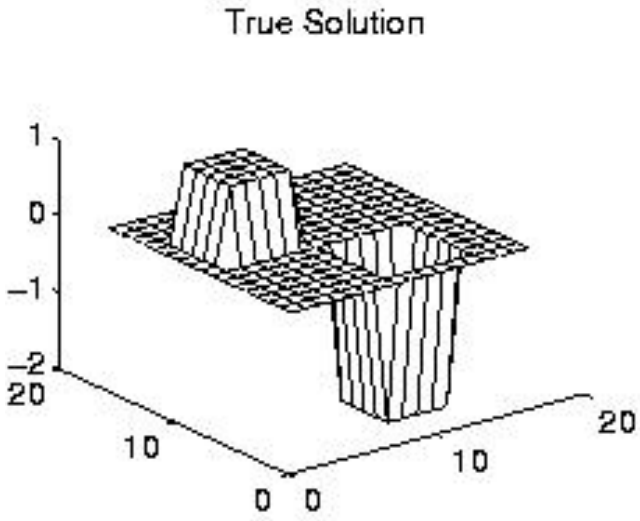


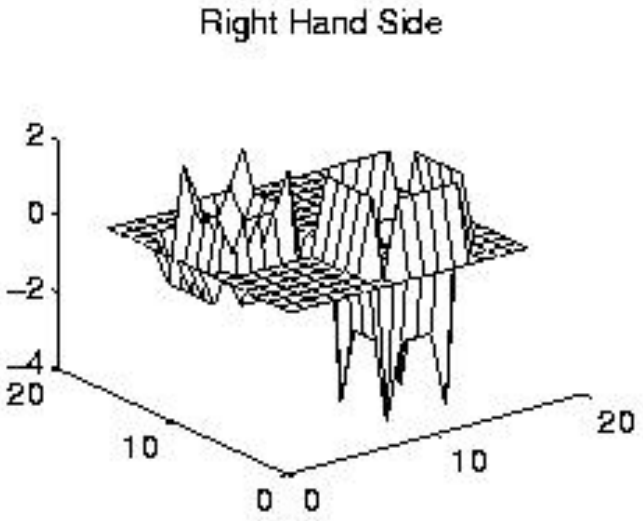
Convergence Picture of Multigrid in 1D

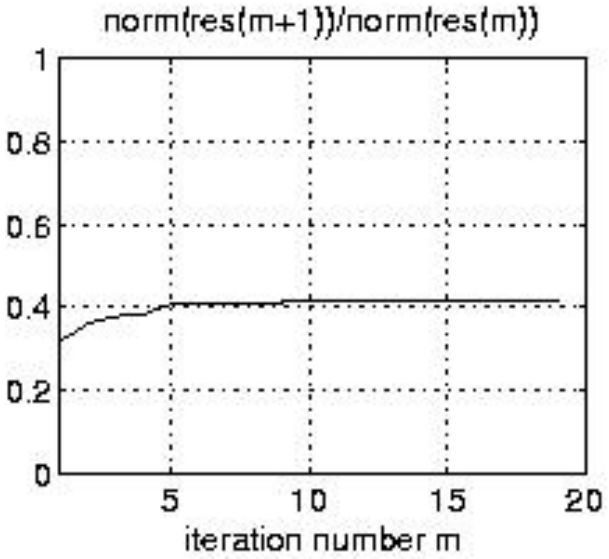


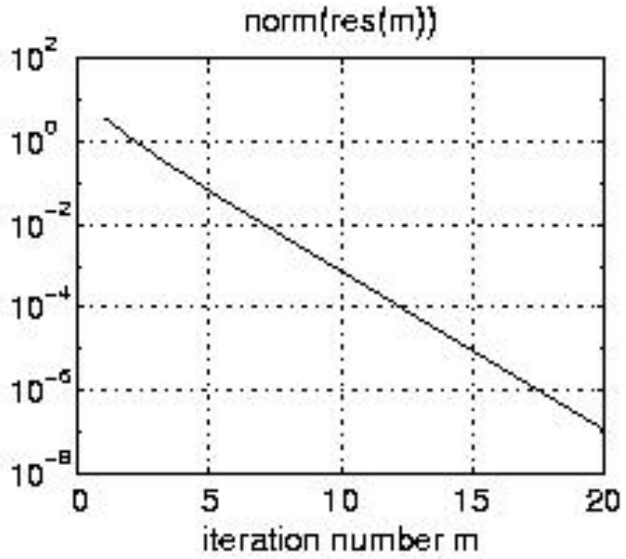


Convergence Picture of Multigrid in 2D







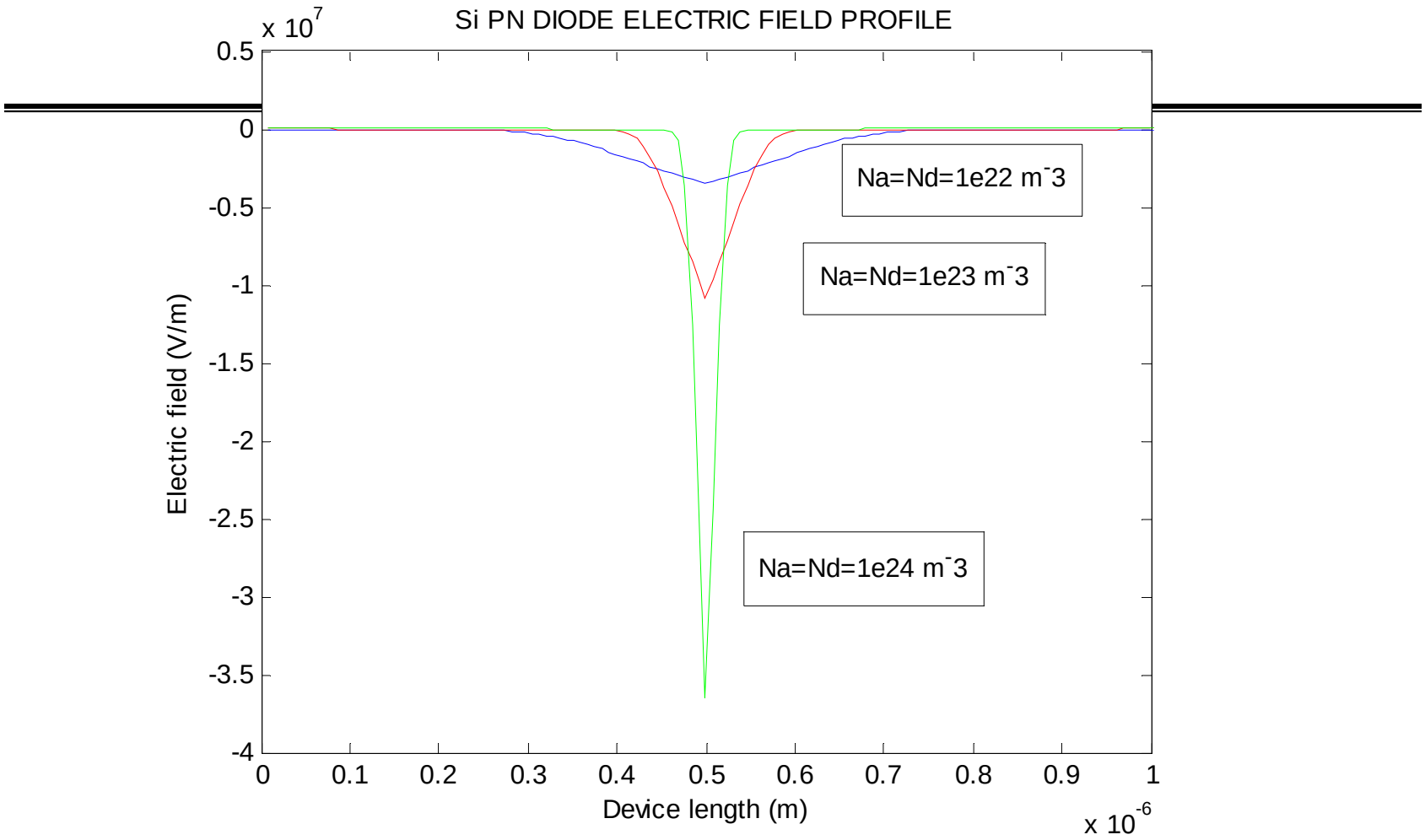


Equilibrium Simulation Results

PN diode	Grid points
Grid level 6	**129x129 x129**
5	**65 x 65 x 65**
4	**33 x 33 x 33**
3	**17 x17 x 17**
2	**9 x 9 x 9**
1	**5 x 5 x 5**

Na (m^-3)	**1.00E+22**	**1.00E+23**	**1.00E+24**
Nd (m^-3)	**1.00E+22**	**1.00E+23**	**1.00E+24**
Depletion Width (m)	**0.601E-06**	**0.206E-06**	**0.069E-06**

Uniform grid organization on six levels (top panel). Comparison of depletion widths for different doping (bottom panel).



Na(m^-3)	**1.00E+22**	**1.00E+23**	**1.00E+24**
Nd(m^-3)	**1.00E+22**	**1.00E+23**	**1.00E+24**
Electric field(V/m)	**-3.269E6**	**-1.1186E7**	**-3.7866E7**