EE5780 Advanced VLSI CAD

Lecture 7

Fast Algorithms for IC modeling and analysis

Zhuo Feng

- We've talked about using LU to solve a sparse linear matrix problem
- Another way of dealing with sparse matrix problems is to solve them iteratively
- There are no fill-ins with iterative solution methods!
- But the challenge is ensuring efficient convergence

- **Iterative methods are great for sparse system storage – if they converge**

$$A\vec{x} = \vec{b}$$

- **Most popular iterative methods are:**
  - **Gauss-Jacobi**
  - **Gauss-Seidel**

- **We will talk about some more complicated methods later**

$$A\vec{x} = \vec{b}$$

$$A\vec{x} + \vec{x} - \vec{x} = \vec{b}$$

$$\vec{x} = (1 - A)\vec{x} + \vec{b}$$

- Iteratively solve for $\vec{x}$

$$\vec{x}_{i+1} = \underbrace{(1 - A)}_{B}\vec{x}_i + \vec{b}$$

- Define: $\vec{\varepsilon}_i = \vec{x}_i - \vec{x}_t$ ← True solution

$$\text{and } \vec{\varepsilon}_{i+1} = \vec{x}_{i+1} - \vec{x}_t$$

- $\vec{\varepsilon}_{i+1} = B\vec{x}_i + \vec{b} - \left(B\vec{x}_t + \vec{b}\right) = B\vec{\varepsilon}_i$

- For an initial guess $\vec{x}_1$

$$\vec{\varepsilon}_{i+1} = B^i\left(\vec{x}_1 - \vec{x}_t\right)$$

- It follows that a necessary and sufficient condition for convergence is:

$$\lim_{i\to\infty}\left(B^i\vec{y}\right) = \vec{0} \quad \text{for all} \quad \vec{y}$$

- **OR:** $\lim_{i\to\infty}\left\|B^i\right\| = 0$

- The eigenvalues of B^i are the i-th powers of the eigenvalues of B

- Another necessary and sufficient condition for convergence is that all of the eigenvalues of B have a magnitude less than 1 (all eigenvalues lie within the unit circle)

- We also want $\|B\|$ to be small so that iterations converge rapidly

- Calculating eigenvalues is more difficult than solving original problem!

- Matrix Norms:

$$\|B\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|b_{ij}|^2}$$

$$\|B\|_1 = \max_{1\le j\le n}\sum_{i=1}^{n}|b_{ij}|$$

$$\|B\|_\infty = \max_{1\le i\le n}\sum_{j=1}^{n}|b_{ij}|$$

- Euclidean norm is fairly high complexity, therefore ∞-norm is most commonly used

- But ∞-norm is sufficient but NOT necessary

- ∞-norm <1 **follows from diagonal dominance condition**

  - **Sufficient but not necessary:**

$$A\vec{x} = \vec{b} \qquad \left\| 1 - A \right\|_{\infty} < 1 \quad ?$$

$$\left\| a_{ii} \right\| > \sum_{\substack{j=1 \\ j \neq i}}^{n} \left\| a_{ij} \right\| \quad \text{for} \quad i = 1,2 \ldots n$$

- Normalize each row of A by diagonal element

$$\begin{bmatrix} 1 & \frac{a_{12}}{a_{11}} & \frac{a_{13}}{a_{11}} & \cdots \\ \frac{a_{21}}{a_{22}} & 1 & \frac{a_{23}}{a_{22}} & \cdots \\ \vdots & & & \ddots \\ & & & \end{bmatrix}$$

- $B = 1 - A$

$$-\begin{bmatrix} 0 & \frac{a_{12}}{a_{11}} & \frac{a_{13}}{a_{11}} & \cdots \\ \frac{a_{21}}{a_{22}} & 0 & \frac{a_{23}}{a_{22}} & \cdots \\ \vdots & & & \ddots \end{bmatrix}$$

- $\|B\|_\infty < 1$ **if** $\sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{\|a_{ij}\|}{\|a_{ii}\|} < 1$

- **Gauss - Jacobi** $A\vec{x} = \vec{b}$

$k \leftarrow 0;$

guess $\vec{x}^0$

repeat $\{$

$k \leftarrow k+1;$

for all $(i \in \{1,2,\ldots n\})$

$$x_i^k = \frac{1}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k-1} - \sum_{j=i+1}^{n} a_{ij}x_j^{k-1}\right];$$

$\}$ until $\left(\left|x_i^k - x_i^{k-1}\right| \le \varepsilon,\ i = 1,2\ldots n\right)$;

Only makes sense for parallel processing applications – otherwise Gauss-Seidel is faster

- Gauss - Seidel

$k \leftarrow 0;$

guess $\vec{x}^0$

repeat $\{$

$k \leftarrow k+1;$

for all $(i \in \{1,2,\ldots n\})$

$$x_i^k = \frac{1}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^k - \sum_{j=i+1}^{n} a_{ij} x_j^{k-1} \right];$$

$\}$ until $\left(\left| x_i^k - x_i^{k-1} \right| \le \varepsilon,\ i = 1,2\ldots n \right);$

- Successive overrelaxation (SOR) can improve the rate of convergence "sometimes"
- Use G-S to calculate $\tilde{\tilde{x}}_i^{k+1}$ -- K+1th G-S iteration
  - But use weighted average of it and $\tilde{\tilde{x}}_i^{k}$ for actual update

$$\tilde{\tilde{x}}_i^{k+1} = (1-\omega)x_i^k + \omega\tilde{x}_i^{k+1}$$

- Selection of ω is nontrivial!!

- **Matrix interpretations**

$$A = D + L + U$$

Diagonal — Lower Triangular — Upper Triangular

- **Guass-Jacobi**

$$D\vec{x}^{k+1} + (L+U)\vec{x}^{k} = \vec{b} \Rightarrow \vec{x}^{k+1} = -D^{-1}(L+U)\vec{x}^{k} + D^{-1}\vec{b}$$

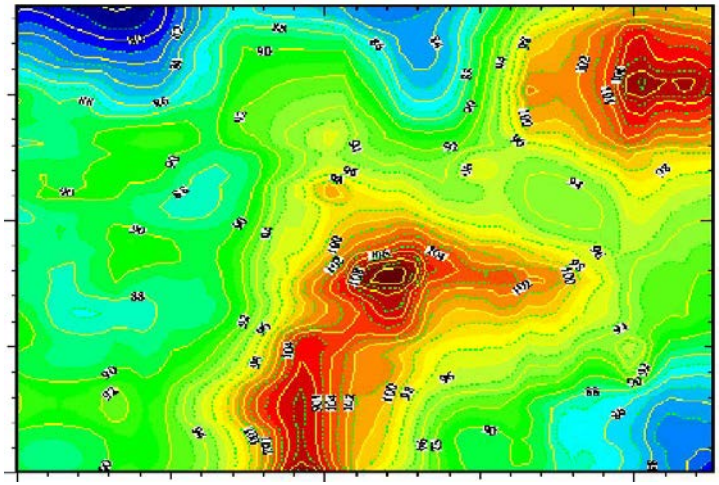
- **Guass-Seidel**

$$(D+L)\vec{x}^{k+1} + U\vec{x}^{k} = \vec{b} \Rightarrow \vec{x}^{k+1} = -(D+L)^{-1}U\vec{x}^{k} + (D+L)^{-1}\vec{b}$$

How would the iterative solution methods apply to some problems other than circuits?

Thermal analysis of ICs is becoming extremely important

Temperature gradients on a chip can substantially impact the performance



- **Heat conduction is governed by the following PDE:**

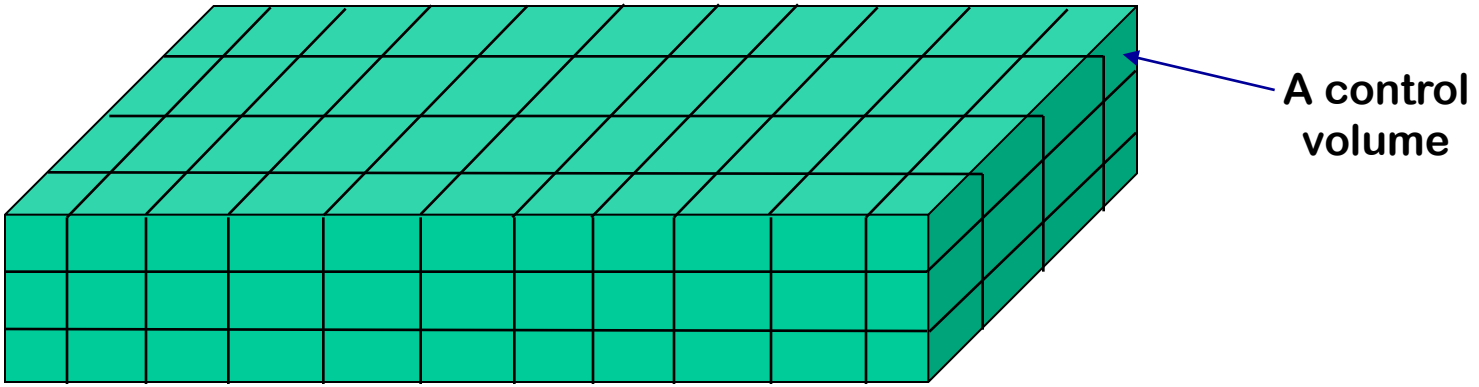
$$\rho C_p \frac{\partial T(x,y,z,t)}{\partial t} = \nabla \cdot [\kappa \nabla T(x,y,z,t)] + p(x,y,z,t)$$

Material density

Specific heat

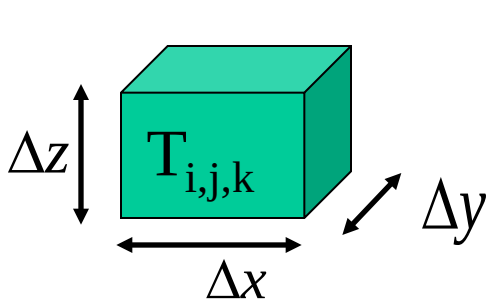
Thermal conductivity

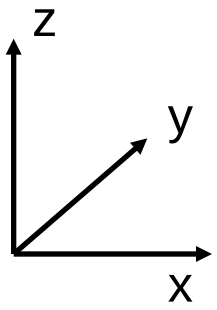
Power density of heat sources



- **The PDE can be numerically solved using finite element/difference discretization**

- **A control volume**





- **Discretize the PDE over all control volumes**

$$\rho C_p \frac{\partial T(i,j,k,t)}{\partial t} = \kappa\left[\frac{\partial^2 T(i,j,k,t)}{\partial x^2} + \frac{\partial^2 T(i,j,k,t)}{\partial y^2} + \frac{\partial^2 T(i,j,k,t)}{\partial z^2}\right] + p(i,j,k,t)$$

$$\frac{\dfrac{\{T(i+1,j,k,t)-T(i,j,k,t)\}}{\Delta x} - \dfrac{\{T(i,j,k,t)-T(i-1,j,k,t)\}}{\Delta x}}{\Delta x}$$

Rewrite the finite difference discretization

$$C\frac{d}{dt}T(i,j,k)+G_x(T(i,j,k)-T(i+1,j,k))+G_x(T(i,j,k)-T(i-1,j,k))$$
$$+G_y(T(i,j,k)-T(i,j+1,k))+G_y(T(i,j,k)-T(i,j-1,k))$$
$$+G_z(T(i,j,k)-T(i,j,k+1))+G_z(T(i,j,k)-T(i,j,k-1))=I(i,j,k)$$

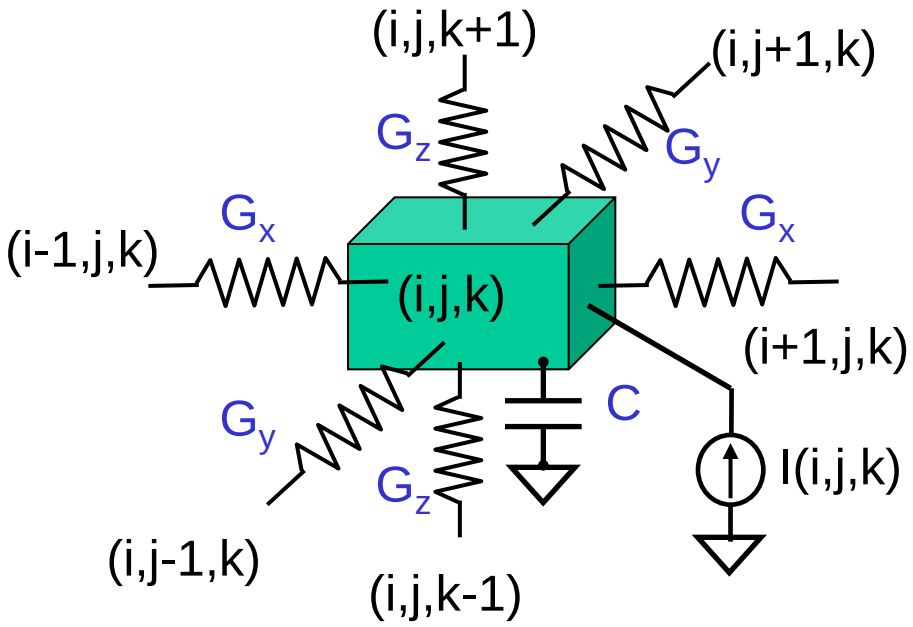
where:

$$G_x=\frac{\kappa(\Delta y\cdot\Delta z)}{\Delta x},G_y=\frac{\kappa(\Delta x\cdot\Delta z)}{\Delta y},G_z=\frac{\kappa(\Delta x\cdot\Delta y)}{\Delta z}$$

$$C=\rho C_p\Delta x\cdot\Delta y\cdot\Delta z$$

$$I(i,j,k)=p(i,j,k)\Delta x\cdot\Delta y\cdot\Delta z$$

- **Translates into a linear circuit of thermal resistance, capacitance, and heat sources**



$$C\frac{d}{dt}T(i,j,k)+G_x(T(i,j,k)-T(i+1,j,k))+G_x(T(i,j,k)-T(i-1,j,k))$$
$$+G_y(T(i,j,k)-T(i,j+1,k))+G_y(T(i,j,k)-T(i,j-1,k))$$
$$+G_z(T(i,j,k)-T(i,j,k+1))+G_z(T(i,j,k)-T(i,j,k-1))=I(i,j,k)$$

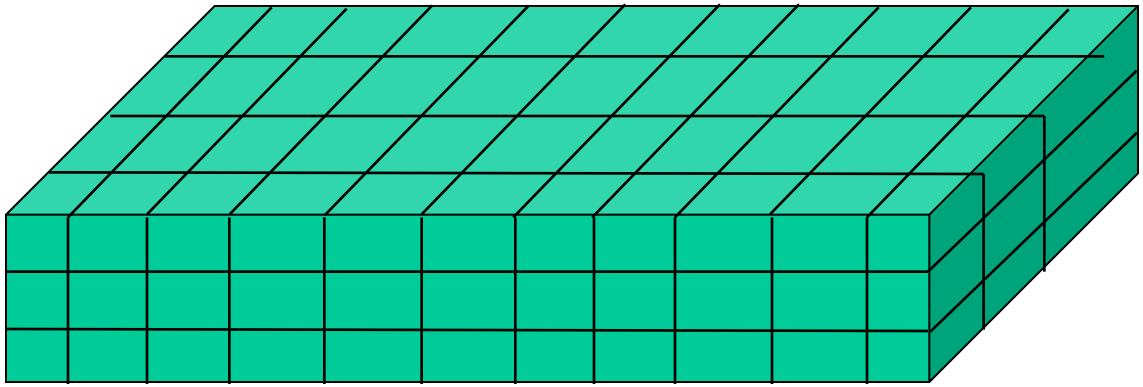
Creates an extremely large "circuit" problem

Generally interested only in the steady state -- *thermal capacitance* is not considered

Direct solution of these large problems can be impractical due to problem size

Iterative methods are very appealing due to sparsity

$$\rho C_p \frac{\partial T(x,y,z,t)}{\partial t} = \nabla \cdot [\kappa \nabla T(x,y,z,t)] + p(x,y,z,t)$$



- **Gauss Jacobi/Seidel can often be applied to these linear problems due to special matrix properties**
  - Diagonally dominant or symmetric positive definite (SPD)
- **We previously discussed the matrix condition of diagonal dominance**

$$A\vec{x} = \vec{b} \quad \Rightarrow \quad \vec{x}_{i+1} = \underbrace{(1 - A)}_{B}\vec{x}_1 + \vec{b}$$

- **Necessary and sufficient condition for convergence is:**

$$\lim_{i \to \infty} \left(B^i \vec{y} \right) = \vec{0} \quad \text{for all} \quad \vec{y}$$

- **OR:** $\lim_{i\to\infty}\left\|B^i\right\| = 0$

$$\left\|B\right\|_\infty = \max_{1\le i\le n}\sum_{j=1}^{n}\left|b_{ij}\right|$$

- **Sufficient but not necessary (diagonal dominance):**

$$A\vec{x} = \vec{b} \qquad \left\|1 - A\right\|_\infty < 1 \quad ?$$

$$\left\|a_{ii}\right\| > \sum_{\substack{j=1\\ j\neq i}}^{n}\left\|a_{ij}\right\| \quad \textbf{for} \quad i = 1,2\ldots n$$

- **We also want** $\left\|B\right\|$ **to be small so that iterations converge rapidly**

Strict diagonal dominance is not a necessary condition

Another *sufficient* condition for convergence is a *near diagonally* dominant M-matrix:

- A is an M-matrix if
  1. $a_{i,i} > 0, i = 1,\ldots,n$
  2. $a_{i,j} \le 0, i \ne j,\ \ i, j = 1,\ldots,n$
  3. ***A*** **is nonsingular**
  4. $A^{-1} > 0$ **, every element of the inverse is nonnegative**

- It can be shown that *A* is an *M*-matrix if it satisfies the following conditions:

1. $a_{i,j} \le 0, i \ne j, \;\; i, j = 1, \ldots, n$

2. $a_{i,i} > 0, i = 1, \ldots, n$

3. $\left|a_{jj}\right| \ge \sum_{\substack{i=1 \\ i \ne j}}^{i=n} \left|a_{ij}\right|, \;\; j = 1, \ldots, n$

4. $\left|a_{jj}\right| > \sum_{\substack{i=1 \\ i \ne j}}^{i=n} \left|a_{ij}\right|$ for at least one *j*

- The matrix from our thermal problem -- with boundary conditions -- is an M-matrix

- **Even with diagonal dominance or an M-matrix, G-S and G-J methods do not scale well with problem size**
  - **Excessive simulation runtime**

- **Multi-level approaches such as multigrid can be an ideal option**

  W. L. Briggs, "A multigrid tutorial", SIAM Press, 1987

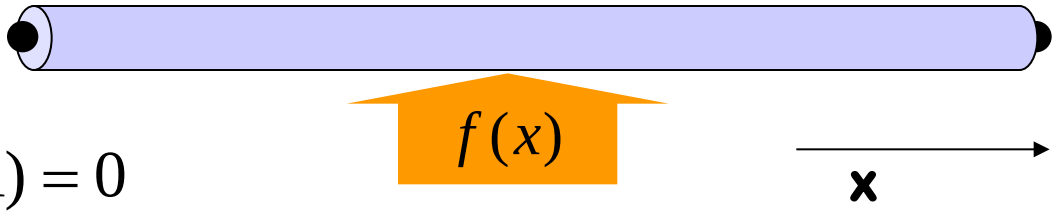
  - ***Method for solving large linear matrix problems iteratively***

A simple thermal problem example

- 1-d Poisson equation to describe the *steady-state* temperature distribution along a uniform rod

$$u \equiv T$$

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



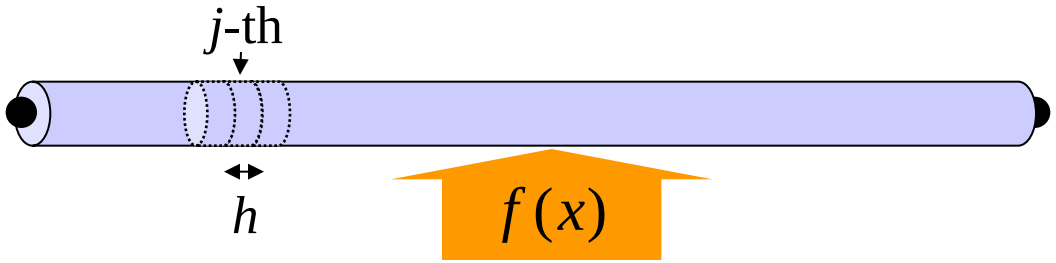
$$\rho C_p \frac{\partial T(x,y,z,t)}{\partial t} = \nabla \cdot [\kappa \nabla T(x,y,z,t)] + p(x,y,z,t)$$

Steady state

$$-G_x \cdot u''(x) = f(x), \quad 0 < x < 1$$

- **Approximate the 2nd order derivative using finite difference (3-point stencil)**

$$-G_x \cdot u''(x) = f(x), \quad 0 < x < 1$$



$$G_x\left(-u_{j-1} + 2u_j - u_{j+1}\right) = h^2 f_j, \quad 1 \le j \le N-1$$

$$u_0 = u_N = 0$$

$$h \equiv \text{step size}$$

- The linear system is:

$$A \cdot u = \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & -1 \\ & & & & -1 & 2 \end{bmatrix} \cdot u = h^2 f / G_x$$

- A is an M-matrix, therefore Gauss Jacobi/Seidel will converge for any initial guess
- Efficient direct solution does exists for this tridiagonal system
- But we use this model problem to study the converge rates of iterative methods

- Note: not strictly diagonally dominant, but an M-matrix with boundary conditions

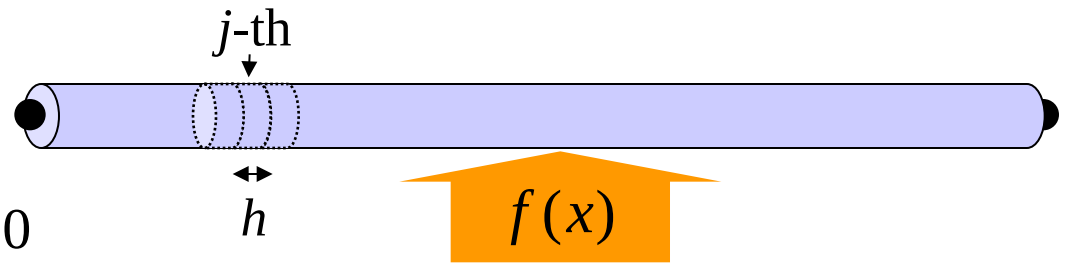
$$A \cdot u = \begin{bmatrix} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & -1 & \\ & & & & -1 & 2 \end{bmatrix} \cdot u = h^2 f / G_x$$

MichiganTech Create the Future

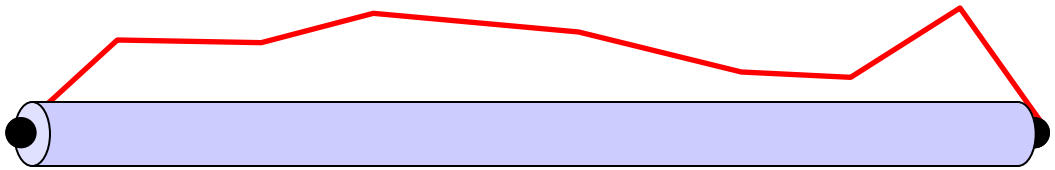
- For our 1d thermal problem, G-J/G-S will converge since underlying matrix is a M-matrix
- Direct application of GJ/GS can be slow due to the "geometric" distribution of the solution error
- Consider the change in the error distribution along the rod as we iterative with basic GJ/GS

$$u \equiv T$$

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



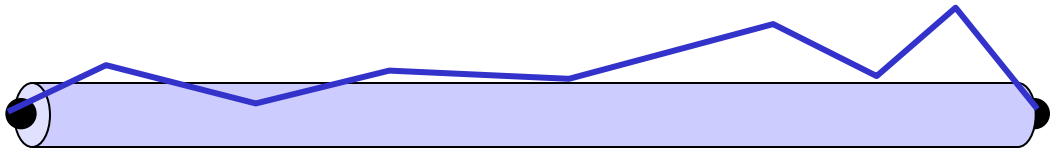
- Exact temperature distribution given a heat source of f(x) along the length of the rod



$$u \equiv T$$

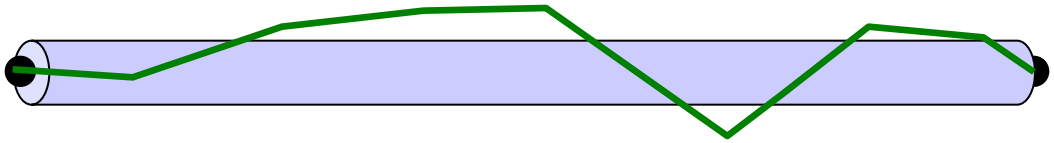
$$u(0) = u(1) = 0$$

- Assume that the initial guess is:

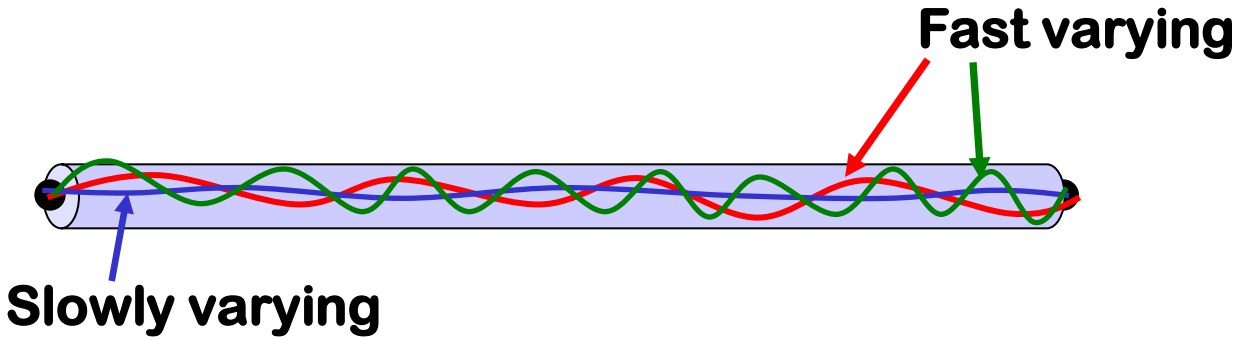


- Error can have different "frequency" components

$$e(x) = u(x) - \hat{u}(x)$$



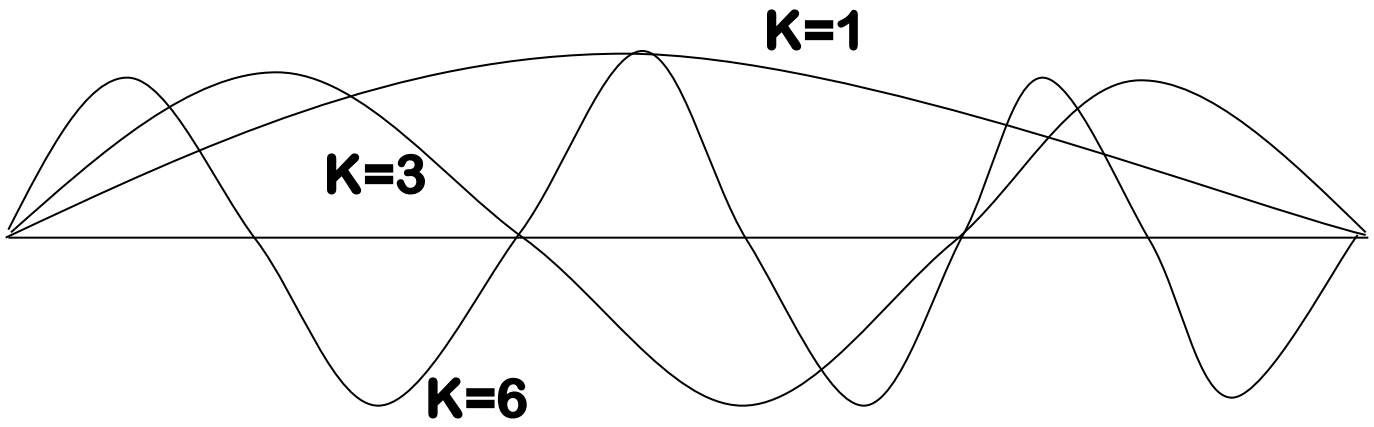
- "Low frequency" vs. "high frequency"



- GJ/GS acts differently on the error components of different frequencies
- "High frequency" errors can be removed rather quickly while "low frequency" ones cannot
- Fourier mode analysis can help us to understand the convergence rate of the model problem

- **Consider the following modes of the thermal response for our 1d model problem:**

$$w_k = \sin(\frac{jk\pi}{N}),\ 0 \le j \le N \ and\ 1 \le k \le N-1$$



- K is referred to as the “wavenumber”
- The Fourier modes with a high wavenumber represent high frequency components of waveform
- Gauss Jacobi/Seidel remove the error components of the initial guess at different rates

- As an example, consider the case for which the RHS is set to zero (f(x)=0) and we solve:

$$-u_{j-1} + 2u_j - u_{j+1} = 0,\ 1 \le j \le N-1$$
$$u_0 = u_N = 0$$

  - N is the number of discretization points along the rod

- Since the exact solution is zero for this case, then the error is equivalent to the initial guess

- We can decompose our initial guess u_{init} as a linear combination of different Fourier modes

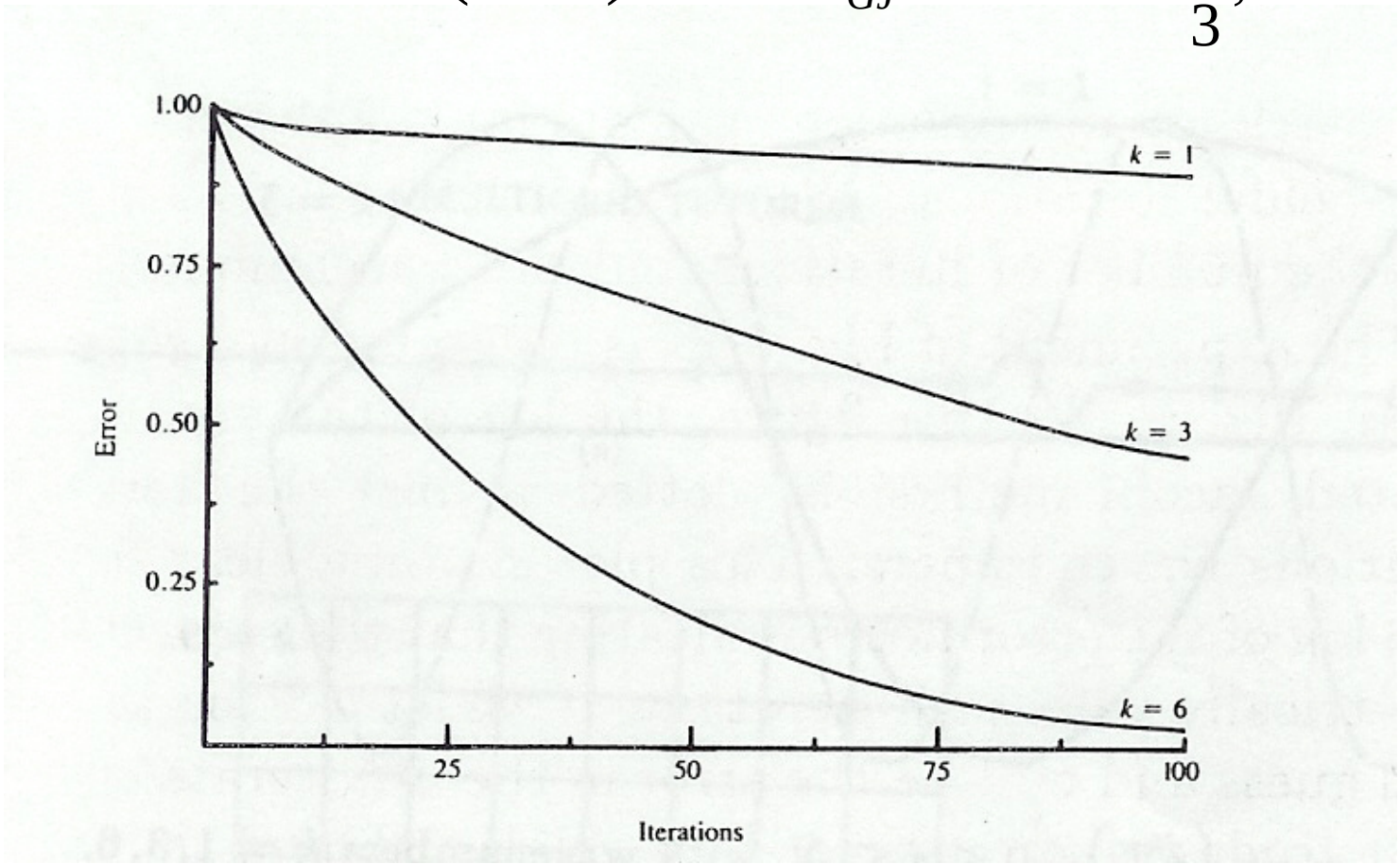
- We would expect the error to decay much more quickly for the high frequency initial guesses
- We'll consider the error behavior for a weighted GJ iteration

$$u^{n+1} = (1-\omega)u^n + \omega u_{GJ}^{n+1}$$ ← Solution obtained in the standard Gauss Jacobi

- The use of relaxation parameter ω can improve the convergence
- We will see that ω affects the damping rates for errors of different frequencies

- **Consider 3 different cases of initial guesses, namely w_1, w_3, w_6, respectively**

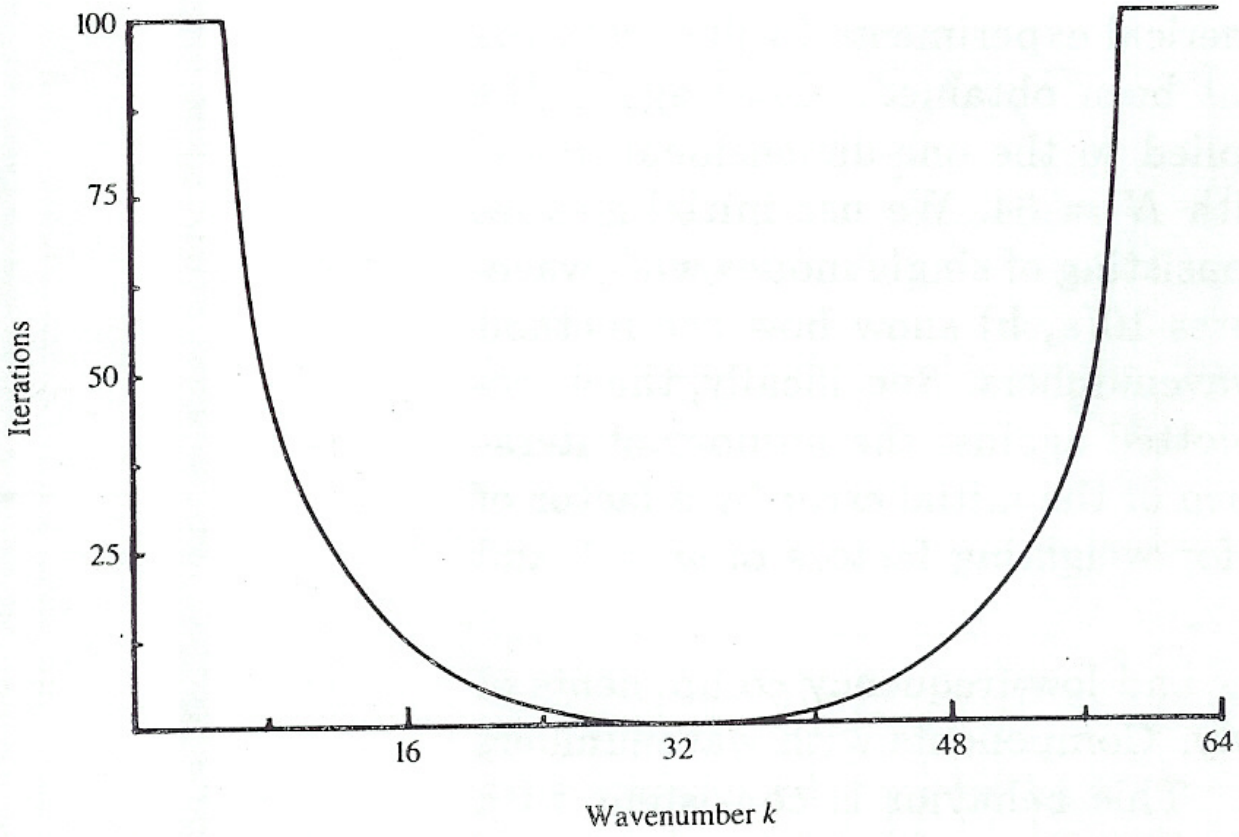
$$u^{n+1} = (1-\omega)u^n + \omega u_{GJ}^n \qquad \omega = \frac{2}{3},\ N = 64$$



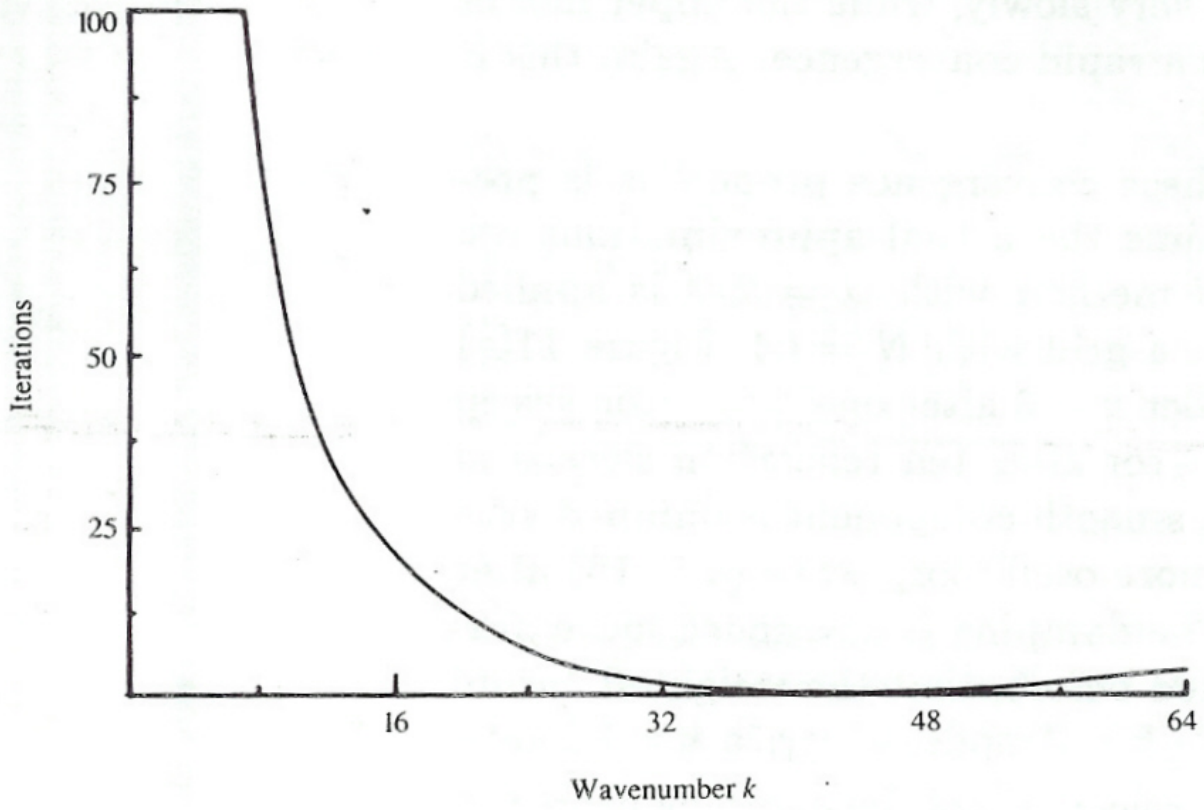
- Let the initial guess consist of all modes

$$w_k, 1 \le k \le 63$$

- Run the weighted Gauss Jacobi with $\omega = 1$ until the norm of each mode (error) is reduced by a factor of 100



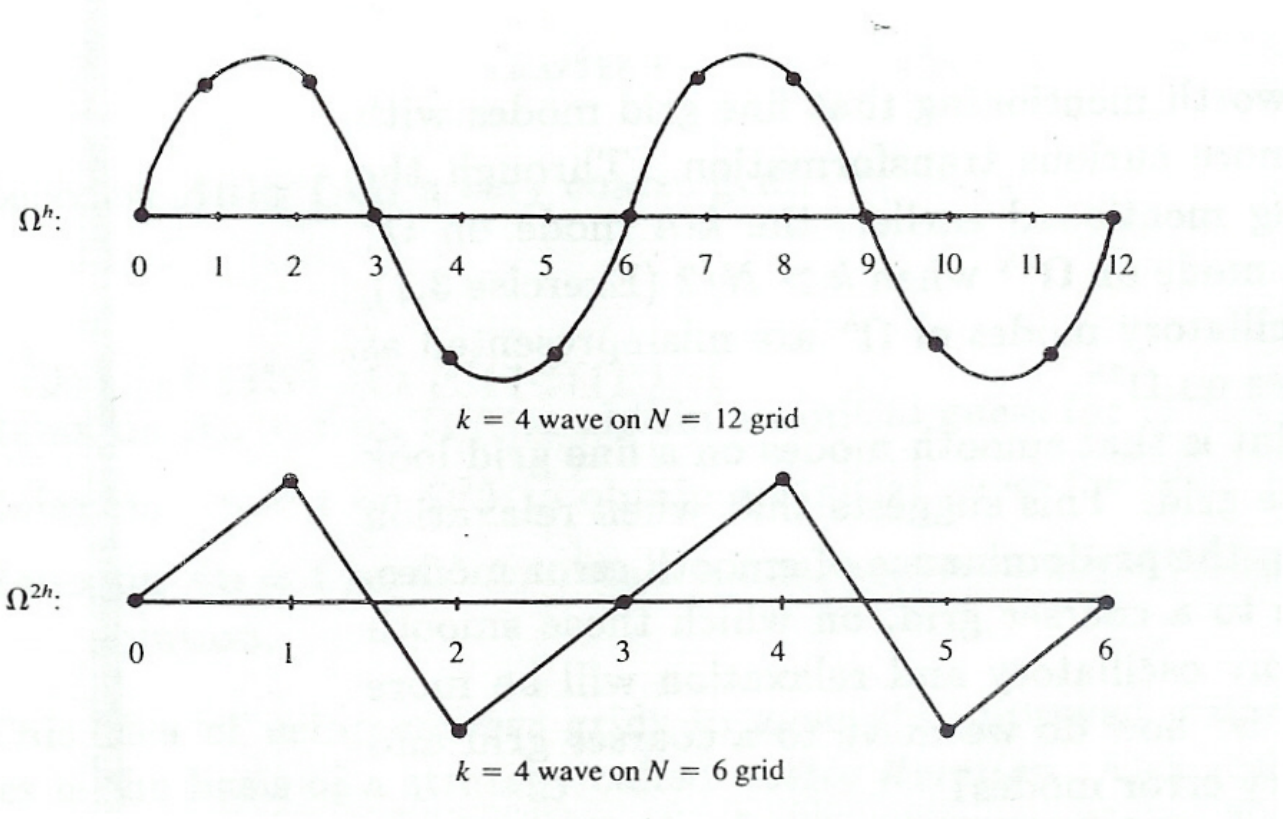
- The result gets better if we change to $\omega = 2/3$

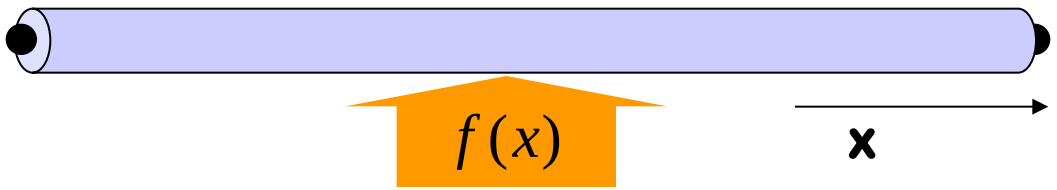


- However convergence is still slow for "smooth" (low frequency) modes

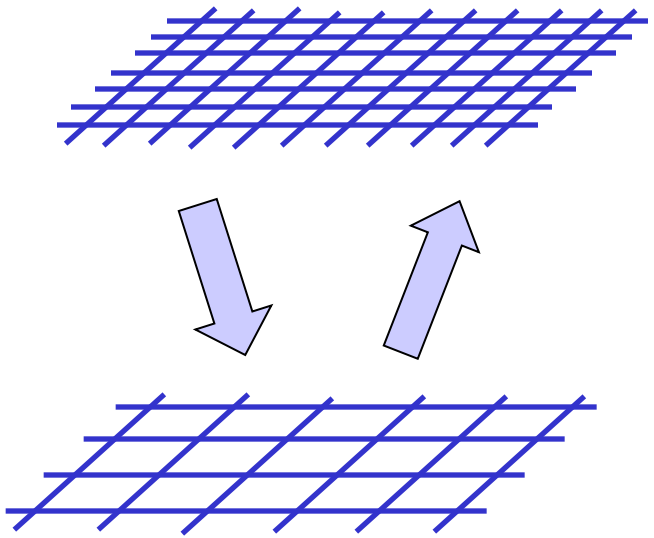
- Rapid decrease for early iterations is due to the fast elimination for high frequency modes of error
- Convergence starts to stall once the high frequency modes are removed
- The “smoothing” property of the basic iteration schemes is a severe limitation
- An effective remedy is multigrid methods

- Since the basic iteration methods struggle with smooth errors, alter the discretization to increase high freq errors





- This suggest that we should move to a coarser grid once the relaxation on the fine grid stalls
- Relaxations on a coarser grid can also be much cheaper since the problem size is smaller
- This idea is systematically explored in multigrid by solving the problem at a hierarchy of grids
- Two grid cycle for a 2D problem:

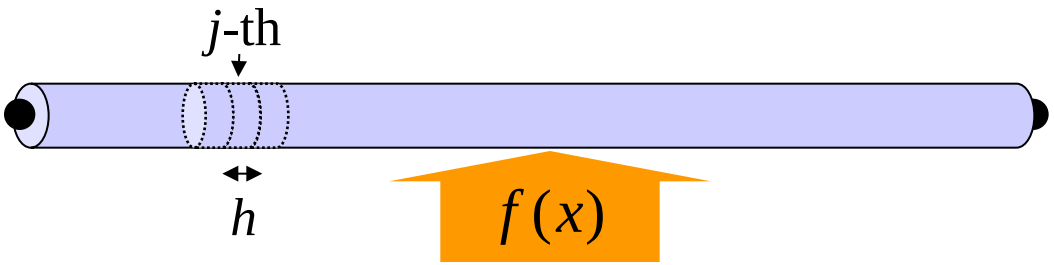


Original Problem

Coarser grid problem

Two grid cycle for our 1D rod problem

- Incomplete relaxation of Au=b on fine grid, Ω^h, to obtain an approximation of u → v^h



$$\begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & -1 \\ & & & & -1 & 2 \end{bmatrix} \cdot u = b \qquad b = h^2 f / G_x$$

- Relaxation of Au=b on fine grid, Ω^h, to obtain intermediate variable v^h
- Compute the residue $r = b - Av^h$
  - Relaxation of the residual equation **Ae = r** on Ω^{2h} (coarser grid) to get an approximation of the error $\mathbf{e^{2h}}$
  - Map $\mathbf{e^{2h}}$ back to $\mathbf{e^h}$ on Ω^h
  - Correct the solution on Ω^h by:

    $\mathbf{v^h \leftarrow v^h + e^h}$

- Restart the first step until convergence

- Could extend this to more than the two levels described here…

- Typically, the grid spacing of a coarse grid is the twice of that of the immediate fine grid

- Transfer of e^{2h} from Ω^{2h} to Ω^{h} is done by interpolation: I_{2h}^{h}

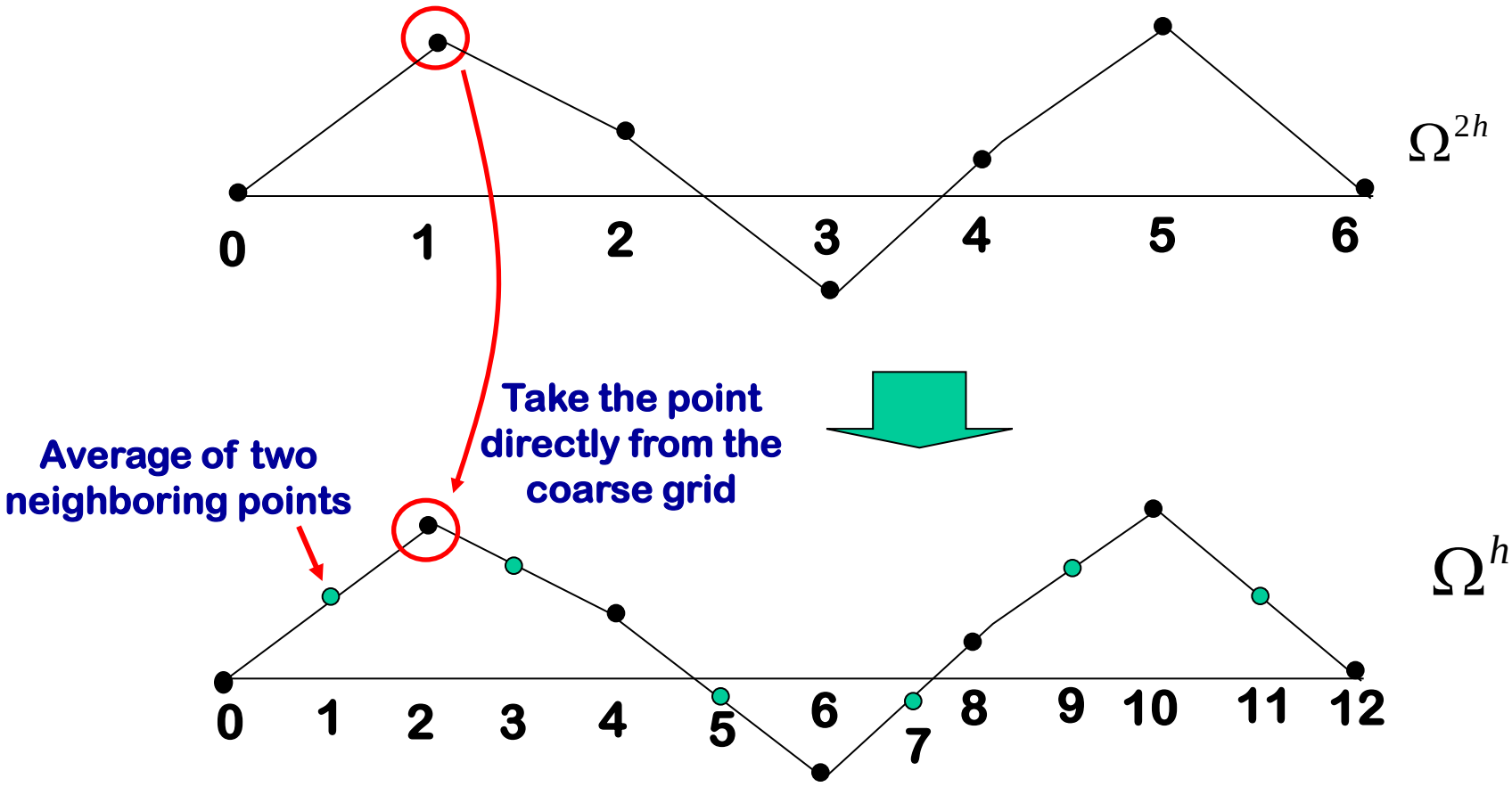
- Interpolation operator $v^h = I_{2h}^{h} v^{2h}$ maps a coarse grid operator to a fine grid operator:

$$v_{2j}^{h} = v_{j}^{2h}$$

$$v_{2j+1}^{h} = \frac{1}{2}\left(v_{j}^{2h} + v_{j+1}^{2h}\right) \qquad 0 \le j \le \frac{N}{2} - 1$$

- Linear interpolation among grid points for any number of levels of hierarchy

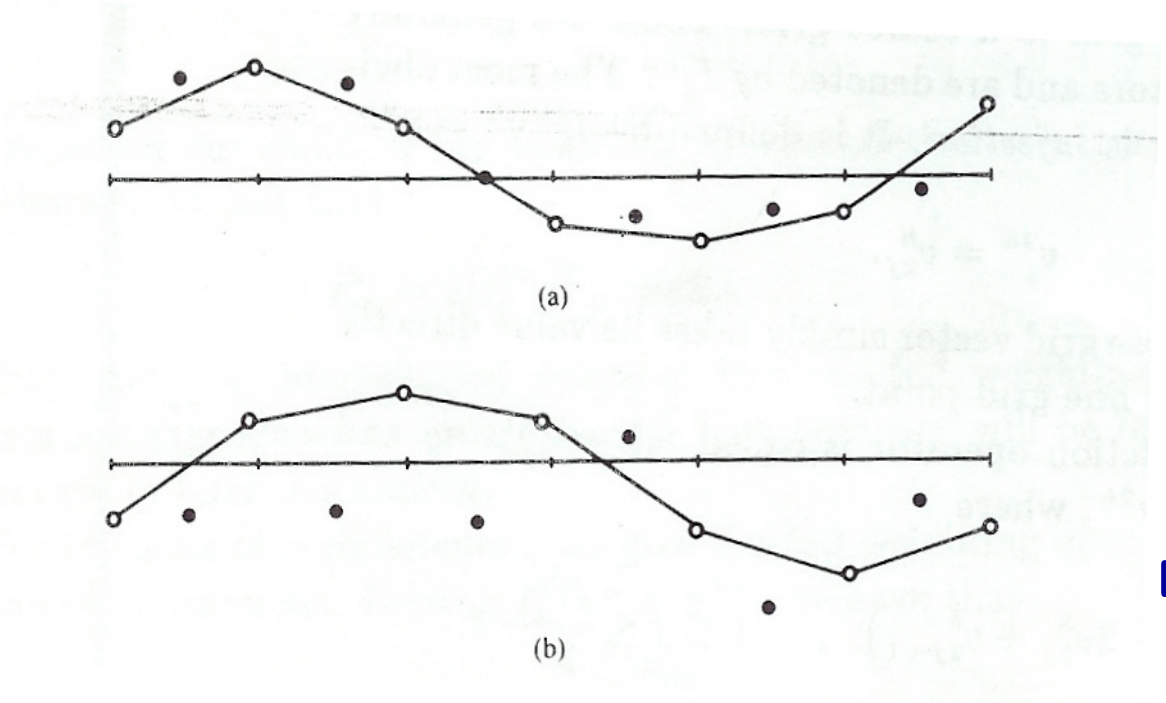
Linear interpolation



- Can express the interpolation operator in a matrix form
- For our specific model problem with N=6 *and zero endpoint* boundary conditions:

$$v^h = I_{2h}^h v^{2h} = \frac{1}{2}\begin{bmatrix} 1 & & & & & \\ 2 & & & & & \\ 1 & 1 & & & & \\ & 2 & & & & \\ & 1 & 1 & & & \\ & & 2 & & & \\ & & 1 & 1 & & \\ & & & 2 & & \\ & & & 1 & 1 & \\ & & & & 2 & \\ & & & & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{bmatrix}_{2h}$$

- Using interpolation implies that the error (on the fine grid) must be smooth enough



Interpolation that produces a good approximation of error on the fine grid

Interpolation that produces a rather poor approximation

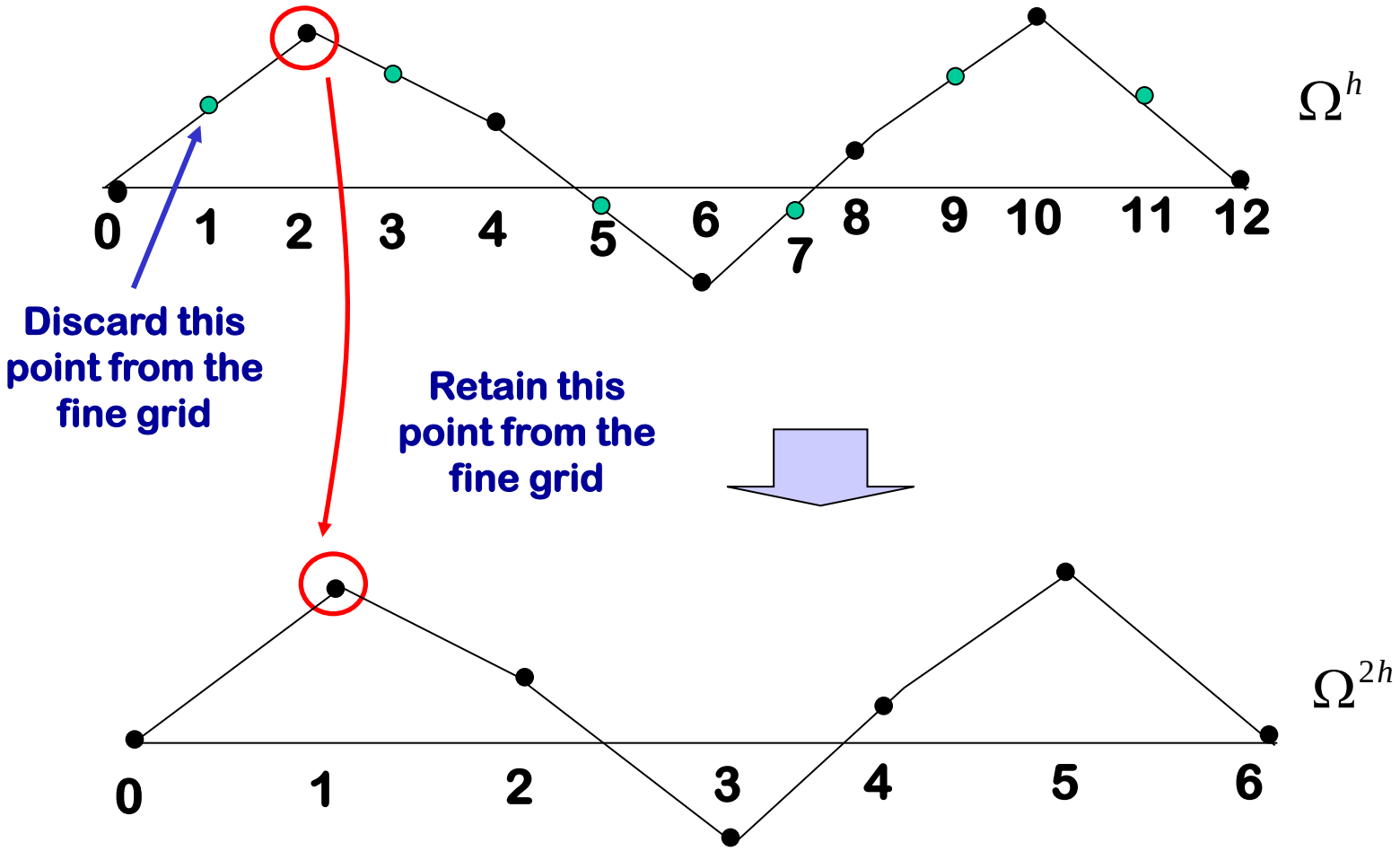
- Coarse grid correction can only be used after high frequency errors are sufficiently damped on the fine grid

- We also need to define the transfer from the fine grid to the coarse one

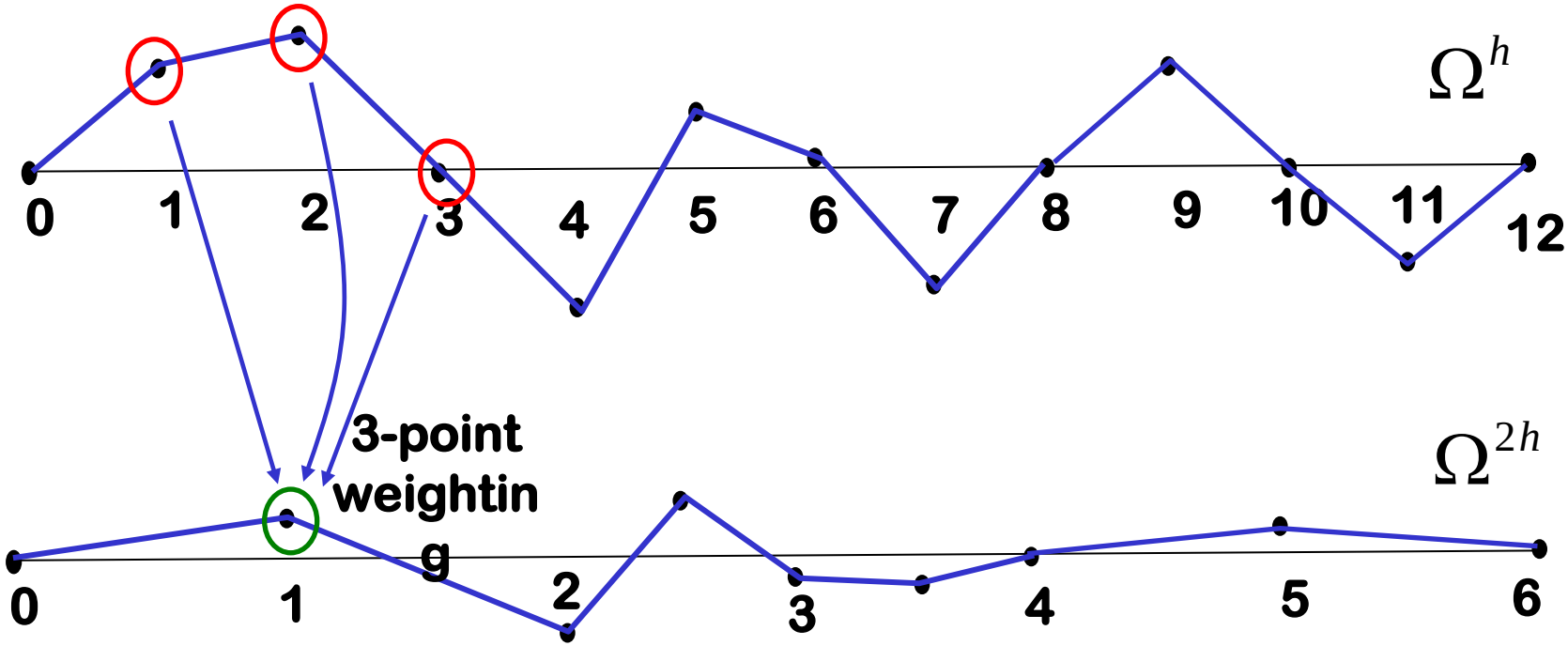
- Restriction operator I_h^{2h} defines the mapping from Ω^h to Ω^{2h}: $v^{2h} = I_h^{2h} v^h$

- The simplest restriction operator is *injection*
  - Points which are not on the coarse grid are simply removed

Injection


Ω^h
0
1
2
3
4
5
6
7
8
9
10
11
12
Discard this point from the fine grid
Retain this point from the fine grid
Ω^{2h}
0
1
2
3
4
5
6

- **A more popular restriction operator is *full weighting***
- **Full weighting in 1D**
  - For specific model problem with N=12:



- Can express the full weighting operator in a matrix form:

$$v_j^{2h} = \frac{1}{4}\left(v_{2j-1}^h + 2v_{2j}^h + v_{2j+1}^h\right), \quad 1 \le j \le \frac{N}{2} - 1$$

$$v^{2h} = I_h^{2h} v^h$$

$$I_h^{2h} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & & & & & & & \\ & & 1 & 2 & 1 & & & & & & & \\ & & & & 1 & 2 & 1 & & & & & \\ & & & & & & 1 & 2 & 1 & & & \\ & & & & & & & & 1 & 2 & 1 & \end{bmatrix} \qquad v^h = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \\ v_{10} \\ v_{11} \end{bmatrix}$$

N=12

- Full weighting operator is proportional to the transpose of linear interpolator (a factor of 2 difference in this case)

- We want to iterate on the residue equation on the coarse grid to adjust the fine grid solution
- Recall that if $\hat{x}$ is an approximate solution for :

$$Ax = b$$

- Then residue is defined as $Ae = r, \quad r = b - A\hat{x}$ such that :

$$A(\hat{x} + e) = b$$

- We define an approximate coarse grid problem for the residue equation

error we want to evaluate

$$A^{2h} e^{2h} = r^{2h}$$

coarse grid operator

residue vector approximated from fine grid solution

- r^{2h} is obtained by restricting (mapping) the residue of the fine grid to the coarse grid:

$$r^{2h} = I_h^{2h} r^h$$

- Note that residue from fine grid is the only "input" to coarse grid problem

- A^{2h} must be a good approximation of the original matrix on the coarse grid
  - ► can be obtained by discretizing the PDE using a step size 2h

- We can also approximate the coarse grid problem from the fine grid for better robustness:
  - ► Galerkin coarse grid approximation

- Recall on the fine grid the residue equation is:

$$A_h e_h = r_h, \quad r_h = b_h - A_h \hat{x}_h$$

- Based on the above, we can say that the following inner product relationship must be satisfied:

$$(A_h e_h, v_h) = (r_h, v_h)$$

↑

For any vector on Ω^h

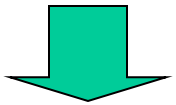
- We can also relate the coarse grid errors and approximations to those of the fine grain model as follows:

$$e_h = I_{2h}^h e_{2h} \qquad v_h = I_{2h}^h v_{2h}$$

- Note that these interpolation operators can be different in some cases, but are the same for us in these examples

- We get the approximate residue equation:

$$(A_h I_{2h}^h e_{2h}, I_{2h}^h v_{2h}) = (r_h, I_{2h}^h v_{2h})$$



$$(I_h^{2h} A_h I_{2h}^h e_{2h}, v_{2h}) = (I_h^{2h} r_h, v_{2h})$$

- For our example, we can use: $I_h^{2h} \equiv \left(I_{2h}^h\right)^T$
- This suggests the Galerkin method to define the coarse grid problem:

$$A_{2h} = I_h^{2h} A_h I_{2h}^h$$

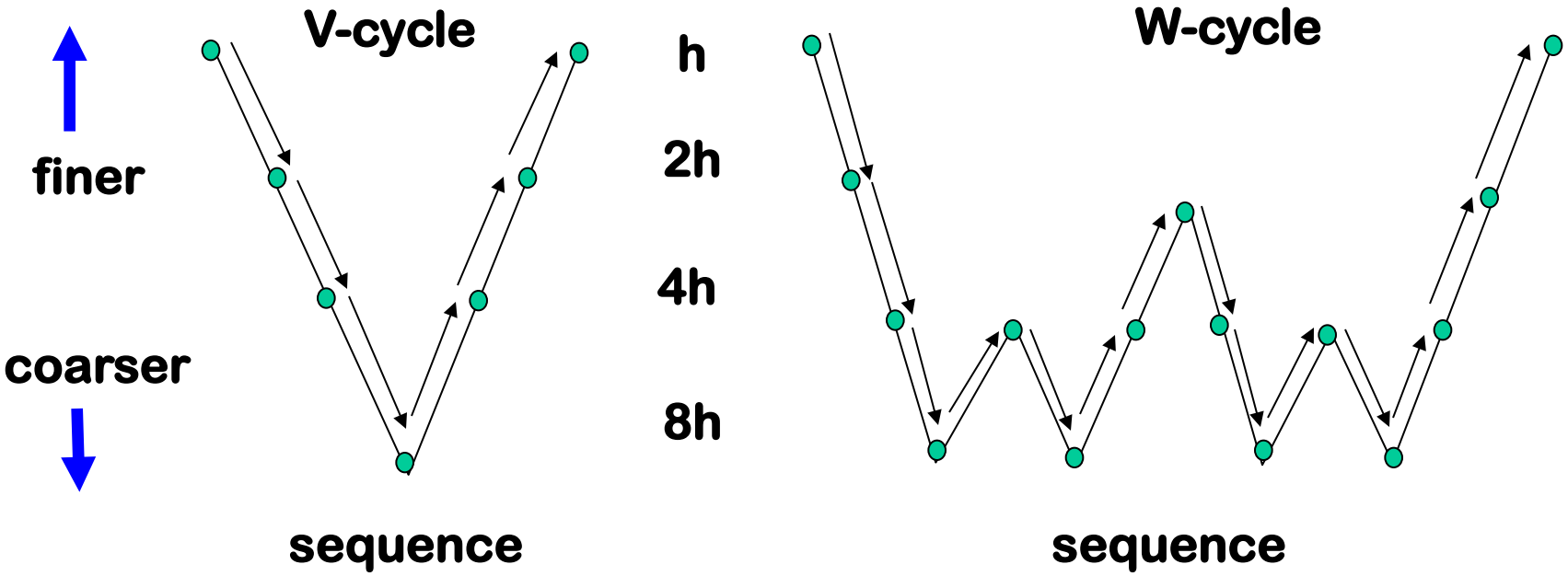
$$r_{2h} = I_h^{2h} r_h$$

Maps fine grid residue to coarse grid residue

$$\boxed{A_{2h} e_{2h} = r_{2h}}$$

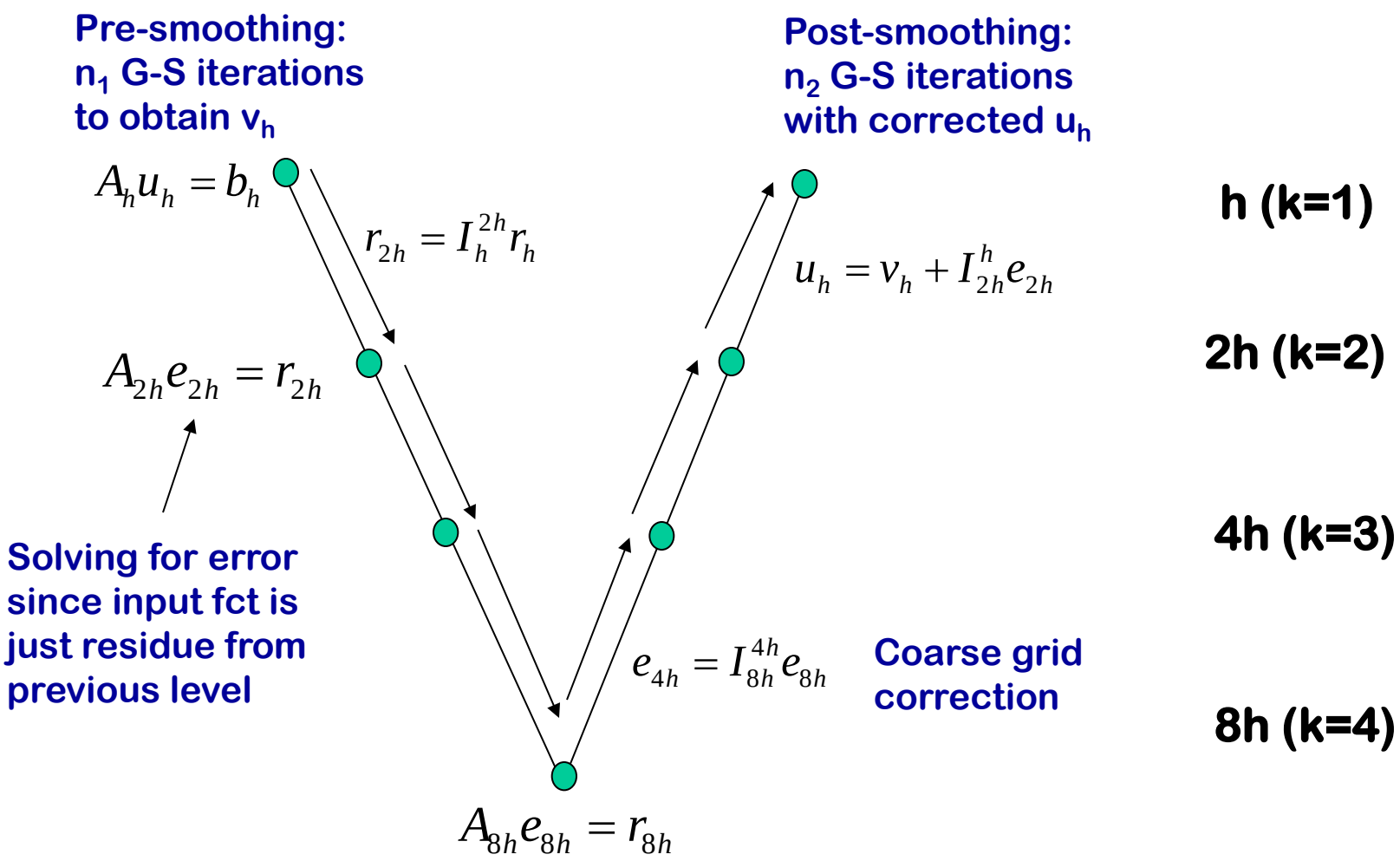
- **Multigrid combines the benefits of fine grid and coarse grid relaxations rather nicely**
  - Fine grid iteration takes care of the “high frequency” errors
  - Coarse grid one removes the “low frequency” errors
  - Fast convergence is achieved by a proper interplay between different grids

- The basic two grid cycle can be applied recursively to lead to the multi-level iterations



- The multigrid sequences to the coarsest grid for which direct solve is efficiently applied to a very small problem

- Recursive procedure until we reach a coarse grid size for which we can solve the problem via direct methods



- **Multigrid cycle:**

$$u_k = MG(k, u_k^0, A_k, b_k, m, n_1, n_2)$$

k: grid level, k=1 indicates the finest level

u_k^0 : initial guess for k-th level based on boundary conditions

A_k : linear problem (matrix) at the kth level

b_k : RHS at the k-th level

m, n_1, n_2 : iteration control parameters

- **Multigrid cycle flow:** $u_k = MG(k, u_k^0, A_k, b_k, m, n_1, n_2)$

1. **Pre-smoothing:**

- Compute v_k by performing n_1 smoothing steps with initial u_k^0 guess

$$v_k = smooth^{n1}(u_k^0, A_k, b_k)$$

2. **Coarse grid correction**

- Compute the residue: $r_k = b_k - A_k v_k$

- Restrict the residue: $r_{k+1} = I_k^{k+1} r_k$

- Compute the solution of residue equation on the (k+1)-*th* grid:

$$A_{k+1} e_{k+1} = r_{k+1}$$

if at coarsest level, direct solve, otherwise: apply multigrid cycle m times with zero initial guess

$$e_{k+1} = MG^m(k+1, 0, A_{k+1}, r_{k+1}, m, n_1, n_2)$$

- Interpolate the correction: $e_k = I_{k+1}^k e_{k+1}$

- Correct the approximation on the kth level: $u_k = v_k + I_{k+1}^k e_{k+1}$

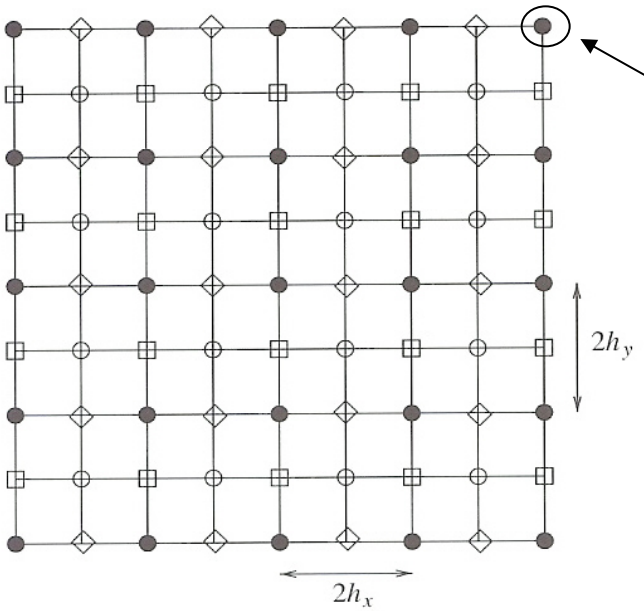
3. Post-smoothing

Perform smoothing steps n_2 times with initial guess u_k

$$u_k = smooth^{n2}(u_k, A_k, b_k)$$

- V-cycle corresponds to m=1 and W-cycle m=2

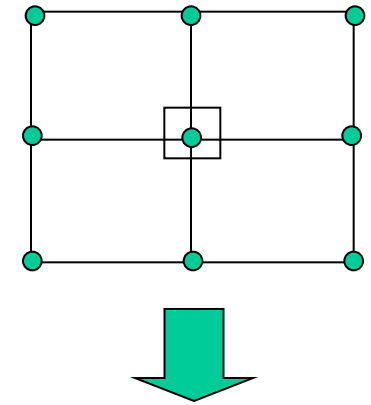
Interpolation in 2D → coarse grid to fine grid



Points on the coarse grid

$$I_{2h}^{h}\hat{v}_{2h}(x,y)=\begin{cases}\hat{v}_{2h}(x,y) & \text{for } \bullet\\ \frac{1}{2}[\hat{v}_{2h}(x,y+h)+\hat{v}_{2h}(x,y-h)] & \text{for } \square\\ \frac{1}{2}[\hat{v}_{2h}(x+h,y)+\hat{v}_{2h}(x-h,y)] & \text{for } \diamond\\ \frac{1}{4}[\hat{v}_{2h}(x+h,y+h)+\hat{v}_{2h}(x+h,y-h) & \\ \quad +\hat{v}_{2h}(x-h,y+h)+\hat{v}_{2h}(x-h,y-h)] & \text{for } \circ\,.\end{cases}$$

Full weighting in 2D → Fine grid to coarse grid



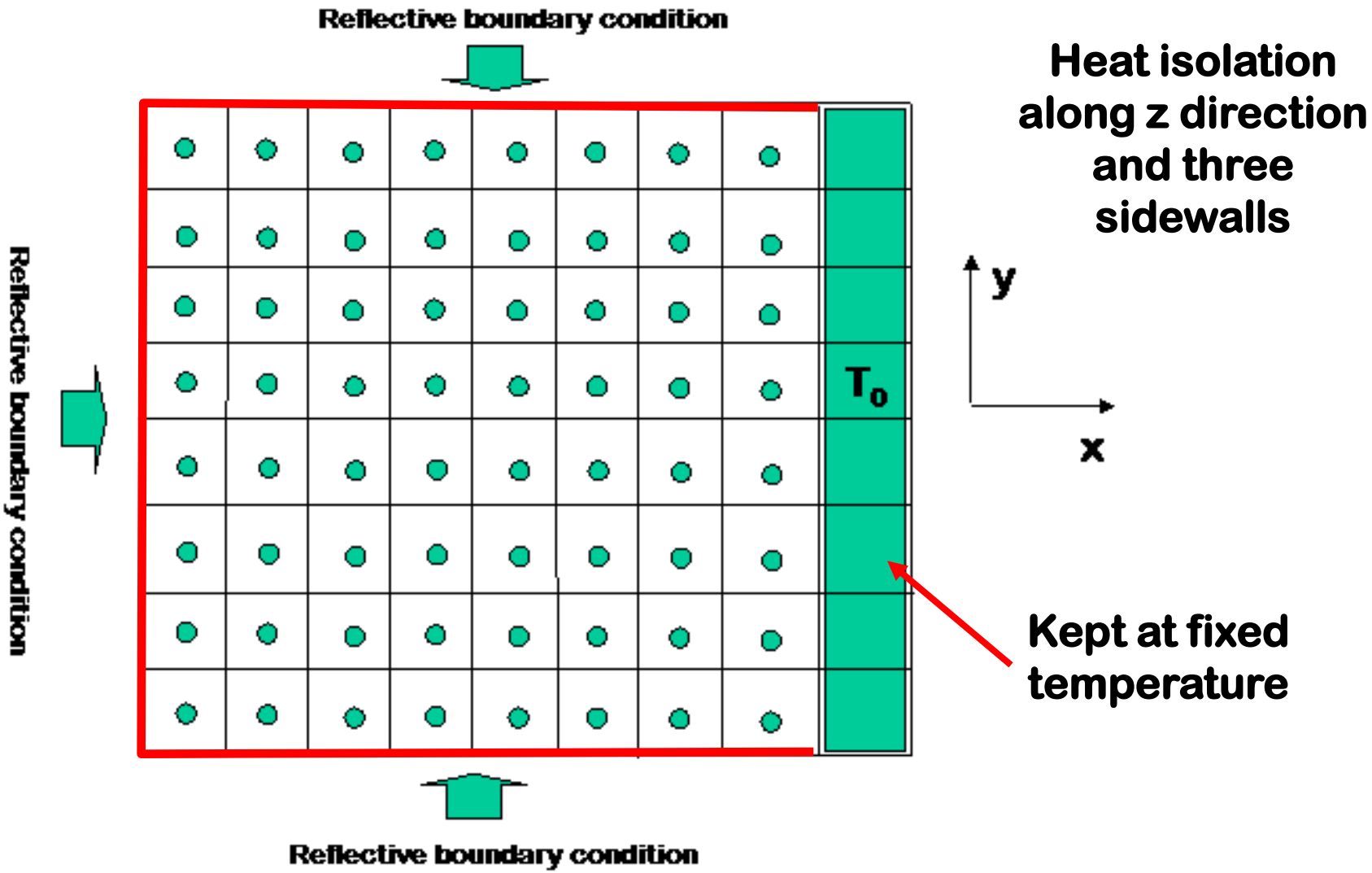
$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_{h}^{2h}$$

The center point in the coarse grid is a nine-point weighted average

Points on the boundary are slightly different

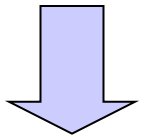
- fewer neighboring points in the average

Example: thermal problem in 2D



- **We are dealing with a 2D discretization of the thermal PDE**

$$\rho C_p \frac{\partial T(x,y,z,t)}{\partial t} = \nabla \cdot [\kappa \nabla T(x,y,z,t)] + p(x,y,z,t) \quad \textbf{3D}$$



Discretization in 2D

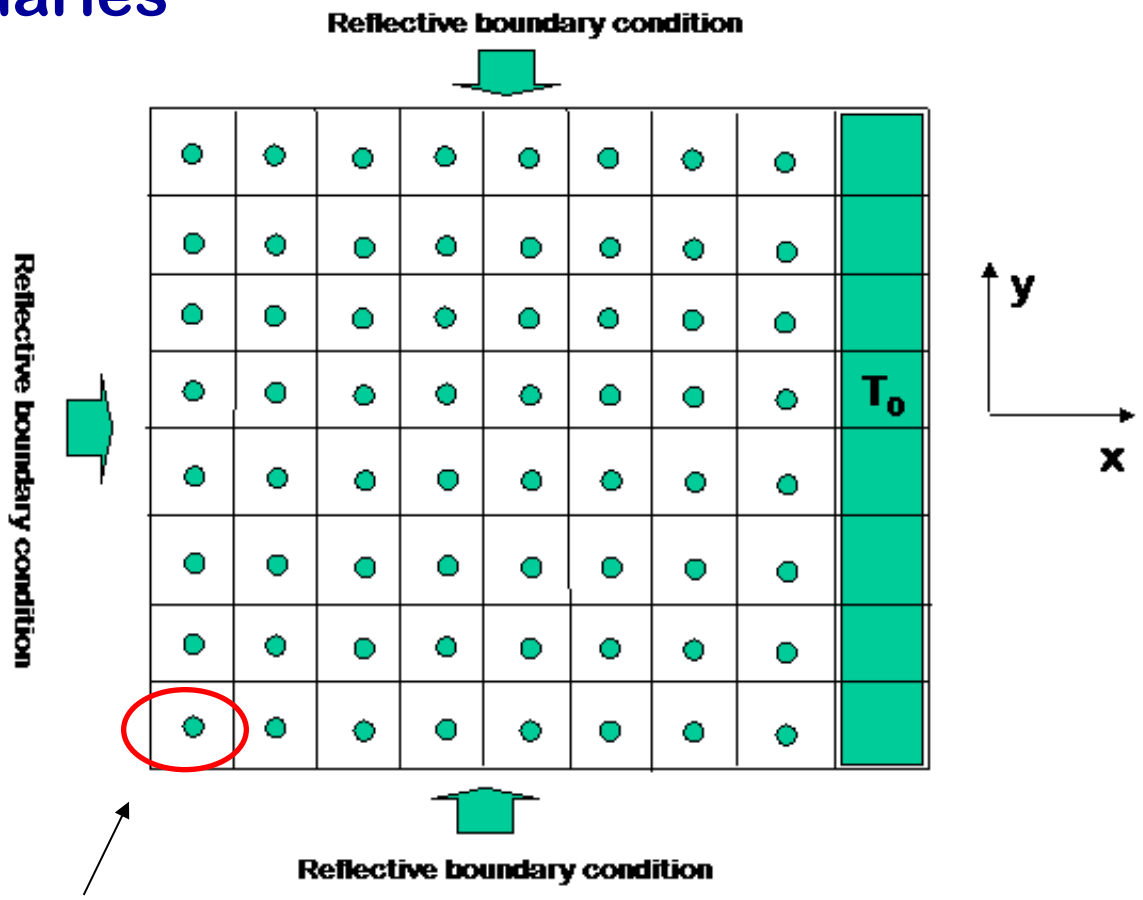
$$2(g_x + g_y)T_{i,j} - g_x T_{i-1,j} - g_x T_{i+1,j} - g_y T_{i,j-1} - g_y T_{i,j+1} = p$$

$$g_x = \frac{\kappa}{\Delta x^2}$$ → Thermal conductance in x

$$g_y = \frac{\kappa}{\Delta y^2}$$ → Thermal conductance in y

p → power ***density*** in a box

- There are fewer heat conduction paths at a reflective boundaries



$$(g_x + g_y)T_{i,j} - g_x T_{i+1,j} - g_y T_{i,j+1} = p$$

- The multigrid technique we discussed is under the category of geometric multigrid
- Algebraic multigrid (AMG) follows a different philosophy where multigrid iteration is carried out completely based on a given matrix
- The principle of multigrid has also been adopted to analyze both IC thermal problems and "discrete" problems such as power grid

J. Kozhaya, S. Nassif and F. Najm, "A multigrid-like technique for power grid analysis," IEEE Trans. on CAD, vol. 21, no. 10, pp. 1148-1160, Oct. 2002.

Li, P.; Pileggi, L.T.; Asheghi, M.; Chandra, R.,
"IC thermal simulation and modeling via efficient multigrid-based approaches", IEEE Transactions on CAD Volume 25, Issue 9, Sept. 2006 Page(s):1763 - 1776