VI. Solution of Linear Systems

2008. 10

담당교수: 주 한 규
joohan@snu.ac.kr, x9241, Rm 32-205
원자핵공학과

VI. Solution of Linear Systems

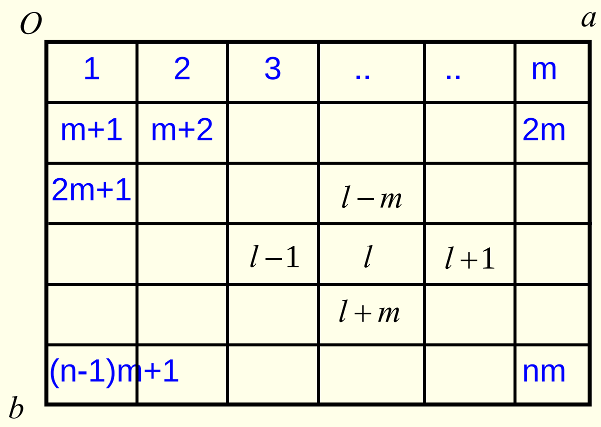
1. Model Problem

□ 2-D Diffusion Equation

$$\nabla\cdot\left(-D\nabla\phi\right)+\sigma\phi=s$$

$$-D\left(\frac{\partial^2\phi(x,y)}{\partial x^2}+\frac{\partial^2\phi(x,y)}{\partial y^2}\right)+\sigma\phi(x,y)=s(x,y)\,,\quad x\in[0,a],\,y\in[0,b]$$

□ Domain and Node Numbering



Solution Vector

$$\begin{bmatrix}\phi_1\\ \phi_2\\ \vdots\\ \phi_m\\ \phi_{m+1}\\ \vdots\\ \phi_{2m}\\ \vdots\\ \phi_N\end{bmatrix}$$

$$\left.\frac{d^2\phi}{dx^2}\right|_l=\left.\frac{d}{dx}\frac{d\phi}{dx}\right|_l\cong\frac{1}{h_x}\left(\left.\frac{d\phi}{dx}\right|_l^R-\left.\frac{d\phi}{dx}\right|_l^L\right)$$

$$\cong\frac{1}{h_x}\left(\frac{\phi_{l+1}-\phi_l}{h_x}-\frac{\phi_l-\phi_{l-1}}{h_x}\right)$$

$$=\frac{1}{h_x^2}\left(\phi_{l+1}-2\phi_l+\phi_{l-1}\right)$$

□ Discretized Equation for Node *l* after dividing by D and defining $B^2=\frac{\sigma}{D}$

$$\frac{-\phi_{l+1}+2\phi_l-\phi_{l-1}}{h_x^2}+\frac{-\phi_{l+m}+2\phi_l-\phi_{l-m}}{h_y^2}+B_i^2\phi_l=\tilde{s}_l$$

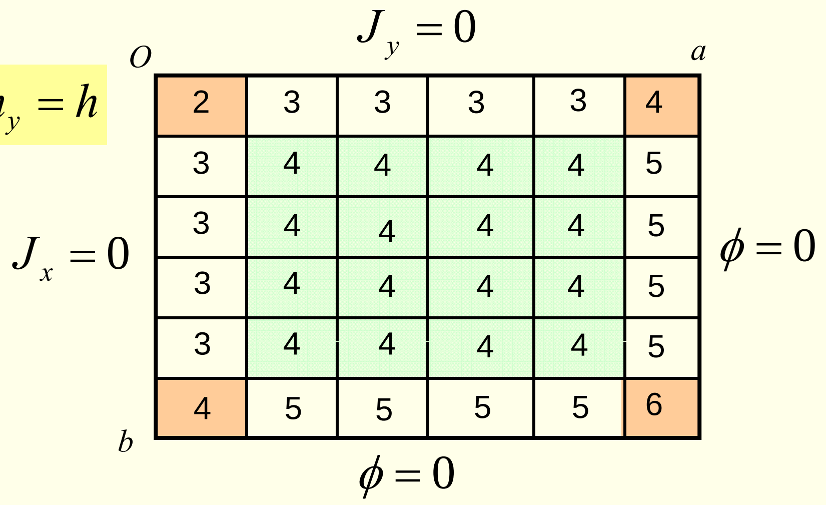
$$\Rightarrow-\frac{1}{h_y^2}\phi_{l-m}-\frac{1}{h_x^2}\phi_{l-1}+\left(B_l^2+\frac{2}{h_x^2}+\frac{2}{h_y^2}\right)\phi_l-\frac{1}{h_y^2}\phi_{l+1}-\frac{1}{h_x^2}\phi_{l+m}=\tilde{s}_l$$

1. Model Problem

□Boundary Conditions

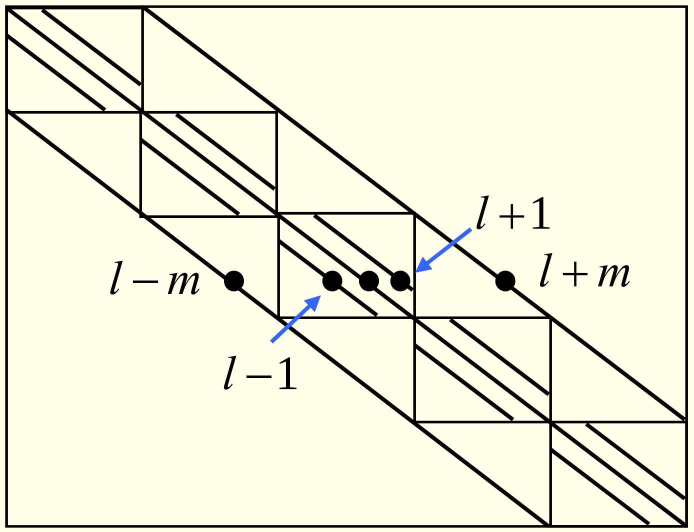
$$J_x = \left.\frac{\partial \phi}{\partial x}\right|_{x=0} = 0 \ , \ J_y = \left.\frac{\partial \phi}{\partial y}\right|_{y=0} = 0 \ , \ \phi(a,y) = 0 \ , \ \phi(x,b) = 0$$

- **Diagonal entries in $\frac{1}{h^2}$ for $h_x = h_y = h$**



2	3	3	3	3	4
3	4	4	4	4	5
3	4	4	4	4	5
3	4	4	4	4	5
3	4	4	4	4	5
4	5	5	5	5	6

□Structure of Matrix



Block Tridiagonal Matrix

* Size of the Block = m x m
(representing each row of meshes)
* Number of the Diagonal Blocks = n

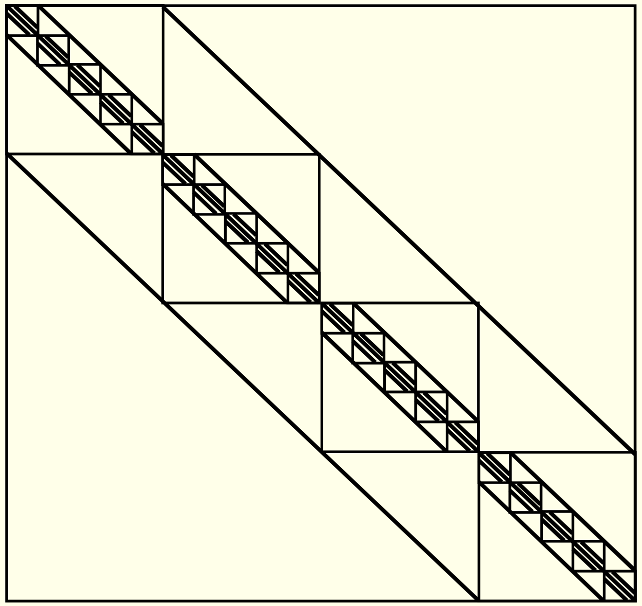
* Off-diagonal Blocks represents
coupling between rows of meshes
* Penta-diagonal Matrix

1. Model Problem

□3-D Extension

$$-D\left(\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}+\frac{\partial^2\phi}{\partial z^2}\right)+\sigma\phi(x,y,z)=s(x,y,z)\,,\quad x\in[0,a],\, y\in[0,b],\, z\in[0,c]$$

- **Stack of planes with the same radial numbering scheme**
- **Diagonal entries added by $2/h_z^2$ in the interior planes**



($3/h_z^2$ for the first and last plane meshes

if zero flux BC applied at both axial boundaries)

Block Tridiagonal Matrix

* Size of the Block = N x N
(representing each plane)

* Number of the Diagonal Blocks = k

* Off-diagonal Blocks represents
coupling between planes

* Septa-diagonal Matrix

2.1 Gauss Elimination

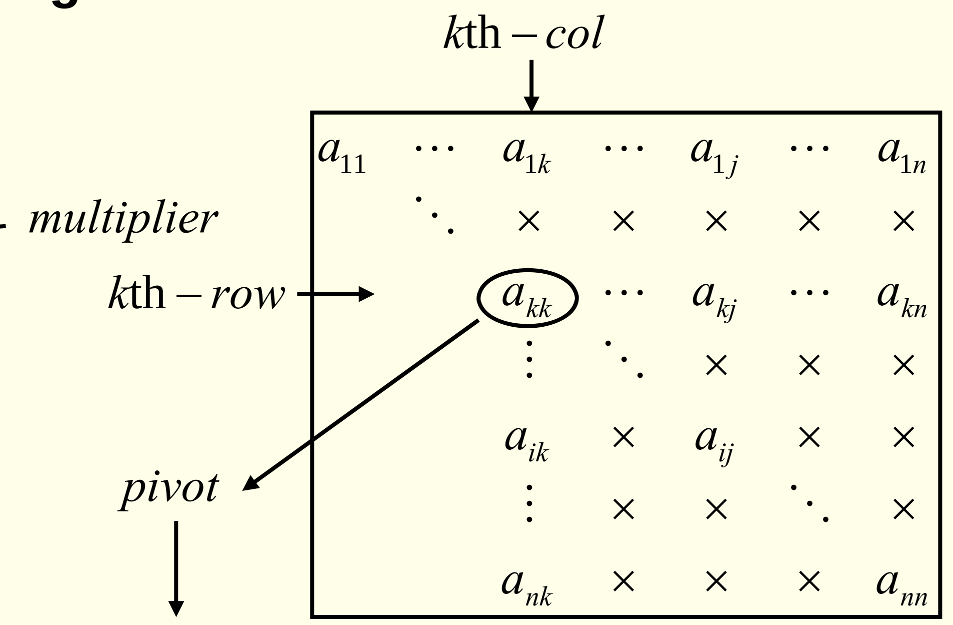
☐ Solve a linear system $Ax = b$

$$A = (a_{ij})\ ,\ \text{Rank(A)} = \text{n};\ x = [x_1 \quad x_2 \quad \cdots \quad x_n]^T;\ b = [b_1 \quad b_2 \quad \cdots \quad b_n]^T$$

☐ Forward Elimination

- **Eliminate lower diagonal entries of the k-th column at the k-th elimination step using the k-th row of the current reduced matrix**

```
for k = 1:n-1
  for i = k+1:n
    $m_{ik} = a_{ik} / a_{kk}$
    for j = k+1:n
      $a_{ij} := a_{ij} - m_{ik} a_{kj}$
    end
    $b_i := b_i - m_{ik} b_k$
  end
end
```



Assume non-zero pivot for now...

2.1 Gauss Elimination

☐ Backward Substitution

- **After elimination, determine x_i utilizing the upper diagonal structure**

```
$x_n := b_n / a_{nn}$
for i = n-1:-1:1
    sumod = 0
    for j = k+1:n
        sumod := sumod $- a_{ij} x_j$
    end
    $x_i := (b_i - \text{sumod}) / a_{ii}$
end
```

$$\begin{bmatrix} a_{11} & \cdots & a_{1k} & \cdots & a_{1j} & \cdots & a_{1n} \\ & \ddots & \times & \times & \times & \times & \times \\ & & a_{kk} & \cdots & a_{kj} & \cdots & a_{kn} \\ & & & \ddots & \times & \times & \times \\ & & & & a_{ij} & \times & \times \\ & & & & & \ddots & \times \\ & & & & & & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \\ \\ \\ \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \\ \\ \\ \\ b_n \end{bmatrix}$$

2.1 Gauss Elimination

□ Floating Point Operation (FLOP) Counts

	Component	Operation		
		Multiplication	Addition	Division
Forward Elimination	Multiplier	n(n-1)/2		n-1
	Update	n(n-1)(2n-1)/6	n(n-1)(2n-1)/6	
	RHS	n(n-1)/2	n(n-1)/2	
Backward Substitution		n(n-1)/2	n(n-1)/2	n-1

$\propto n^3$

for half mesh size in 3D $N' = 8N$

$\rightarrow 8^3 = 2^9 = 512\, times\ in\, FLOPS$

$and\ Fill-in$ requires huge storage

Area of the square with decreasing size

$$\rightarrow \sum_{k=n-1}^{1} k^2$$

$$* \sum_{l=1}^{m} l^2 = \frac{1}{6} m(m+1)(2m+1)$$

$$\begin{array}{cccccccc}
 & a_{11} & \cdots & a_{1k} & \cdots & a_{1j} & \cdots & a_{1n} \\
 & & \ddots & \times & \times & \times & \times & \times \\
 & & & a_{kk} & \cdots & a_{kj} & \cdots & a_{kn} \\
 & & & \vdots & \ddots & \times & \times & \times \\
\longrightarrow & & & a_{ik} & \times & a_{ij} & \times & \times \\
 & & & \vdots & \times & \times & \ddots & \times \\
 & & & a_{nk} & \times & \times & \times & a_{nn}
\end{array}$$

2.2 LU Factorization Method

□Column Elimination in terms of Matrix Multiplication

$$A_k \equiv \begin{bmatrix} a_{11} & \cdots & a_{1k} & \cdots & a_{1j} & \cdots & a_{1n} \\ & \ddots & \times & \times & \times & \times & \times \\ & & a_{kk} & \cdots & a_{kj} & \cdots & a_{kn} \\ & & \vdots & \ddots & \times & \times & \times \\ & & a_{ik} & \times & a_{ij} & \times & \times \\ & & \vdots & \times & \times & \ddots & \times \\ & & a_{nk} & \times & \times & \times & a_{nn} \end{bmatrix}$$

$$M_k \equiv \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & -m_{k+1k} & \ddots & & & \\ & & -m_{ik} & & 1 & & \\ & & \vdots & & & \ddots & \\ & & -m_{nk} & & & & 1 \end{bmatrix}$$

$$R_i' = R_i - m_{ik} R_k \quad \forall i > k$$

$$\updownarrow$$

$$A_{k+1} = M_k A_k$$

$$\tilde{M}_k \equiv \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & m_{k+1k} & \ddots & & & \\ & & m_{ik} & & 1 & & \\ & & \vdots & & & \ddots & \\ & & m_{nk} & & & & 1 \end{bmatrix}$$

$$\tilde{M}_k A_{k+1} = A_k$$

$$\because R_i = R_i' + m_{ik} R_k'$$

$$\tilde{M}_k A_{k+1} = \tilde{M}_k M_k A_k = A_k$$

$$\rightarrow \tilde{M}_k = M_k^{-1}$$

2.2 LU Factorization Method

□ After the complete Elimination

$$A_n = M_{n-1} \cdots M_2 M_1 A_1 \xrightarrow[A_n \equiv U;\ A_1 = A]{} U = M_{n-1} \cdots M_2 M_1 A \longrightarrow \boxed{M_1^{-1} \cdots M_{n-2}^{-1} M_{n-1}^{-1} U = A}$$

$$M_{k-1}^{-1} M_k^{-1} \equiv \begin{bmatrix} 1 &&&&&& \\ & 1 &&&&& \\ & m_{kk-1} & 1 &&&& \\ & m_{k+1k-1} && \ddots &&& \\ & m_{ik-1} &&& 1 && \\ & \vdots &&&& \ddots & \\ & m_{nk-1} &&&&& 1 \end{bmatrix} \begin{bmatrix} 1 &&&&&& \\ & 1 &&&&& \\ && 1 &&&& \\ && m_{k+1k} & \ddots &&& \\ && m_{ik} && 1 && \\ && \vdots &&& \ddots & \\ && m_{nk} &&&& 1 \end{bmatrix} = \begin{bmatrix} 1 &&&&&& \\ & 1 &&&&& \\ & m_{kk-1} & 1 &&&& \\ & m_{k+1k-1} & m_{k+1k} & \ddots &&& \\ & m_{ik-1} & m_{ik} && 1 && \\ && \vdots &&& \ddots & \\ & m_{nk-1} & m_{nk} &&&& 1 \end{bmatrix}$$

$$\because R_i' = R_i + m_{ik} R_k$$

$$L \equiv M_1^{-1} \cdots M_{n-2}^{-1} M_{n-1}^{-1} = \begin{bmatrix} 1 &&&&&& \\ m_{21} & 1 &&&&& \\ m_{31} & m_{32} & 1 &&&& \\ m_{41} & m_{42} & m_{43} & \ddots &&& \\ &&&& 1 && \\ & \vdots &&&& \ddots & \\ m_{n1} & m_{n2} & m_{n3} &&& m_{nn-1} & 1 \end{bmatrix} \longrightarrow \boxed{A = LU}$$

2.2 LU Factorization Method

□ Advantage of LU Factorization

- **Repeated solution of Ax=b with several b's**

$LUx = b$

Let $y = Ux$

$Ly = b$

Forward Substitution $\longrightarrow$

```
$y_1 := b_n$
for i = 2 : n
    sumod = 0
    for j = 1 : i - 1
        sumod := sumod $- m_{ij} y_j$
    end
    $y_i := (b_i - \text{sumod})/(1)$
end
```

Now Solve $Ux = y$ for known y by back substitution! $\longleftarrow$

- **Operation Counts**
  - FLOPs for two substitutions only
    - 2*n(n-1)
  - Significantly less than elimination, particularly for large n

2.3 Pivoting

- Needs:
  - Pivot can become 0 during the elimination step leading to infinite multiplier
  - Truncation error may lead to incorrect solution when the multiplier becomes very large

$$\begin{aligned} 0.003\,x &+ 91.21\,y = 91.24 \\ 4.151\,x &- 6.09\,y = 35.42 \end{aligned}$$

4-digit arithmetic

$$m_{21} = \frac{4.151}{0.003} = 1383.666 \cong .1384\times 10^4$$

$$(-6.09 - 1384\times 91.21)y = 35.42 - \underbrace{1384\times 91.24}_{126300}$$

$$-126200y = -126300 \rightarrow y = 1.001$$

Back substitution

$$0.003x = 91.24 - \underbrace{91.21\times 1.001}_{91.30} = -0.06 \rightarrow x = -20.0$$

$$\begin{aligned} 4.151\,x &- 6.09\,y = 35.42 \\ 0.003\,x &+ 91.21\,y = 91.24 \end{aligned}$$

$$m_{21} = \frac{0.003}{4.151} = .7227\times 10^{-3}$$

$$(91.21 + \underbrace{6.09\times .7227\times 10^{-3}}_{0.004401})y$$

$$= 91.24 - \underbrace{35.42\times .7227\times 10^{-3}}_{0.02575} = 91.22$$

$$91.21y = 91.22 \rightarrow y = 1.000$$

Back substitution

$$4.151x = 35.42 + 6.09\times 1.000 = 41.51 \rightarrow x = 10.0$$

- Problems of small pivot
  - Large multiplier introduces large perturbation to the row being eliminated
  - Accumulated error can be amplified by a small pivot during backsubstitution
  - Keep the original row as much as possible using small multiplier or large pivot when the multiplier becomes very large

2.3 Pivoting

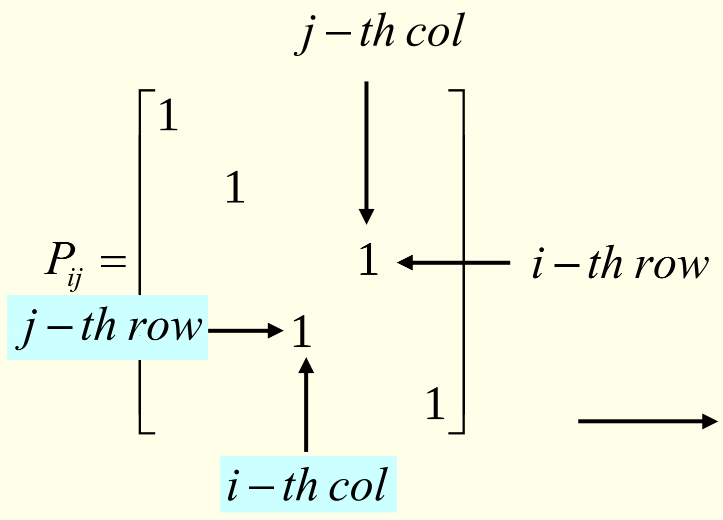
□Partial Pivoting

- During the k-th elimination step, choose the row among the remaining rows that would give the minimum multiplier as the pivoting row (find the largest pivot)
- Then exchange the k-th row and the selected row

□Pivoting in terms of Matrix

- Row exchange of Identity Matrix

```
$m_p = |a_{kk}|$
for i = k + 1 : n
    $m_i = |a_{ik}|$
    if $m_i > m_p$
        $ip = i$
        $m_p = m_i$
    end
end
```



$$\longrightarrow \tilde{A}_k = P_{ij} A_k$$

$$P_{ij}^T = P_{ji} = P_{ij}$$

2.3 Pivoting

□Properties of Pivoting Matrix

- Multiple Exchange: k-l and i-j th rows exchange

$$P = P_{kl}P_{ij}$$

- Identical Inverse for two row exchange, orthonormal in general

$$I = P_{ij}P_{ij} \Rightarrow P_{ij}^{-1} = P_{ij} \qquad PP^T = I \Rightarrow P^T = P^{-1}$$

□LU Factorization after Pivoting

- Suppose that pivoting sequence is known after elimination as P

$$PA = LU \qquad P = P_{n-1}P_{n-2}\cdots P_2P_1$$

- How to generate L from the result of Gauss Elimination?

First elimination after pivoting by $P_1 : M_1P_1A_1 = A_2 \Rightarrow P_1A_1 = L_1A_2$

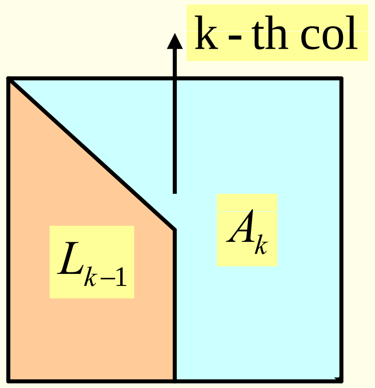
Suppose that $\Pi_{k-1}A = L_{k-1}A_k$

$\Pi_{k-1} = P_{k-1}\cdots P_2P_1$: Permutation Matrix after (k-1)th Step

L_{k-1} = Lower Triangular Matrix after (k-1)th step

A_k = Upper Triangular Matrix as the result of (k-1)th step

$\Rightarrow L_{k-1}^{-1}\Pi_{k-1}A = A_k$



SNURPL

2.3 Pivoting

☐LU Factorization after Pivoting

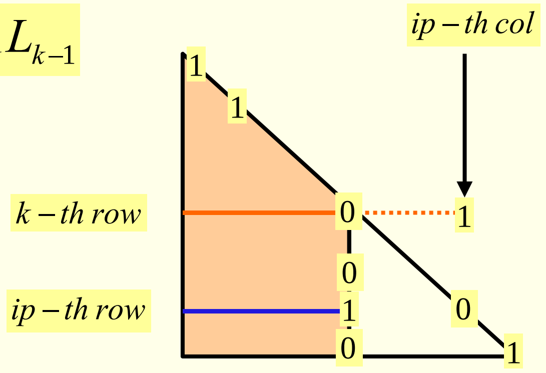
- **After k-th step of elimination**

$$M_k P_k A_k \equiv A_{k+1}$$

$$M_k P_k L_{k-1}^{-1} \Pi_{k-1} A = A_{k+1}$$

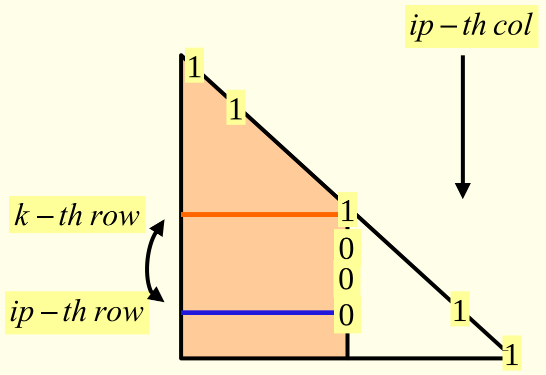
$$\Pi_{k-1} A = L_{k-1} P_k^{-1} M_k^{-1} A_{k+1}$$
$$P_k \Pi_{k-1} A = P_k L_{k-1} P_k M_k^{-1} A_{k+1}$$
$$\Pi_k A = \left(P_k L_{k-1} P_k\right) M_k^{-1} A_{k+1}$$

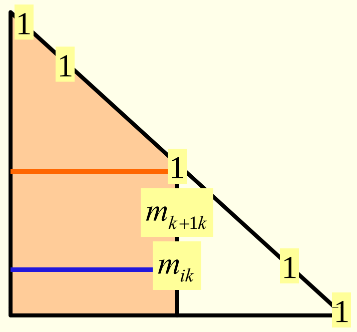
$P_k L_{k-1}$



$(P_k L_{k-1}) P_k \Rightarrow$ Column Exchange

$(P_k L_{k-1} P_k) M_k^{-1} \equiv L_k \longrightarrow \Pi_k A = L_k A_{k+1}$





Perform row exchange in L after each partial pivoting and add the multiplier to the k - th col.

2.3 Pivoting

☐ Scaled Pivoting

- **When some elements on a row is exceptionally larger than the other rows, partial pivoting is not sufficient**

$$\begin{array}{ccccc} 10x & + & 10^7 y & = & 10^7 \\ x & + & y & = & 2 \end{array} \qquad \Leftrightarrow 10^{-6}x + y = 1$$

multiplier

$$m_{21} = 10^{-1}$$

5-digit arithmetic

$$(1\text{-}10^6)y = 2 - 10^6 \Rightarrow 10^6 y = 10^6 \Rightarrow y = 1$$

back substitution

$$x = 0$$

$\longleftarrow$ small pivot case can be hidden by merely mutiplipiying a big number to the k-th row equation

- **Divide each row by the maximum absolute value of the row**
- **Then do the partial pivoting**

☐ Complete Pivoting

- **Exchange columns as well as rows using the maximum entry of A_k as the pivot**
- **Best but expensive for large matrices**

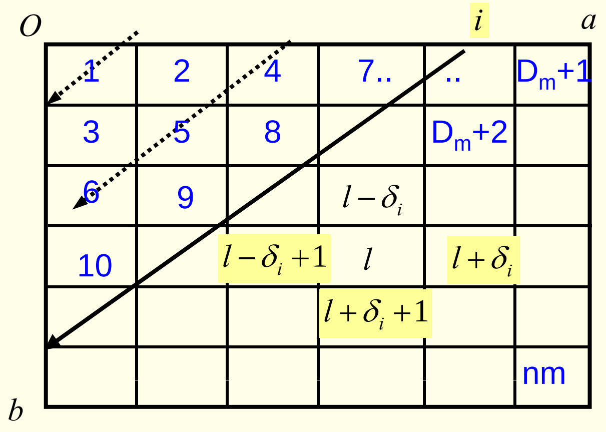
2.4 Reordering

Purpose

- **Minimize fill-in during the elimination to save memory and flops**

Cuthill-McKee Ordering

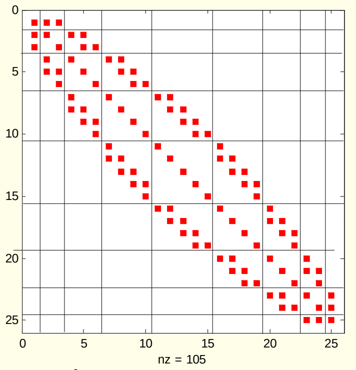
- **Minimize bandwidth**



δ_i = number of meshes in the i-th diagonal

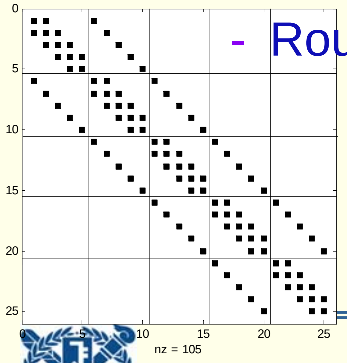
for $i < m$

$\delta_i = i$



$$D_i = \sum_{k=1}^{i-1} \delta_i = \frac{(i-1)(i-2)}{2}$$

width of the nonzero elements increases with diagonal location index



- Rough estimate of fill-ins in case of square domain (m=n>>1)

$$\mathrm{Fi} = 2\times\sum_{k=0}^{m-1} k = (m-1)(m-2) \qquad < \mathrm{Fi}_{\text{Natural Ordering}} \approx m\times N = m^3$$

2.4 Reordering

☐ Minimum Degree Ordering

- **Determine the row and column that would have minimum number of nonzeros during Gauss elimination at the k-th step**
  - At the k-th elimination step, count non-zero entries at each remaining rows, then store to nzr(i)
  - Count non-zero entries at each column, then store to nzc(j)
  - Find location where (nzr(i)-1)*(nzc(j)-1) becomes minimum which is the number of fillin's when a(i,j) is chosen as pivot
  - Exchange row k and row i
  - Exchange col k and col j
  - Perform elimination with the permuted A_k
- **Will cost much for large matrices**
  - Approximate Minimum Degree Ordering available

$$\begin{matrix}
a_{11} & \cdots & a_{1k} & \cdots & a_{1j} & \cdots & a_{1n} \\
 & \ddots & \times & \times & \times & \times & \times \\
 & & a_{kk} & \cdots & a_{kj} & \cdots & a_{kn} \\
 & & \vdots & \ddots & \times & \times & \times \\
 & & a_{ik} & \times & a_{ij} & \times & \times \\
 & & \vdots & \times & \times & \ddots & \times \\
 & & a_{nk} & \times & \times & \times & a_{nn}
\end{matrix}$$

3.1 Introduction to Iterative Solution Methods

- ☐ Solution of Linear System by Iterative Update of Solution Vector starting from an Arbitrary Guess
- ☐ Generic Approach by Matrix Splitting when solving Ax=b

Let $A = M + N$

$$(M+N)x = b$$

$$Mx = b - Nx$$

Establish an iteration scheme

$$Mx^{(k)} = b - Nx^{(k-1)} = \tilde{b} \cdots (1)$$

Solve instead of $Ax = b$ after choosing M such that $Mx^{(k)} = \tilde{b}$ much easier to solve than $Ax = b$

$$e.g.\, M = D$$

Would it work?

Unconditionally?

Let x^* be the true solution.

$$Mx^* = b - Nx^* \cdots (2)$$

$(1)-(2)$

$$M(\underbrace{x^{(k)} - x^*}_{e^{(k)}}) = -N(x^{(k-1)} - x^*)$$

$$Me^{(k)} = -Ne^{(k-1)} \qquad e^{(k)} = \underbrace{-M^{-1}N}_{T} e^{(k-1)}$$

$$e^{(k)} = Te^{(k-1)} \longrightarrow e^{(k)} = T^k e^{(0)}$$

3.1 Introduction to Iterative Solution Methods

☐Convergence Condition

Let λ_i, u_i be the i-th eigenvalue and eigenvector of T

$e^{(0)} = c_1u_1 + c_2u_2 + \cdots + c_nu_n \quad \because u_i$'s are linearly independent.

$$T^k e^{(0)} = T^k(c_1u_1 + c_2u_2 + \cdots + c_nu_n)$$
$$= c_1T^ku_1 + c_2T^ku_2 + \cdots + c_nT^ku_n$$
$$= c_1\lambda_1^ku_1 + c_2\lambda_2^ku_2 + \cdots + c_n\lambda_n^ku_n \quad \to 0 \text{ as k increases} \quad \Rightarrow e^{(k)} \to 0$$

$$\downarrow$$

$$\text{if } |\lambda_i| < 1 \quad \forall i \qquad x^{(k)} \to x^*$$

$$\rho(T) = \max_i(|\lambda_i|) < 1 \to \text{Spectral Radius}$$

$$= \lambda_1^k\left(c_1u_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^k c_2u_2 + \left(\frac{\lambda_3}{\lambda_1}\right)^k c_3u_3 + \right)$$

$\cong \lambda_1^kc_1u_1$ for sufficiently large k with $\lambda_1 = \rho$ being the largest eigenvalue

or $e^k \cong \rho e^{k-1}$ $\leftarrow$ only fundamental mode remaining in e^{k-1}

2.2 Jacobi and Gauss Seidel Method

□ Jacobi Method

$A = M + N$

Let $M = D$, the diagonal matrix consisting of the diagonal entries of A

$N = A - D$

$$Mx^{(k)} = b - Nx^{(k-1)} \longrightarrow Dx^{(k)} = b - (A-D)x^{(k-1)}$$

$$d_i x_i^{(k)} = b_i - \left(\sum_{j=1}^{i-1} a_{ij} x_j^{(k-1)} + \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \right) = \tilde{b}_i \longrightarrow x_i^{(k)} = \frac{\tilde{b}_i}{d_i}$$

□ Gauss-Seidel Method

Let $A = L + D + U$, the lower tridiagonal, diagonal, and upper tridiagonal matrix of A

$M = L + D, \quad N = U$

$(L+D)x^{(k)} = b - Ux^{(k-1)}$

$$d_i x_i^{(k)} = b_i - \left(\sum_{j=1}^{i-1} a_{ij} x_j^{(k)} + \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \right) = \tilde{b}_i$$

$$\sum_{j=1}^{i-1} a_{ij} x_j^{(k)} + d_i x_i^{(k)} = b_i - \left(\sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \right) \qquad x_i^{(k)} = \frac{\tilde{b}_i}{d_i}$$

2.2 Jacobi and Gauss Seidel Method

□ Iteration Matrices

- **Jacobi**

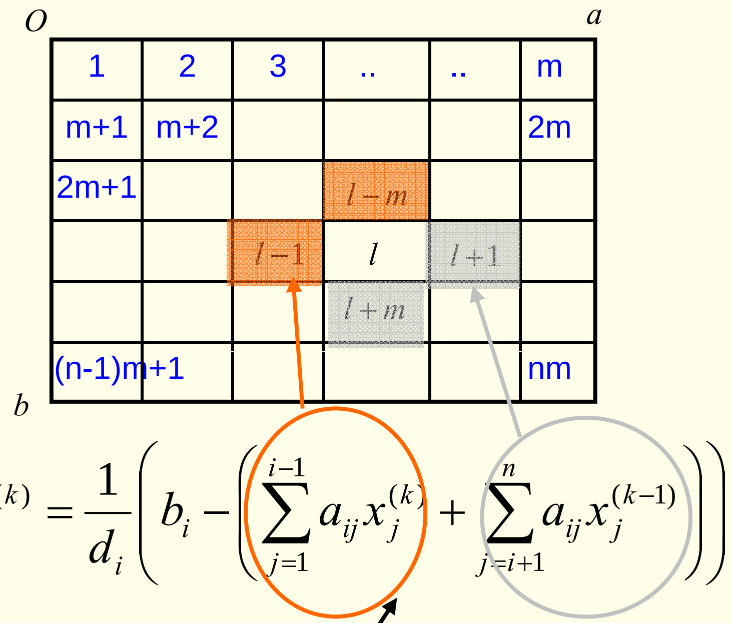
$$T=-M^{-1}N=-D^{-1}(A-D)=I-D^{-1}A$$

- **Gauss-Seidel**

$$T=-M^{-1}N=-(L+D)^{-1}U$$

□ When Converge?

$$|d_i|>\sum_{\substack{j=1\\ j\neq i}}^{n}|a_{ij}| \Rightarrow \text{Diagonally Dominant!}$$



$$x_i^{(k)}=\frac{1}{d_i}\left(b_i-\left(\sum_{j=1}^{i-1}a_{ij}x_j^{(k)}+\sum_{j=i+1}^{n}a_{ij}x_j^{(k-1)}\right)\right)$$

□ Geometrical Interpretation

- **Sweeping over nodes by updating local value by**
- **Off-diagonal entries = Summation of neighbor nodes' contribution**
- **Converges when self effect is stronger than neighbors' effect**

Diagonal Dominance of Jacobi Iteration Matrix

- Jacobi Iteration Matrix T for a Diagonally Dominant Matrix

$$s_i^{OD} = \sum_{j\neq i} \left| a_{ij} \right| < a_{ii} \quad \forall i$$

$$T = \left[t_{ij} \right] = -D^{-1}\left(L+U \right) = \left[-\frac{a_{ij}}{a_{ii}} \right] \qquad \rightarrow t_{ii} = 0, \sum_{j\neq k} \left| t_{ij} \right| < 1$$

  – Let u be an eigenvector of T with λ being the corresponding eigenvalue $\rightarrow$ $Tu = \lambda u$

- Max Eigenvalue of T?

$u = \left[u_1, \cdots, u_k, \cdots, u_n \right]$

Let u_k be the maximum element (Absolute)

  – k-th row $\quad \sum_j t_{kj} u_j = \lambda u_k$

$$\sum_{j\neq k} t_{kj} u_j = \lambda u_k \qquad \because t_{kk} = 0$$

$$\left| \lambda \right| \left| u_k \right| \le \sum_{j\neq k} \left| t_{kj} \right| \left| u_j \right| \le \sum_{j\neq k} \left| t_{kj} \right| \left| u_k \right| \rightarrow \left| \lambda \right| \le \sum_{j\neq k} \left| t_{kj} \right| < 1$$ $\rightarrow$ Jacobi converges for diagonally dominant matrix

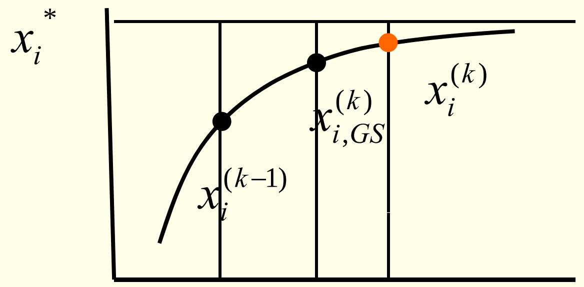
$$e_i^{(k)} = \frac{1}{d_i} \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij} e_i^{(k-1)} < \frac{1}{d_i} \sum_{\substack{j=1 \\ j\neq i}}^{n} \left| a_{ij} \right| e_{max}^{(k-1)} < e_{max}^{(k-1)} \quad if \ \frac{1}{d_i} \sum_{\substack{j=1 \\ j\neq i}}^{n} \left| a_{ij} \right| < 1 \quad \rightarrow e_i^{(k)} \text{ reduces as } k \uparrow$$

6.2.3 Successive Over-Relaxation Method

□Extrapolation of G-S Estimate of Solution for each element

$$x_i^{(k)} = \omega x_{i,GS}^{(k)} + (1-\omega)x_i^{(k-1)} \qquad \text{with } \omega \geq 1.0$$

$\rightarrow$ give more weight to G-S estimates



$$x_{i,GS}^{(k)} = \frac{1}{d_i}\left(b_i - \left(\sum_{j=1}^{i-1} a_{ij}x_j^{(k)} + \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}\right)\right)$$

$$= \left[D^{-1}\left(b - Lx^{(k)} - Ux^{(k-1)}\right)\right]_i$$

$$x_i^{(k)} = \left[\omega D^{-1}\left(b - Lx^{(k)} - Ux^{(k-1)}\right) + (1-\omega)Ix^{(k-1)}\right]_i$$

$$x^{(k)} = \omega\left(D^{-1}(b - Lx^{(k)} - Ux^{(k-1)})\right) + (1-\omega)Ix^{(k-1)}$$

$$Dx^{(k)} = \omega\left(b - Lx^{(k)} - Ux^{(k-1)}\right) + (1-\omega)Dx^{(k-1)}$$

$$\begin{tikzcd}
 & \begin{aligned} N &= \omega A - D - \omega L \\ &= \omega(D+L+U) - D - \omega L \\ &= -(1-\omega)D + \omega U \end{aligned} & \\
 & \omega A \text{를 } M = D + \omega L \text{과 나머지}(N)\text{로 분리} \arrow[u] \arrow[dl] & \\
(D+\omega L)x^{(k)} = \omega b - \omega Ux^{(k-1)} + (1-\omega)Dx^{(k-1)} \arrow[r, "\text{If we put } x^{(k)} = x^{(k-1)} = x"'] & \omega(D+L+U)x = \omega b \arrow[r] & \omega Ax = \omega b \arrow[ul]
\end{tikzcd}$$

SNURPL

6.2.3 Successive Over-Relaxation Method

☐ In terms of Matrix Splitting

- **Split Matrix**

$$\omega A = \omega(L + D + U) = (D + \omega L) + (\omega U + \omega D - D)$$

- **Apply to Modified Linear System**

$$\omega A x = \omega b$$

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

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

$$x^{(k)} = \omega D^{-1}\left(b - L x^{(k)} - U x^{(k-1)}\right) + (1-\omega) x^{(k-1)} \longrightarrow$$

Extrapolation Form

- **Iteration Matrix**

$$x^{(k)} = (D + \omega L)^{-1}\left(\omega b - \omega U x^{(k-1)} + (1-\omega) D x^{(k-1)}\right)$$
$$= -(D + \omega L)^{-1}(\omega U - (1-\omega) D) x^{(k-1)} + (D + \omega L)^{-1} \omega b$$

$$\longrightarrow \quad T = -(D + \omega L)^{-1}(\omega U - (1-\omega) D)$$

$\leftarrow$ spectral radius depends on ω!

2.3 Successive Over-Relaxation Method

□ Eigenvalues of Iteration Matrix

Let λ_i be i-th eigenvalue of T. Then $(-1)^n \prod_{i=1}^{n} \lambda_i = Det(T)$.

$$Det(T) = Det\left(-(D+\omega L)^{-1}(\omega U - (1-\omega)D)\right) = Det\left(-(D+\omega L)^{-1}\right) Det\left(\omega U - (1-\omega)D\right)$$

$$Let\ D = \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix},\ (D+\omega L) = \begin{bmatrix} d_1 & & & \\ * & d_2 & & \\ * & * & \ddots & \\ * & * & * & d_n \end{bmatrix},\ (D+\omega L)^{-1} = \begin{bmatrix} 1/d_1 & & & \\ * & 1/d_2 & & \\ * & * & \ddots & \\ * & * & * & 1/d_n \end{bmatrix}$$

$\uparrow$ Gauss – Jordan Elimination to Find Inverse...

$$\det(D+\omega L)^{-1} = \prod_{i=1}^{n} 1/d_i$$

$$-(\omega U - (1-\omega)D) = \begin{bmatrix} (1-\omega)d_1 & * & * & * \\ & (1-\omega)d_2 & * & * \\ & & \ddots & * \\ & & & (1-\omega)d_n \end{bmatrix},\quad \det(\omega U - (1-\omega)D) = (1-\omega)^n \prod_{i=1}^{n} d_i$$

$\longrightarrow$

$$\Rightarrow \prod_{i=1}^{n} \lambda_i = (1-\omega)^n$$

2.3 Successive Over-Relaxation Method

☐ Optimum Over-relaxation Parameter

Significance of $\prod_{i=1}^{n} \lambda_i = (1-\omega)^n \quad |1-\omega| < 1 \rightarrow 0 < \omega < 2$ in order to $|\lambda_i| < 1 \, \forall i$

The width of eigenvalue spectrum $\left(1 > |\lambda_1| > |\lambda_2| >> |\lambda_n| > 0\right)$ can be minimized by adjusting ω.

$|\lambda_1|$ can be minimum when the width is 0.0. i.e. $|\lambda_1| = |\lambda_n| = \cdots = |\lambda_n|$

Thus at the optimum point, $\lambda^n = (1-\omega)^n \rightarrow |\lambda| = |1-\omega|$

☐ For Block Tri-Diagonal Matrix (2 cyclic Matrices)

$(\lambda + \omega - 1)^2 = \lambda\omega^2\mu^2 \leftarrow$ David Young, 1950, $\mu = \rho_{\text{Jac}} = \rho\left(D^{-1}(L+U)\right)$

$\omega = 1 \rightarrow \lambda = \mu^2 = \rho_{GS} \rightarrow$ GS 2 times more effective than Jacobi!

$$4(\omega-1)^2 = (\omega-1)\omega^2\mu^2 \longrightarrow \omega_{opt} = \frac{2(1-\sqrt{1-\mu^2})}{\mu^2} = \frac{2(1-\sqrt{1-\rho_{GS}})}{\rho_{GS}} \longrightarrow \rho_{opt} = \omega_{opt} - 1$$

2.3 Successive Over-Relaxation Method

☐ Effectiveness of SOR

$$\text{For } \rho_{\text{Jac}} = 0.99,\ \rho_{\text{GS}} = 0.99^2 = 0.98$$

$$\omega = \frac{2(1-\sqrt{1-0.98})}{0.98} = 1.75$$

$$\rho_{SOR} = 0.75 \approx 0.98^{14}$$

- **1 step of SOR is as efficient as ~15 steps of Gauss-Seidel (or 30 steps of Jacobi)**