3D Code for Static Linear-Elastic Problems (3/3) Linear Solver

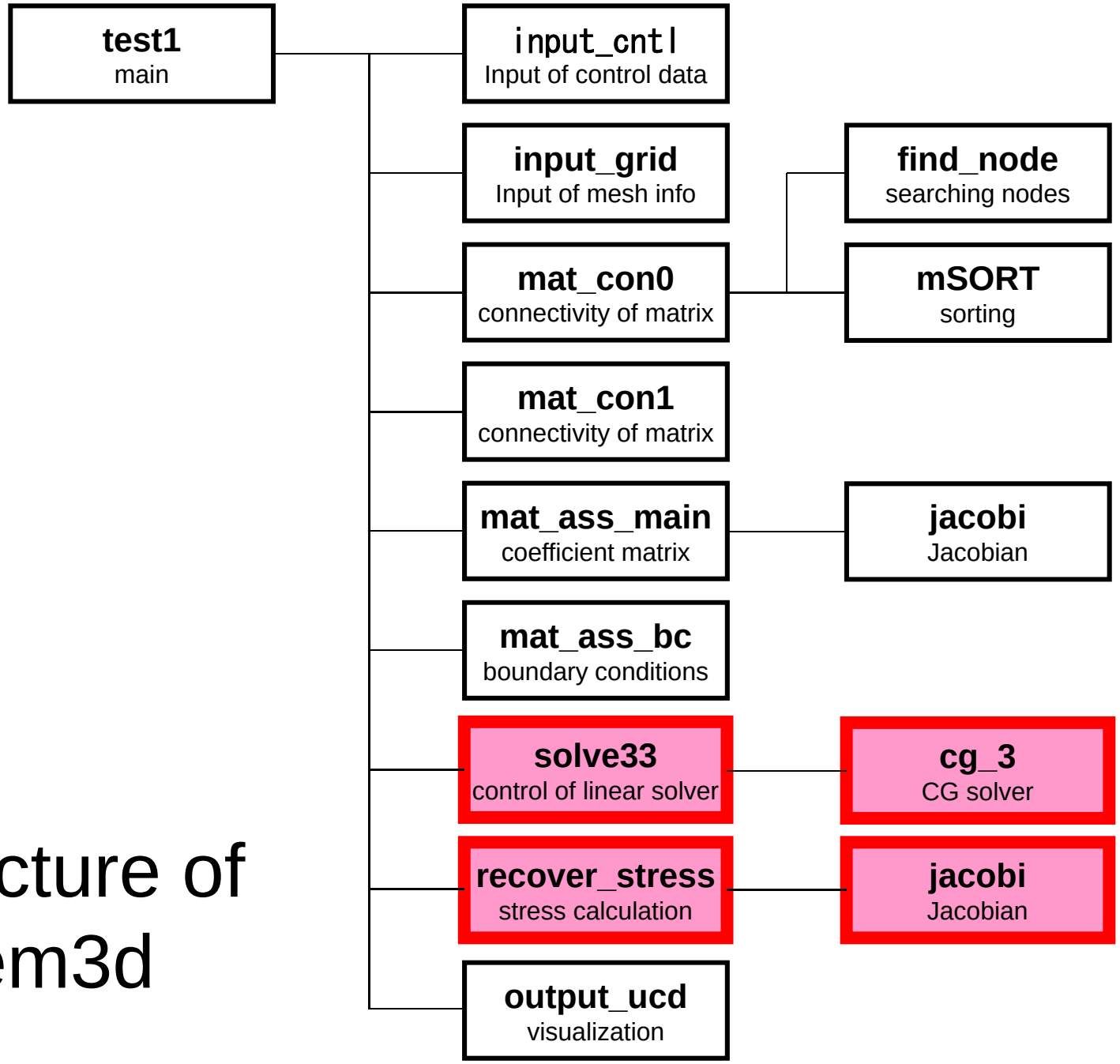
Kengo Nakajima
Information Technology Center

Technical & Scientific Computing I (4820-1027)
Seminar on Computer Science I (4810-1204)

FEM Procedures: Program

- Initialization
  - Control Data
  - Node, Connectivity of Elements (N: Node#, NE: Elem#)
  - Initialization of Arrays (Global/Element Matrices)
  - Element-Global Matrix Mapping (Index, Item)
- Generation of Matrix
  - Element-by-Element Operations (do icel= 1, NE)
    - Element matrices
    - Accumulation to global matrix
  - Boundary Conditions
- Linear Solver
  - Conjugate Gradient Method
- Calculation of Stress

Structure of fem3d

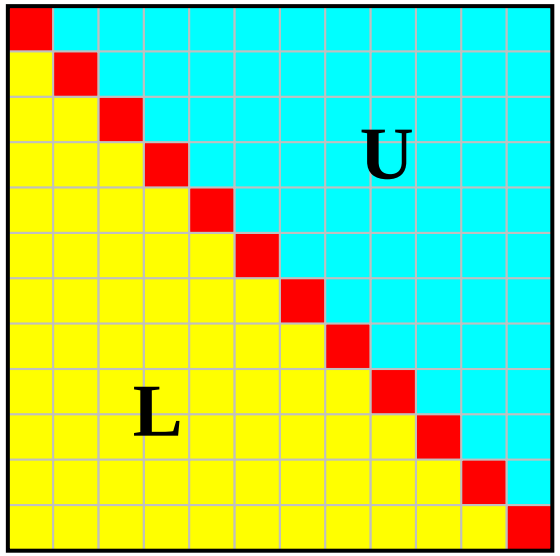


Main Part

```
#include <stdio.h>
#include <stdlib.h>
FILE* fp_log;
#define GLOBAL_VALUE_DEFINE
#include "pfem_util.h"

extern void INPUT_CNTL();
extern void INPUT_GRID();
extern void MAT_CON0();
extern void MAT_CON1();
extern void MAT_ASS_MAIN();
extern void MAT_ASS_BC();
extern void SOLVE33();
extern void RECOVER_STRESS();
extern void OUTPUT_UCD();
int main()
{

/** Logfile for debug **/
        if( (fp_log=fopen("log.log","w")) == NULL){
                fprintf(stdout,"input file cannot be opened!¥n");
                exit(1);
        }

        INPUT_CNTL();
        INPUT_GRID();


        MAT_CON0();
        MAT_CON1();

        MAT_ASS_MAIN();
        MAT_ASS_BC()   ;

        SOLVE33();

        RECOVER_STRESS();
        OUTPUT_UCD()    ;
}
```

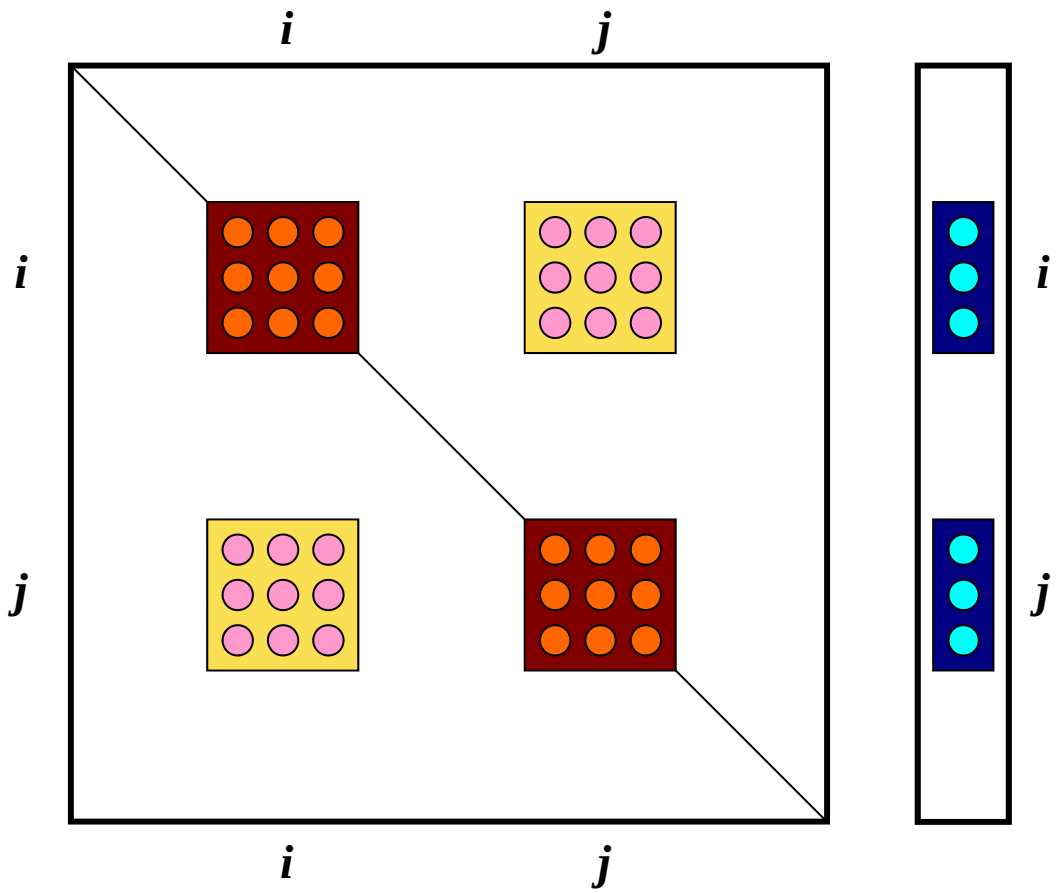
Some Features of “fem3d”

- Non-Zero Off-Diagonals
  - Upper/Lower triangular components are stored separately.

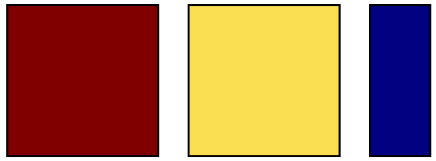


- Stored as Block
  - Vector: 3-components per node
  - Matrix: 9-components per block
  - Processed as “block” based on 3 variables on each node (not each component of matrix)

Definitions of Terms



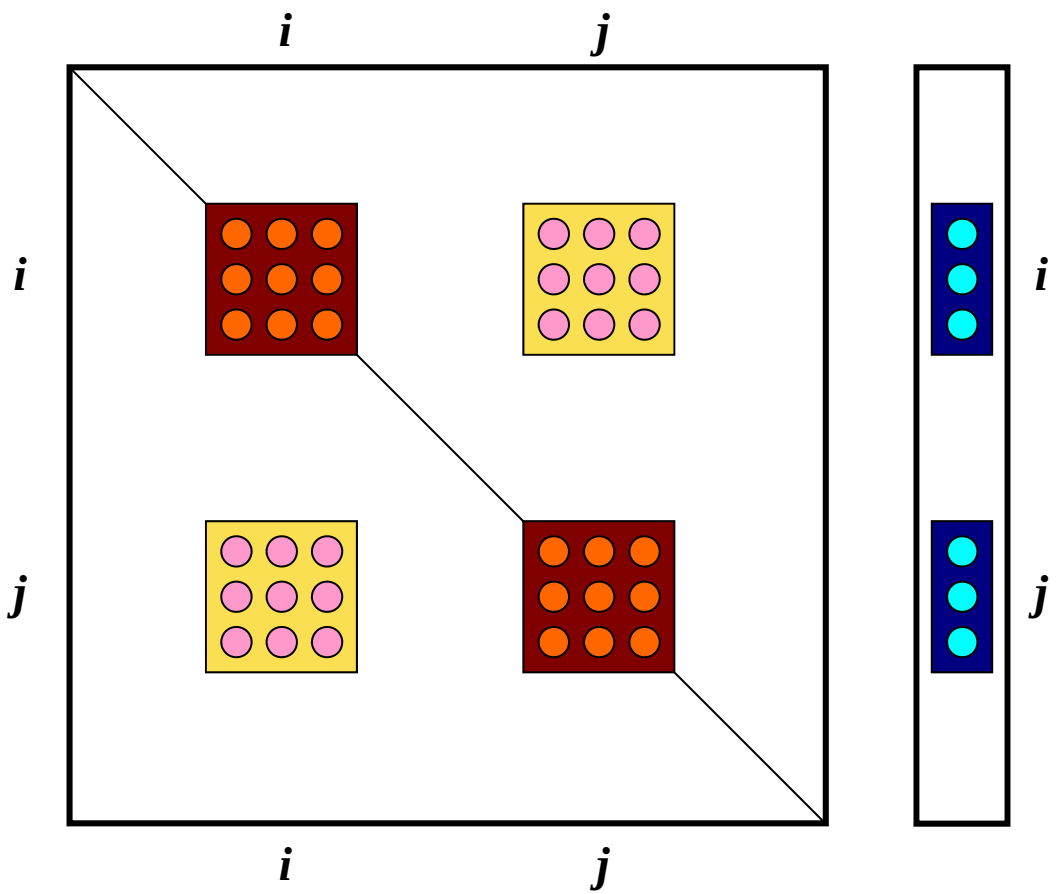
Block (Node):
ブロック（節点）:



Component (DOF):
成分（自由度）:

Storing 3x3 Block (1/3)

- Less memory requirement
  - Index, Item



Storing 3x3 Block (2/3)

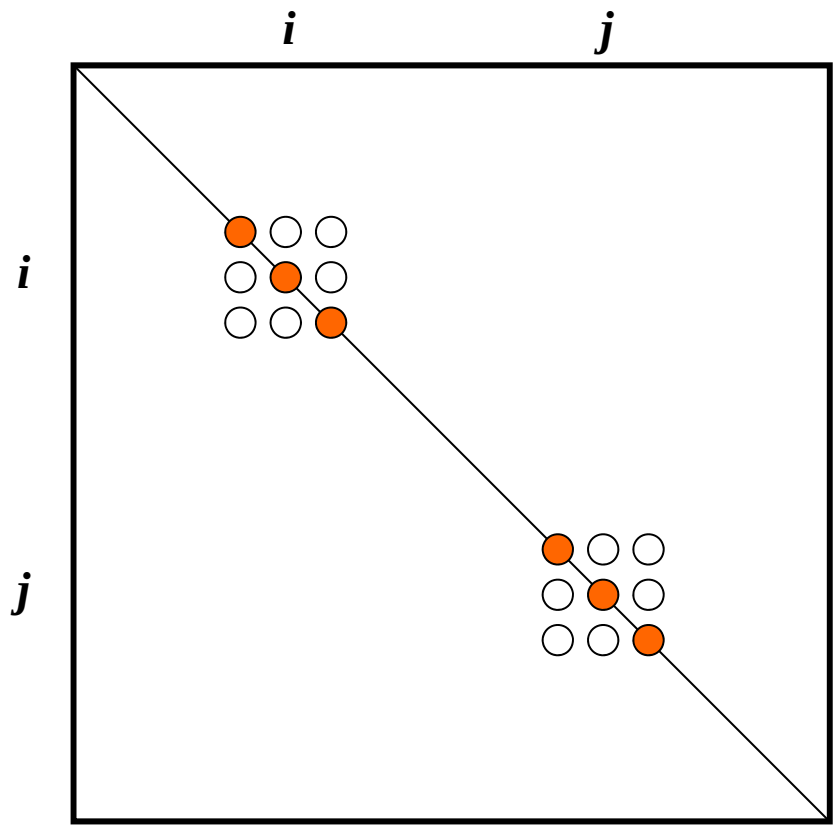
- Computational Efficiency
  - Ratio of (Comptation/Indirect Memory Access) is larger
  - >2x speed-up both for vector/scalar processors
    - Contiguous memory access, Cache Utilization, Larger Flop/Byte

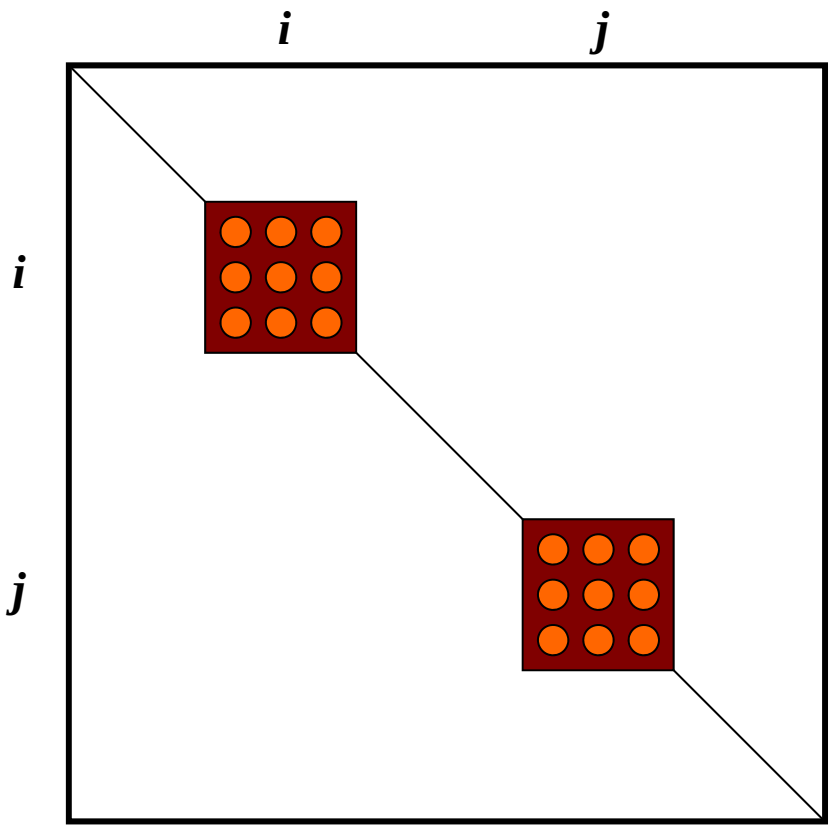
```
do i= 1, 3*N
  Y(i)= D(i)*X(i)
  do k= index(i-1)+1, index(i)
    kk= item(k)
    Y(i)= Y(i) + AMAT(k)*X(kk)
  enddo
enddo
```

```
do i= 1, N
  X1= X(3*i-2)
  X2= X(3*i-1)
  X3= X(3*i)
  Y(3*i-2)= D(9*i-8)*X1+D(9*i-7)*X2+D(9*i-6)*X3
  Y(3*i-1)= D(9*i-5)*X1+D(9*i-4)*X2+D(9*i-3)*X3
  Y(3*I  )= D(9*i-2)*X1+D(9*i-1)*X2+D(9*I  )*X3
  do k= index(i-1)+1, index(i)
    kk= item(k)
    X1= X(3*kk-2)
    X2= X(3*kk-1)
    X3= X(3*kk)
    Y(3*i-2)= Y(3*i-2)+AMAT(9*k-8)*X1+AMAT(9*k-7)*X2 &
                                     +AMAT(9*k-6)*X3
    Y(3*i-1)= Y(3*i-1)+AMAT(9*k-5)*X1+AMAT(9*k-4)*X2 &
                                     +AMAT(9*k-3)*X3
    Y(3*I  )= Y(3*I  )+AMAT(9*k-2)*X1+AMAT(9*k-1)*X2 &
                                     +AMAT(9*k  )*X3
  enddo
enddo
```

Storing 3x3 Block (3/3)

- Stabilization of Computation
  - Instead of division by diagonal components, full LU factorization of 3x3 Diagonal Block is applied.
  - Effective for ill-conditioned problems





Global Variables: pfem_util.h (1/3)

Name	Type	Size	I/O	Definition
fname	C	[80]	I	Name of mesh file
N,NP	I		I	# Node
ICELTOT	I		I	# Element
NODGRPtot	I		I	# Node Group
XYZ	R	[N][3]	I	Node Coordinates
ICELNOD	I	[ICELTOT][8]	I	Element Connectivity
NODGRP_INDEX	I	[NODGRPtot+1]	I	# Node in each Node Group
NODGRP_ITEM	I	[NODGRP_INDEX[NODGRPTOT+1]]	I	Node ID in each Node Group
NODGRP_NAME	C80	[NODGRP_INDEX[NODGRPTOT+1]]	I	Name of NodeGroup
NL, NU	I		O	# Upper/Lower Triangular Non-Zero Off-Diagonals at each node
NPL, NPU	I		O	# Upper/Lower Triangular Non-Zero Off-Diagonals
D	R	[9*N]	O	Diagonal Block of Global Matrix
B,X	R	[3*N]	O	RHS, Unknown Vector

Global Variables: pfem_util.h (2/3)

Name	Type	Size	I/O	Definition
ALUG	R	[9*N]	O	Full LU factorization of Diagonal Blocks D
AL,AU	R	[9*NPL],[9*NPU]	O	Upper/Lower Triangular Components of Global Matrix
indexL, indexU	I	[N+1]	O	# Non-Zero Off-Diagonal Blocks
itemL, itemU	I	[NPL], [NPU]	O	Column ID of Non-Zero Off-Diagonal Blocks
INL, INU	I	[N]	O	Number of Off-Diagonal Blocks at Each Node
IAL, IAU	I	[N][NL],[N][NU]	O	Off-Diagonal Blocks at Each Node
IWKX	I	[N][2]	O	Work Arrays
METHOD	I		I	Iterative Method (fixed as 1)
PRECOND	I		I	Preconditioning Method (0: SSOR, 1: Block Diagonal Scaling)
ITER, ITERactual	I		I	Number of CG Iterations (MAX, Actual)
RESID	R		I	Convergence Criteria (fixed as 1.e-8)
SIGMA_DIAG	R		I	Coefficient for LU Factorization (fixed as 1.00)
pfemIarray	I	[100]	O	Integer Parameter Array
pfemRarray	R	[100]	O	Real Parameter Array

Global Variables: pfem_util.h (3/3)

Name	Type	Size	I/O	Definition
O8th	R		I	$= 0.125$
PNQ, PNE, PNT	R	[2][2][8]	O	$\frac{\partial N_i}{\partial \xi}, \frac{\partial N_i}{\partial \eta}, \frac{\partial N_i}{\partial \zeta} (i = 1 \sim 8)$ at each Gaussian Quad. Point
POS, WEI	R	[2]	O	Coordinates, Weighting Factor at each Gaussian Quad. Point
NCOL1, NCOL2	I	[100]	O	Work arrays for sorting
SHAPE	R	[2][2][2][8]	O	N_i (i=1~8) at each Gaussian Quad Point
PNX, PNY, PNZ	R	[2][2][2][8]	O	$\frac{\partial N_i}{\partial x}, \frac{\partial N_i}{\partial y}, \frac{\partial N_i}{\partial z} (i = 1 \sim 8)$ at each Gaussian Quad. Point
DETJ	R	[2][2][2]	O	Determinant of Jacobian Matrix at each Gaussian Quad. Point
ELAST, POISSON	R		I	Young's Modulus, Poisson's Ratio
SIGMA_N, TAU_N	R	[N][3]	O	Normal/Shear Stress at each Node

- Preconditioning
- Linear Solver in “fem3D”
- Computation of Stress
- Report #2

Preconditioning for Iterative Solvers

- Convergence rate of iterative solvers strongly depends on the spectral properties (eigenvalue distribution) of the coefficient matrix **A**.
  - Eigenvalue distribution is small, eigenvalues are close to 1
  - In "ill-conditioned" problems, "condition number" (ratio of max/min eigenvalue if **A** is symmetric) is large（条件数）.
- A preconditioner **M** (whose properties are similar to those of **A**) transforms the linear system into one with more favorable spectral properties（前処理）
  - **M** transforms **Ax=b** into **A'x=b'** where $\mathbf{A'=M^{-1}A}$, $\mathbf{b'=M^{-1}b}$
  - If **M~A**, $\mathbf{M^{-1}A}$ is close to identity matrix.
  - If $\mathbf{M^{-1}=A^{-1}}$, this is the best preconditioner (Gaussian Elim.)
  - Generally, **A'x'=b'** where $\mathbf{A'=M_L^{-1}AM_R^{-1}}$, $\mathbf{b'=M_L^{-1}b}$, $\mathbf{x'=M_Rx}$
  - $\mathbf{M_L/M_R}$: Left/Right Preconditioning（左／右前処理）

Preconditioned CG Solver (PCG)

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    solve [M]z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

$[M]= [M_1][M_2]$

$[A']x'=b'$

$[A']=[M_1]^{-1}[A][M_2]^{-1}$

$x'=[M_2]x,\ b'=[M_1]^{-1}b$

$p' \Rightarrow [M_2]p,\ r' \Rightarrow [M_1]^{-1}r$

$p'^{(i)}= r'^{(i-1)} + \beta'_{i-1}\ p'^{(i-1)}$

$[M_2]p^{(i)}= [M_1]^{-1}r^{(i-1)} + \beta'_{i-1}\ [M_2]p^{(i-1)}$

$p^{(i)}= [M_2]^{-1}[M_1]^{-1}r^{(i-1)} + \beta'_{i-1}\ p^{(i-1)}$

$p^{(i)}= [M]^{-1}r^{(i-1)} + \beta'_{i-1}\ p^{(i-1)}$

$\beta'_{i-1}= ([M]^{-1}r^{(i-1)},r^{(i-1)})/$
$\quad ([M]^{-1}r^{(i-2)},r^{(i-2)})$

$\alpha'_{i-1}= ([M]^{-1}r^{(i-1)},r^{(i-1)})/$
$\quad (p^{(i-1)},[A]p^{(i-1)})$

Preconditioned CG Solver (PCG)

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    solve [M]z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

Solving the following equation:

$$\{z\} = [M]^{-1}\{r\}$$

"Approximate Inverse Matrix"
（近似逆行列）

$$[M]^{-1} \approx [A]^{-1}, \quad [M] \approx [A]$$

Ultimate Preconditioning:
Inverse Matrix

$$[M]^{-1} = [A]^{-1}, \quad [M] = [A]$$

Diagonal Scaling: Simple but weak

$$[M]^{-1} = [D]^{-1}, \quad [M] = [D]$$

Diagonal Scaling, Point-Jacobi

$$[M]=\begin{bmatrix} D_1 & 0 & \dots & 0 & 0 \\ 0 & D_2 & & 0 & 0 \\ \dots & & \dots & & \dots \\ 0 & 0 & & D_{N-1} & 0 \\ 0 & 0 & \dots & 0 & D_N \end{bmatrix}$$

- **solve $[M]z^{(i-1)}= r^{(i-1)}$** is very easy.
- Provides fast convergence for simple problems.
- 1d.f, 1d.c

ILU(0), IC(0)

- Widely used Preconditioners for Sparse Matrices
  - Incomplete LU Factorization（不完全LU分解）
  - Incomplete Cholesky Factorization (for Symmetric Matrices)（不完全コレスキー分解）

- Incomplete Direct Method
  - Even if original matrix is sparse, inverse matrix is not necessarily sparse.
  - fill-in
  - ILU(0)/IC(0) without fill-in have same non-zero pattern with the original (sparse) matrices

LU Factorization/Decomposition: Complete LU Factorization LU分解・完全LU分解

- A kind of direct method for solving linear eqn's
  - compute "inverse matrix" directly
  - Information of "inverse matrix" can be saved, therefore it's efficient for multiple RHS cases
  - "Fill-in" may occur during factorization/decomposition
    - entries which change from an initial zero to a non-zero value during the execution of factorization/decomposition
- LU factorization

Incomplete LU Factorization

- ILU factorization
  - Incomplete LU factorization

- Preconditioning method using “incomplete” inverse matrices, where generation of “fill-in” is controlled
  - Approximate/Incomplete Inverse Matrix, Weak Direct method
  - ILU(0): NO fill-in is allowed

Solving Linear Equations by LU Factorization

LU factorization of matrix A (n × n):

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{pmatrix}$$

$$\mathbf{A} = \mathbf{LU}$$

L : Lower triangular part of matrix A

U : Upper triangular part of matrix A

Matrix Form of Linear Equation

General Form of Linear Equation with “n” unknowns

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$
$$\vdots$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$$

Matdix Form

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \Leftrightarrow \mathbf{Ax} = \mathbf{b}$$

$$\mathbf{A} \qquad \mathbf{x} \qquad \mathbf{b}$$

Solving Ax=b by LU Factorization

1 $\mathbf{A} = \mathbf{LU}$ LU factorization of A

2 $\mathbf{Ly} = \mathbf{b}$ Compute {y} (easy)

3 $\mathbf{Ux} = \mathbf{y}$ Compute {x} (easy)

This {x} satisfies $\mathbf{Ax} = \mathbf{b}$

$$\because \mathbf{Ax} = \mathbf{LUx} = \mathbf{Ly} = \mathbf{b}$$

Forward Substitution：前進代入 Solving Ly=b

$$\mathbf{Ly} = \mathbf{b} \quad \Longleftrightarrow \quad \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

$$\Longleftrightarrow$$

$$\begin{aligned} y_1 &= b_1 \\ l_{21} y_1 + y_2 &= b_2 \\ &\vdots \\ l_{n1} y_1 + l_{n2} y_2 + \cdots + y_n &= b_n \end{aligned} \quad \Longleftrightarrow \quad \begin{aligned} y_1 &= b_1 \\ y_2 &= b_2 - l_{21} y_1 \\ &\vdots \\ y_n &= b_n - l_{n1} y_1 - l_{n2} y_2 = b_n - \sum_{i=1}^{n-1} l_{ni} y_i \end{aligned}$$

row-by-row substitutio

Backward Substitution：後退代入
Solving Ux=y

$$\mathbf{Ux} = \mathbf{y} \quad \Longleftrightarrow \quad \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

$$\Longleftrightarrow$$

$$\begin{aligned} u_{nn} x_n &= y_n \\ u_{n-1,n-1} x_{n-1} + u_{n-1,n} x_n &= y_{n-1} \\ &\vdots \\ u_{11} x_1 + u_{12} x_2 + \cdots + u_{1n} x_n &= y_1 \end{aligned} \quad \Longleftrightarrow \quad \begin{aligned} x_n &= y_n / u_{nn} \\ x_{n-1} &= (y_{n-1} - u_{n-1,n} x_n) / u_{n-1,n-1} \\ &\vdots \\ x_1 &= \left(y_1 - \sum_{i=2}^{n} u_{1j} x_j \right) / u_{11} \end{aligned}$$

row-by-row substitutio

Computation of LU Factorization

①

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{pmatrix}$$

③

② ④

① ➡ $a_{11} = u_{11}, a_{12} = u_{12}, \cdots, a_{1n} = u_{1n} \Rightarrow u_{11}, u_{12}, \cdots, u_{1n}$

② ➡ $a_{21} = l_{21}u_{11}, a_{31} = l_{31}u_{11}, \cdots, a_{n1} = l_{n1}u_{11} \Rightarrow l_{21}, l_{31}, \cdots, l_{n1}$

③ ➡ $a_{22} = l_{21}u_{12} + u_{22}, \cdots, a_{2n} = l_{21}u_{1n} + u_{2n} \Rightarrow u_{22}, u_{23}, \cdots, u_{2n}$

④ ➡ $a_{32} = l_{31}u_{12} + l_{32}u_{22}, \cdots \Rightarrow l_{32}, l_{42}, \cdots, l_{n2}$

Example

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 6 & 7 & 10 \\ 2 & 2 & 8 & 7 \\ 0 & -4 & 7 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix}$$

1st row ➡ $1 = u_{11}, 2 = u_{12}, 3 = u_{13}, 4 = u_{14}$

1st col. ➡ $2 = l_{21}u_{11} \Rightarrow l_{21} = 2/u_{11} = 2, \quad 2 = l_{31}u_{11} \Rightarrow l_{31} = 2/u_{11} = 2$

$0 = l_{41}u_{11} \Rightarrow l_{41} = 0/u_{11} = 0$

2nd row ➡ $6 = l_{21}u_{12} + u_{22} \Rightarrow u_{22} = 2, \quad 7 = l_{21}u_{13} + u_{23} \Rightarrow u_{23} = 1$

$10 = l_{21}u_{14} + u_{24} \Rightarrow u_{24} = 2$

2nd col. ➡ $2 = l_{31}u_{12} + l_{32}u_{22} \Rightarrow l_{32} = -1, \quad -4 = l_{41}u_{12} + l_{42}u_{22} \Rightarrow l_{42} = -2$

Example (cont.)

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 6 & 7 & 10 \\ 2 & 2 & 8 & 7 \\ 0 & -4 & 7 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix}$$

3rd row ➡ $8 = l_{31}u_{13} + l_{32}u_{23} + u_{33} \Rightarrow u_{33} = 3,$

$7 = l_{31}u_{14} + l_{32}u_{24} + u_{34} \Rightarrow u_{34} = 1$

3rd col ➡ $7 = l_{41}u_{13} + l_{42}u_{23} + l_{43}u_{33} \Rightarrow l_{43} = 3$

4th row/col ➡ $1 = l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34} + u_{44} \Rightarrow u_{44} = 2$

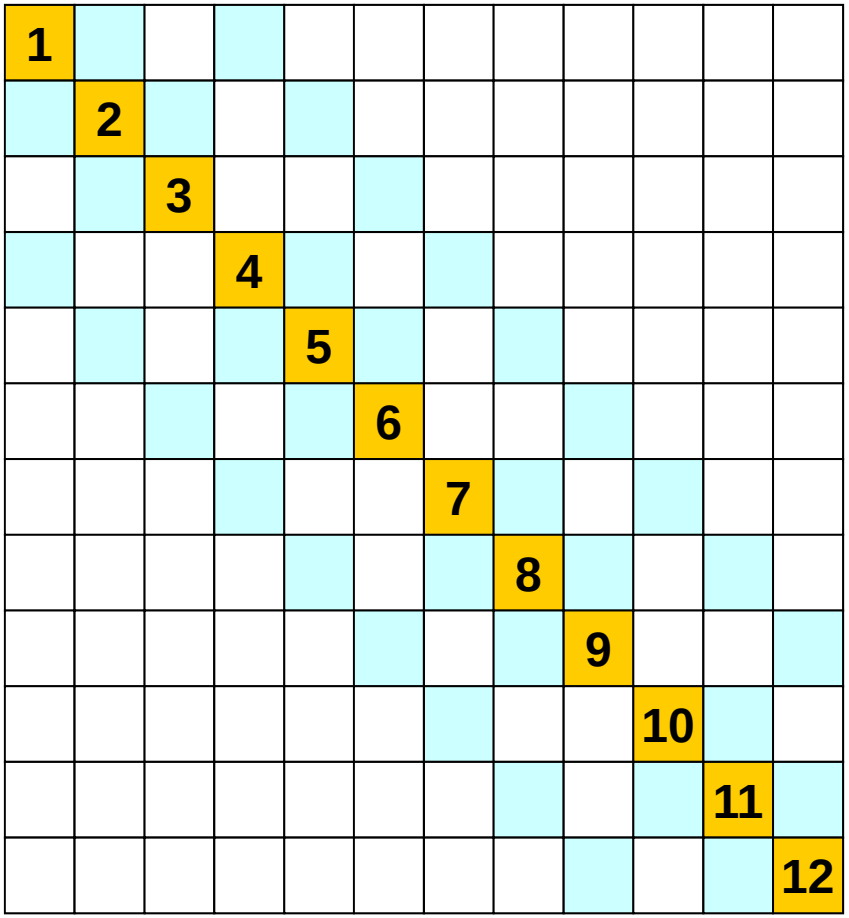
Solving according to 1st row−column, 2nd row−column, 3rd row−column ⋯

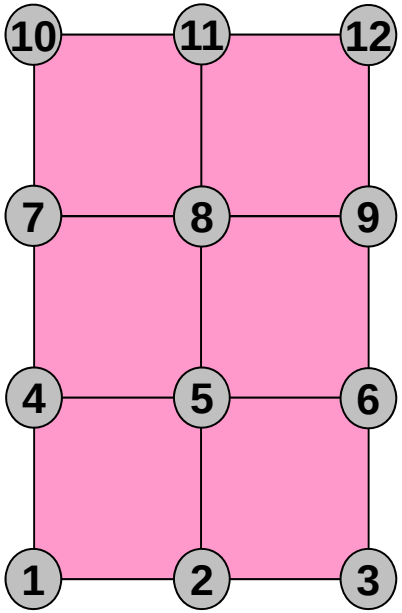
Example (cont.)

Finally:

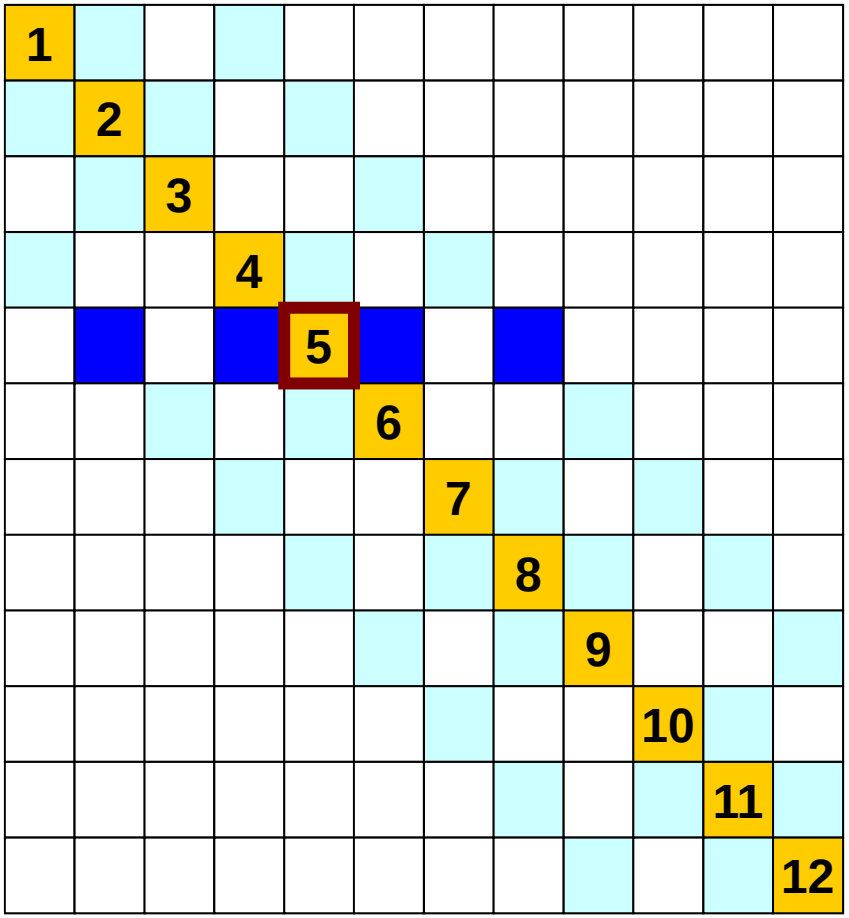
$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 6 & 7 & 10 \\ 2 & 2 & 8 & 7 \\ 0 & -4 & 7 & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 2 & -1 & 1 & 0 \\ 0 & -2 & 3 & 1 \end{pmatrix}}_{\mathbf{L}} \underbrace{\begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 2 & 1 & 2 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 2 \end{pmatrix}}_{\mathbf{U}}$$

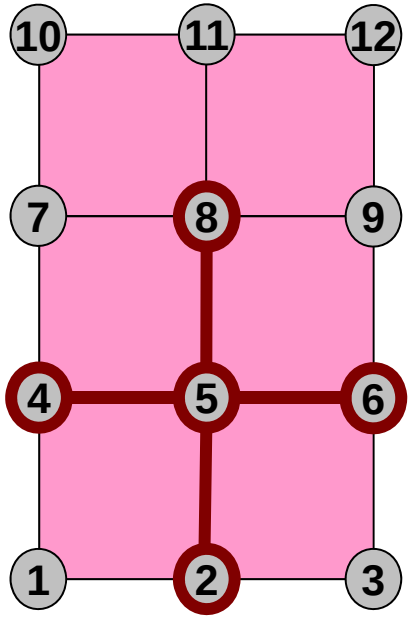
Example: 5-Point Stencil (FDM)
五点差分





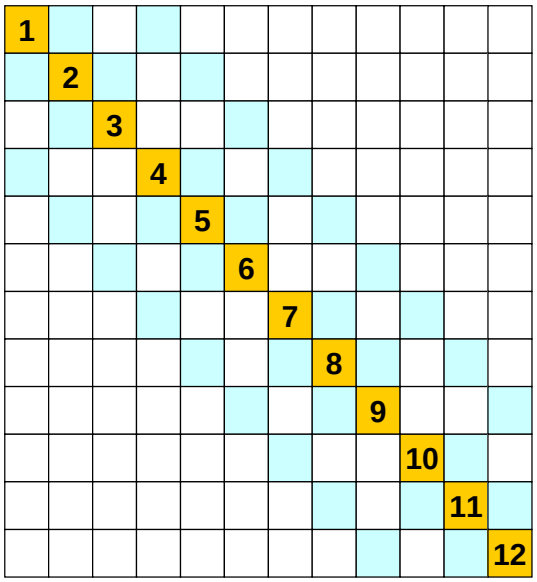
Example: 5-Point Stencil (FDM)

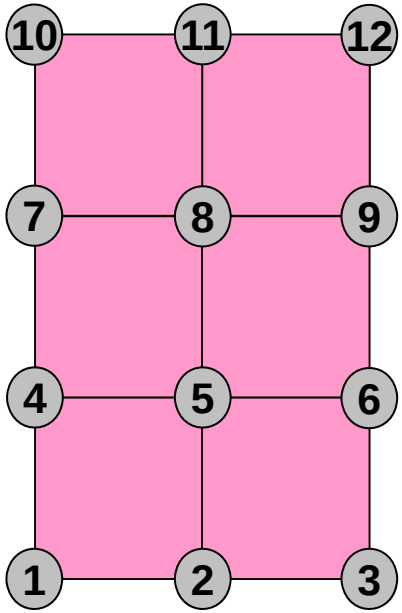




Coef. Matrix: Diag. Component=6.00

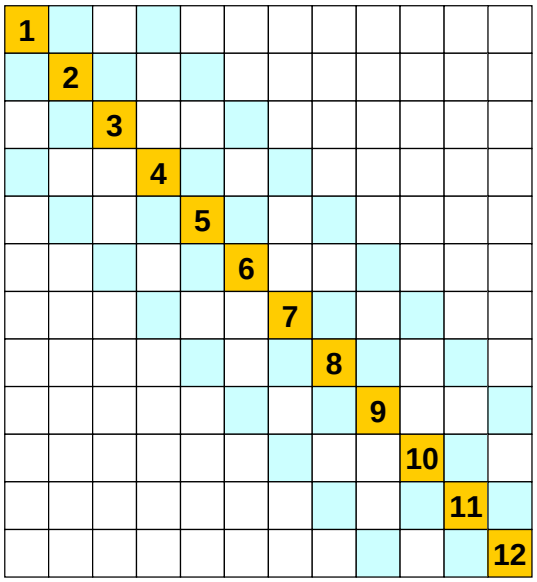
$$
\begin{bmatrix}
6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-1.00 & 6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & -1.00 & 6.00 & 0.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-1.00 & 0.00 & 0.00 & 6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & -1.00 & 0.00 & -1.00 & 6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & -1.00 & 0.00 & -1.00 & 6.00 & 0.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & -1.00 & 0.00 & 0.00 & 6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & -1.00 & 0.00 & -1.00 & 6.00 & -1.00 & 0.00 & -1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -1.00 & 0.00 & -1.00 & 6.00 & 0.00 & 0.00 & -1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -1.00 & 0.00 & 0.00 & 6.00 & -1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -1.00 & 0.00 & -1.00 & 6.00 & -1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -1.00 & 0.00 & -1.00 & 6.00
\end{bmatrix}
\begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}
X
=
\begin{bmatrix}
0.00 \\ 3.00 \\ 10.00 \\ 11.00 \\ 10.00 \\ 19.00 \\ 20.00 \\ 16.00 \\ 28.00 \\ 42.00 \\ 36.00 \\ 52.00
\end{bmatrix}
$$

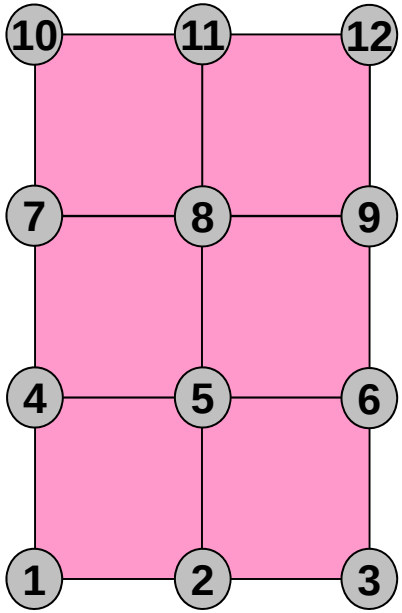




Solution

$$
\begin{bmatrix}
6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-1.00 & 6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & -1.00 & 6.00 & 0.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-1.00 & 0.00 & 0.00 & 6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & -1.00 & 0.00 & -1.00 & 6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & -1.00 & 0.00 & -1.00 & 6.00 & 0.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & -1.00 & 0.00 & 0.00 & 6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & -1.00 & 0.00 & -1.00 & 6.00 & -1.00 & 0.00 & -1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -1.00 & 0.00 & -1.00 & 6.00 & 0.00 & 0.00 & -1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -1.00 & 0.00 & 0.00 & 6.00 & -1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -1.00 & 0.00 & -1.00 & 6.00 & -1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -1.00 & 0.00 & -1.00 & 6.00
\end{bmatrix}
\begin{bmatrix}
1.00 \\ 2.00 \\ 3.00 \\ 4.00 \\ 5.00 \\ 6.00 \\ 7.00 \\ 8.00 \\ 9.00 \\ 10.00 \\ 11.00 \\ 12.00
\end{bmatrix}
=
\begin{bmatrix}
0.00 \\ 3.00 \\ 10.00 \\ 11.00 \\ 10.00 \\ 19.00 \\ 20.00 \\ 16.00 \\ 28.00 \\ 42.00 \\ 36.00 \\ 52.00
\end{bmatrix}
$$





Installation of files

```
>$ cd <$fem1>
>$ cp /home03/skengon/Documents/class/fem1/lu.tar .

>$ tar xvf lu.tar
>$ cd lu

>$ g95 lu1.f –o lu1
>$ g95 lu2.f –o lu2
>$ g95 lu3.f –o lu3
```

Complete LU Factorization

type “./lu1”

Original Matrix

6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
−1.00	6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	−1.00	6.00	0.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00
−1.00	0.00	0.00	6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00
0.00	−1.00	0.00	−1.00	6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00
0.00	0.00	−1.00	0.00	−1.00	6.00	0.00	0.00	−1.00	0.00	0.00	0.00
0.00	0.00	0.00	−1.00	0.00	0.00	6.00	−1.00	0.00	−1.00	0.00	0.00
0.00	0.00	0.00	0.00	−1.00	0.00	−1.00	6.00	−1.00	0.00	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	−1.00	0.00	−1.00	6.00	0.00	0.00	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	−1.00	0.00	0.00	6.00	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	−1.00	0.00	−1.00	6.00	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−1.00	0.00	−1.00	6.00

LU Factorization

Both of [L] and [U] are shown
Diag. of [L] are “1” (not shown)

fill-in occurs: some of zero components became non-zero.

6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	5.83	−1.00	−0.17	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	5.83	−0.03	−0.17	−1.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	−0.03	0.00	5.83	−1.03	0.00	−1.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	−0.03	−0.18	5.64	−1.03	−0.18	−1.00	0.00	0.00	0.00	0.00
0.00	0.00	−0.17	0.00	−0.18	5.64	−0.03	−0.18	−1.00	0.00	0.00	0.00
0.00	0.00	0.00	−0.17	−0.03	−0.01	5.82	−1.03	−0.01	−1.00	0.00	0.00
0.00	0.00	0.00	0.00	−0.18	−0.03	−0.18	5.63	−1.03	−0.18	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.63	−0.03	−0.18	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	−0.17	−0.03	−0.01	5.82	−1.03	−0.01
0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	−0.03	−0.18	5.63	−1.03
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.63

Incomp. LU fact. with no fill-in's

type "./lu2"

Incomplete LU Factorization without fill-in's

Both of [L] and [U] are shown
Diag. of [L] are "1" (not shown)

6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	5.83	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	5.83	0.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	0.00	0.00	5.83	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	0.00	−0.17	5.66	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00
0.00	0.00	−0.17	0.00	−0.18	5.65	0.00	0.00	−1.00	0.00	0.00	0.00
0.00	0.00	0.00	−0.17	0.00	0.00	5.83	−1.00	0.00	−1.00	0.00	0.00
0.00	0.00	0.00	0.00	−0.18	0.00	−0.17	5.65	−1.00	0.00	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.65	0.00	0.00	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	−0.17	0.00	0.00	5.83	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.17	5.65	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.65

LU Factorization

Both of [L] and [U] are shown
Diag. of [L] are "1" (not shown)

fill-in occurs: some of zero components became non-zero.

6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	5.83	−1.00	−0.17	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	5.83	−0.03	−0.17	−1.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	−0.03	0.00	5.83	−1.03	0.00	−1.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	−0.03	−0.18	5.64	−1.03	−0.18	−1.00	0.00	0.00	0.00	0.00
0.00	0.00	−0.17	0.00	−0.18	5.64	−0.03	−0.18	−1.00	0.00	0.00	0.00
0.00	0.00	0.00	−0.17	−0.03	−0.01	5.82	−1.03	−0.01	−1.00	0.00	0.00
0.00	0.00	0.00	0.00	−0.18	−0.03	−0.18	5.63	−1.03	−0.18	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.63	−0.03	−0.18	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	−0.17	−0.03	−0.01	5.82	−1.03	−0.01
0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	−0.03	−0.18	5.63	−1.03
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.63

Slightly “Inaccurate” Solution

Incomplete LU

$$
\begin{pmatrix}
6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-0.17 & 5.83 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & -0.17 & 5.83 & 0.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-0.17 & 0.00 & 0.00 & 5.83 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & -0.17 & 0.00 & -0.17 & 5.66 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & -0.17 & 0.00 & -0.18 & 5.65 & 0.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & -0.17 & 0.00 & 0.00 & 5.83 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & -0.18 & 0.00 & -0.17 & 5.65 & -1.00 & 0.00 & -1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.18 & 0.00 & -0.18 & 5.65 & 0.00 & 0.00 & -1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.17 & 0.00 & 0.00 & 5.83 & -1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.18 & 0.00 & -0.17 & 5.65 & -1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.18 & 0.00 & -0.18 & 5.65
\end{pmatrix}
\rightarrow
\begin{pmatrix}
0.92 \\ 1.75 \\ 2.76 \\ 3.79 \\ 4.46 \\ 5.57 \\ 6.66 \\ 7.25 \\ 8.46 \\ 9.66 \\ 10.54 \\ 11.83
\end{pmatrix}
$$

Complete LU

$$
\begin{pmatrix}
6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-0.17 & 5.83 & -1.00 & -0.17 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & -0.17 & 5.83 & -0.03 & -0.17 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-0.17 & -0.03 & 0.00 & 5.83 & -1.03 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & -0.17 & -0.03 & -0.18 & 5.64 & -1.03 & -0.18 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & -0.17 & 0.00 & -0.18 & 5.64 & -0.03 & -0.18 & -1.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & -0.17 & -0.03 & -0.01 & 5.82 & -1.03 & -0.01 & -1.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & -0.18 & -0.03 & -0.18 & 5.63 & -1.03 & -0.18 & -1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.18 & 0.00 & -0.18 & 5.63 & -0.03 & -0.18 & -1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.17 & -0.03 & -0.01 & 5.82 & -1.03 & -0.01 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.18 & -0.03 & -0.18 & 5.63 & -1.03 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.18 & 0.00 & -0.18 & 5.63
\end{pmatrix}
\rightarrow
\begin{pmatrix}
1.00 \\ 2.00 \\ 3.00 \\ 4.00 \\ 5.00 \\ 6.00 \\ 7.00 \\ 8.00 \\ 9.00 \\ 10.00 \\ 11.00 \\ 12.00
\end{pmatrix}
$$

ILU(0), IC(0)

- “Incomplete” factorization without fill-in’s
  - Reduced memory, computation

- Solving equations by ILU(0)/IC(0) factorization provides slightly “inaccurate” solution, although it’s not far from exact one.
  - “Accurateness” depends on problems (feature of equations).

Full LU and ILU(0)/IC(0)

Full LU

```
do i= 2, n
  do k= 1, i-1
    a_ik := a_ik/a_kk
    do j= k+1, n
      a_ij := a_ij - a_ik*a_kj
    enddo
  enddo
enddo
```

ILU(0) : keep non-zero pattern of the original coefficient matrix

```
do i= 2, n
  do k= 1, i-1
    if ((i,k)∈ NonZero(A)) then
      a_ik := a_ik/a_kk
    endif
    do j= k+1, n
      if ((i,j)∈ NonZero(A)) then
        a_ij := a_ij - a_ik*a_kj
      endif
    enddo
  enddo
enddo
```

Deep Fill-in: ILU(p)/IC(p)

p: level of fill-in. If “p” increases, ILU(p)/IC(p) become closer to complete ILU/IC and provide more robust preconditioners, but become more expensive: <u>trade-off</u>

```
LEV_ij=0 if ((i,j)∈ NonZero(A)) otherwise LEV_ij= p+1

do i= 2, n
  do k= 1,  i-1
    if (LEV_ik≦p) then
      a_ik := a_ik/a_kk
    endif
    do j= k+1, n
      if (LEV_ij = min(LEV_ij, 1+LEV_ik+ LEV_kj)≦p) then
        a_ij := a_ij - a_ik*a_kj
      endif
    enddo
  enddo
enddo
```

LU Gauss-Seidel (LU-GS)
LU Symmetric GS (LU-SGS)
in this class

- ILU(0)

```
do i= 2, n
    do k= 1,  i-1
      if ((i,k)∈ NonZero(A)) then
        a_ik := a_ik/a_kk
      endif
      do j= k+1, n
        if ((i,j)∈ NonZero(A)) then
          a_ij := a_ij - a_ik*a_kj
        endif
      enddo
    enddo
  enddo
```

LU Gauss-Seidel (LU-GS)
LU Symmetric GS (LU-SGS)
in this class

- More Simplified Version of ILU(0)

```
do i= 2, n
    do k= 1,  i-1
      if ((i,k)∈ NonZero(A)) then
        a_ik := a_ik/a_kk
      endif
      do j= k+1, n
        if ((i,j)∈ NonZero(A)) then
          a_ij := a_ij - a_ik*a_kj
        endif
      enddo
    enddo
  enddo
```

Only do this

LU Gauss-Seidel (LU-GS)
LU Symmetric GS (LU-SGS)
in this class

- More Simplified Version of ILU(0)

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ a_{21}/a_{22} & 1 & 0 & \cdots & 0 \\ a_{31}/a_{33} & a_{32}/a_{33} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1}/a_{nn} & a_{n2}/a_{nn} & a_{n3}/a_{nn} & \cdots & 1 \end{pmatrix}$$

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n} \\ 0 & 0 & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{pmatrix}$$

```
do i= 2, n
    do k= 1, i-1
      if ((i,k)∈ NonZero(A)) then
        a_ik := a_ik/a_kk
      endif
      do j= k+1, n
        if ((i,j)∈ NonZero(A)) then
          a_ij := a_ij - a_ik*a_kj
        endif
      enddo
    enddo
  enddo
```

ILU, LU-GS

type “./lu3”

Incomplete LU Factorization without Fill-in’s

6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	5.83	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	5.83	0.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	0.00	0.00	5.83	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	0.00	−0.17	5.66	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00
0.00	0.00	−0.17	0.00	−0.18	5.65	0.00	0.00	−1.00	0.00	0.00	0.00
0.00	0.00	0.00	−0.17	0.00	0.00	5.83	−1.00	0.00	−1.00	0.00	0.00
0.00	0.00	0.00	0.00	−0.18	0.00	−0.17	5.65	−1.00	0.00	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.65	0.00	0.00	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	−0.17	0.00	0.00	5.83	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.17	5.65	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.65

LU-GS without Fill-in’s

6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	6.00	0.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	0.00	0.00	6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	0.00	−0.17	6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00
0.00	0.00	−0.17	0.00	−0.17	6.00	0.00	0.00	−1.00	0.00	0.00	0.00
0.00	0.00	0.00	−0.17	0.00	0.00	6.00	−1.00	0.00	−1.00	0.00	0.00
0.00	0.00	0.00	0.00	−0.17	0.00	−0.17	6.00	−1.00	0.00	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	−0.17	0.00	−0.17	6.00	0.00	0.00	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	−0.17	0.00	0.00	6.00	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.17	0.00	−0.17	6.00	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.17	0.00	−0.17	6.00

Solution is more “inaccurate”

ILU(0)

$$
\begin{pmatrix}
6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-0.17 & 5.83 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & -0.17 & 5.83 & 0.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-0.17 & 0.00 & 0.00 & 5.83 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & -0.17 & 0.00 & -0.17 & 5.66 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & -0.17 & 0.00 & -0.18 & 5.65 & 0.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & -0.17 & 0.00 & 0.00 & 5.83 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & -0.18 & 0.00 & -0.17 & 5.65 & -1.00 & 0.00 & -1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.18 & 0.00 & -0.18 & 5.65 & 0.00 & 0.00 & -1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.17 & 0.00 & 0.00 & 5.83 & -1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.18 & 0.00 & -0.17 & 5.65 & -1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.18 & 0.00 & -0.18 & 5.65
\end{pmatrix}
\rightarrow
\begin{pmatrix}
0.92 \\ 1.75 \\ 2.76 \\ 3.79 \\ 4.46 \\ 5.57 \\ 6.66 \\ 7.25 \\ 8.46 \\ 9.66 \\ 10.54 \\ 11.83
\end{pmatrix}
$$

LU-GS

$$
\begin{pmatrix}
6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-0.17 & 6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & -0.17 & 6.00 & 0.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-0.17 & 0.00 & 0.00 & 6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & -0.17 & 0.00 & -0.17 & 6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & -0.17 & 0.00 & -0.17 & 6.00 & 0.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & -0.17 & 0.00 & 0.00 & 6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & -0.17 & 0.00 & -0.17 & 6.00 & -1.00 & 0.00 & -1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.17 & 0.00 & -0.17 & 6.00 & 0.00 & 0.00 & -1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.17 & 0.00 & 0.00 & 6.00 & -1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.17 & 0.00 & -0.17 & 6.00 & -1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.17 & 0.00 & -0.17 & 6.00
\end{pmatrix}
\rightarrow
\begin{pmatrix}
0.86 \\ 1.60 \\ 2.60 \\ 3.54 \\ 3.99 \\ 5.09 \\ 6.26 \\ 6.52 \\ 7.73 \\ 9.22 \\ 9.70 \\ 10.96
\end{pmatrix}
$$

Forward/Backward Substitution in LU-GS

$$[M]\{z\} = [\tilde{L}\tilde{U}]\{z\} = \{r\}$$

$$\{z\} = [\tilde{L}\tilde{U}]^{-1}\{r\} \longrightarrow \boxed{\begin{aligned} [\tilde{L}]\{y\} &= \{r\} \\ [\tilde{U}]\{z\} &= \{y\} \end{aligned}}$$

$$[\tilde{L}] = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ a_{21}/a_{22} & 1 & 0 & \cdots & 0 \\ a_{31}/a_{33} & a_{32}/a_{33} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1}/a_{nn} & a_{n2}/a_{nn} & a_{n3}/a_{nn} & \cdots & 1 \end{pmatrix}$$

$$[\tilde{U}] = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n} \\ 0 & 0 & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{pmatrix}$$

$$\left[\overline{L}\right]=\begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ a_{21} & 0 & 0 & \cdots & 0 \\ a_{31} & a_{32} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & 0 \end{pmatrix} \qquad \left[\overline{U}\right]=\begin{pmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & 0 & a_{23} & \cdots & a_{2n} \\ 0 & 0 & 0 & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

$$\left[\overline{D}\right]=\begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{pmatrix}$$

$$
\begin{aligned}
[M] &= [\tilde{L}][\tilde{U}] = [\overline{L}+\overline{D}][\overline{D}^{-1}][\overline{D}+\overline{U}] = [\overline{L}\,\overline{D}^{-1}+I][\overline{D}+\overline{U}] \\
&= [\overline{L}+\overline{D}][I+\overline{D}^{-1}\overline{U}]
\end{aligned}
$$

$$
[\overline{L}]+[\overline{D}] = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ a_{21} & 0 & 0 & \cdots & 0 \\ a_{31} & a_{32} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & 0 \end{pmatrix} + \begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ a_{21} & a_{22} & 0 & \cdots & 0 \\ a_{31} & a_{32} & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} = [\tilde{L}]
$$

$$
[I]+[\overline{D}^{-1}\overline{U}] = \begin{pmatrix} 1 & a_{12}/a_{11} & a_{13}/a_{11} & \cdots & a_{1n}/a_{11} \\ 0 & 1 & a_{23}/a_{22} & \cdots & a_{2n}/a_{22} \\ 0 & 0 & 1 & \cdots & a_{3n}/a_{33} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} = [\tilde{U}]
$$

Forward/Backward Subst. in LU-GS

$$[M]=[\tilde{L}][\tilde{U}]=[\overline{L}+\overline{D}][\overline{D}^{-1}][\overline{D}+\overline{U}]=[\overline{L}\,\overline{D}^{-1}+I][\overline{D}+\overline{U}]=[\overline{L}+\overline{D}][I+\overline{D}^{-1}\overline{U}]$$

Forward Substitution

$$[\overline{L}+\overline{D}]\{y\}=\{r\}\Rightarrow\{y\}=[\overline{D}^{-1}](\{r\}-[\overline{L}]\{y\})\Rightarrow y_i=\overline{D}_{ii}^{-1}\left(r_i-\sum_{j=1}^{i-1}\overline{L}_{ij}y_j\right)$$

Backward Substitution

$$[I+\overline{D}^{-1}\overline{U}]\{z\}=\{y\}\Rightarrow\{z\}=\{y\}-[\overline{D}^{-1}][\overline{U}]\{z\}\Rightarrow z_i=y_i-\overline{D}_{ii}^{-1}\left[\sum_{j=i+1}^{N}\overline{U}_{ij}z_j\right]$$

$$[\overline{L}]+[\overline{D}]=\begin{pmatrix}0&0&0&\cdots&0\\a_{21}&0&0&\cdots&0\\a_{31}&a_{32}&0&\cdots&0\\\vdots&\vdots&\vdots&\ddots&\vdots\\a_{n1}&a_{n2}&a_{n3}&\cdots&0\end{pmatrix}+\begin{pmatrix}a_{11}&0&0&\cdots&0\\0&a_{22}&0&\cdots&0\\0&0&a_{33}&\cdots&0\\\vdots&\vdots&\vdots&\ddots&\vdots\\0&0&0&\cdots&a_{nn}\end{pmatrix}=\begin{pmatrix}a_{11}&0&0&\cdots&0\\a_{21}&a_{22}&0&\cdots&0\\a_{31}&a_{32}&a_{33}&\cdots&0\\\vdots&\vdots&\vdots&\ddots&\vdots\\a_{n1}&a_{n2}&a_{n3}&\cdots&a_{nn}\end{pmatrix}=[\tilde{L}]$$

$$[I]+[\overline{D}^{-1}\overline{U}]=\begin{pmatrix}1&a_{12}/a_{11}&a_{13}/a_{11}&\cdots&a_{1n}/a_{11}\\0&1&a_{23}/a_{22}&\cdots&a_{2n}/a_{22}\\0&0&1&\cdots&a_{3n}/a_{33}\\\vdots&\vdots&\vdots&\ddots&\vdots\\0&0&0&\cdots&1\end{pmatrix}=[\tilde{U}]$$

Forward/Backward Subst. in LU-GS

$$\left[\tilde{L}\right]\{y\}=\{r\}$$

$$\left[\tilde{U}\right]\{z\}=\{y\}$$

```
!C
!C +----------------+
!C | {z}= [Minv]{r} |
!C +----------------+
!C===



      do i= 1, N
        WVAL= W(i,R)
        do k= indexL(i-1)+1, indexL(i)
          WVAL= WVAL -  AL(k) * W(itemL(k),Y)
        enddo
        W(i,Y)= WVAL / D(i)
      enddo

      do i= N, 1, -1
        SW  = 0.0d0
        do k= indexU(i), indexU(i-1)+1, -1
          SW= SW + AU(k) * W(itemU(k),Z)
        enddo
        W(i,Z)= W(i,Y) - SW / D(i)
      enddo
!C===
```

Forward Substitution

Computation of lower-triangular components have been completed.

$$y_i = \overline{D}_{ii}^{-1}\left(r_i - \sum_{j=1}^{i-1}\overline{L}_{ij}y_j\right)$$

Backward Substitution

Computation of upper-triangular components have been completed:

$$z_i = y_i - \overline{D}_{ii}^{-1}\left[\sum_{j=i+1}^{n}\overline{U}_{ij}z_j\right]$$

Forward/Backward Subst. in LU-GS

$$\left[\tilde{L}\right]\{y\}=\{r\}$$

$$\left[\tilde{U}\right]\{z\}=\{y\}$$

```
!C
!C +----------------+
!C | {z}= [Minv]{r} |
!C +----------------+
!C===



      do i= 1, N
        WVAL= W(i,R)
        do k= indexL(i-1)+1, indexL(i)
          WVAL= WVAL -  AL(k) * W(itemL(k),Y)
        enddo
        W(i,Y)= WVAL / D(i)
      enddo

      do i= N, 1, -1
        SW  = 0.0d0
        do k= indexU(i), indexU(i-1)+1, -1
          SW= SW + AU(k) * W(itemU(k),Z)
        enddo
        W(i,Z)= W(i,Y) - SW / D(i)
      enddo
!C===
```

Forward Substitution

If i=1, there are no lower-triangular components.

$$y_1 = \overline{D}_{11}^{-1} r_1$$

Upper Substitution

If i=n, there are no upper-triangular components.

$$z_n = y_n$$

Forward/Backward Subst. in LU-GS

$$\left[\tilde{L}\right]\{z\}=\{r\}$$

$$\left[\tilde{U}\right]\{z\}=\{z\}$$

```
!C
!C +----------------+
!C | {z}= [Minv]{r} |
!C +----------------+
!C===



      do i= 1, N
        WVAL= W(i,R)
        do k= indexL(i-1)+1, indexL(i)
          WVAL= WVAL -  AL(k) * W(itemL(k),Z)
        enddo
        W(i,Z)= WVAL / D(i)
      enddo

      do i= N, 1, -1
        SW  = 0.0d0
        do k= indexU(i), indexU(i-1)+1, -1
          SW= SW + AU(k) * W(itemU(k),Z)
        enddo
        W(i,Z)= W(i,Z) - SW / D(i)
      enddo
!C===
```

Separated arrays ({y} and {z}) are not needed.

Forward Substitution

Computation of lower-triangular components have been completed.

$$z_i = \overline{D}_{ii}^{-1}\left(r_i - \sum_{j=1}^{i-1}\overline{L}_{ij} z_j\right)$$

Backward Substitution

Computation of upper-triangular components have been completed:

$$z_i = z_i - \overline{D}_{ii}^{-1}\left[\sum_{j=i+1}^{n}\overline{U}_{ij} z_j\right]$$

Forward/Backward Subst. in LU-GS

$$\left[\tilde{L}\right]\{z\}=\{z\}$$

$$\left[\tilde{U}\right]\{z\}=\{z\}$$

```
!C
!C +----------------+
!C | {z}= [Minv]{r} |
!C +----------------+
!C===
      do i= 1, N
        W(i,Z)= W(i,R)
      enddo

      do i= 1, N
        WVAL= W(i,Z)
        do k= indexL(i-1)+1, indexL(i)
          WVAL= WVAL -  AL(k) * W(itemL(k),Z)
        enddo
        W(i,Z)= WVAL / D(i)
      enddo

      do i= N, 1, -1
        SW  = 0.0d0
        do k= indexU(i), indexU(i-1)+1, -1
          SW= SW + AU(k) * W(itemU(k),Z)
        enddo
        W(i,Z)= W(i,Z) - SW / D(i)
      enddo
!C===
```

Separated arrays ({r}, {y} and {z}) are not needed.

$$z_1 = \overline{D}_{11}^{-1} r_1$$

Forward Substitution

Computation of lower-triangular components have been completed.

$$z_i = \overline{D}_{ii}^{-1}\left(z_i - \sum_{j=1}^{i-1} \overline{L}_{ij} z_j \right)$$

Backward Substitution

Computation of upper-triangular components have been completed:

$$z_i = z_i - \overline{D}_{ii}^{-1}\left[\sum_{j=i+1}^{n} \overline{U}_{ij} z_j \right]$$

Forward/Backward Subst. in LU-GS

$$\left[\tilde{L}\right]\{z\}=\{z\}$$

$$\left[\tilde{U}\right]\{z\}=\{z\}$$

```
!C
!C +----------------+
!C | {z}= [Minv]{r} |
!C +----------------+
!C===
      do i= 1, N
        W(i,Z)= W(i,R)
      enddo

      do i= 1, N
        WVAL= W(i,Z)
        do k= indexL(i-1)+1, indexL(i)
          WVAL= WVAL -  AL(k) * W(itemL(k),Z)
        enddo
        W(i,Z)= WVAL / D(i)
      enddo

      do i= N, 1, -1
        SW  = 0.0d0
        do k= indexU(i), indexU(i-1)+1, -1
          SW= SW + AU(k) * W(itemU(k),Z)
        enddo
        W(i,Z)= W(i,Z) - SW / D(i)
      enddo
!C===
```

$$z_i = \overline{D}_{ii}^{-1}\left(z_i - \sum_{j=1}^{i-1}\overline{L}_{ij}z_j\right)$$

$$\mathrm{WVAL} = z_i - \sum_{j=1}^{i-1}\overline{L}_{ij}z_j$$

$$z_i = z_i - \overline{D}_{ii}^{-1}\left[\sum_{j=i+1}^{N}\overline{U}_{ij}z_j\right]$$

$$\mathrm{SW} = \sum_{j=i+1}^{N}\overline{U}_{ij}z_j$$

Forward/Backward Subst. in LU-GS

$$\left[\tilde{L}\right]\{z\}=\{z\}$$

$$\left[\tilde{U}\right]\{z\}=\{z\}$$

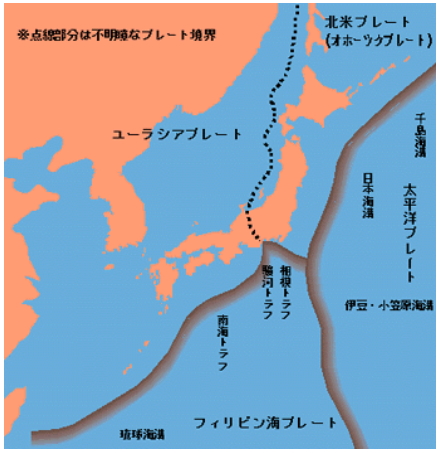
```
!C
!C +----------------+
!C | {z}= [Minv]{r} |
!C +----------------+
!C===
      do i= 1, N
        W(i,Z)= W(i,R)
      enddo

      do i= 1, N
        WVAL= W(i,Z)
        do k= indexL(i-1)+1, indexL(i)
          WVAL= WVAL -  AL(k) * W(itemL(k),Z)
        enddo
        W(i,Z)= WVAL / D(i)
      enddo

      do i= N, 1, -1
        SW  = 0.0d0
        do k= indexU(i), indexU(i-1)+1, -1
          SW= SW + AU(k) * W(itemU(k),Z)
        enddo
        W(i,Z)= W(i,Z) - SW / D(i)
      enddo
!C===
```

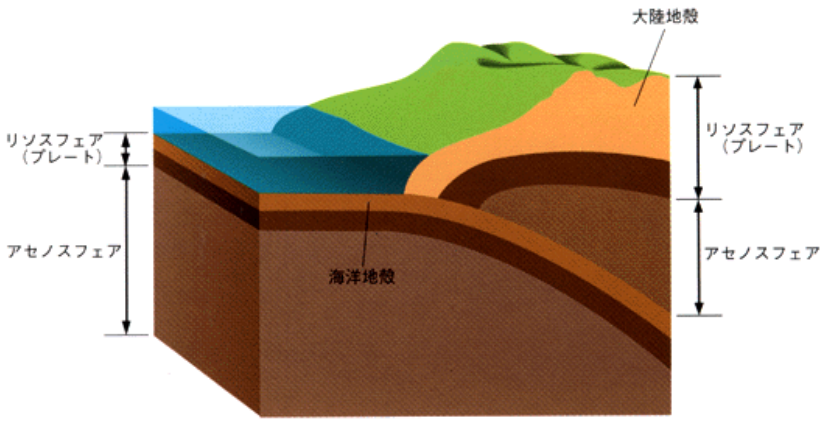
Full LU factorization of 3x3 diagonal block.

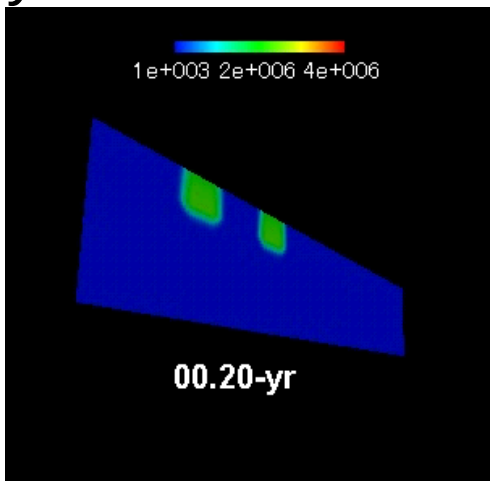
Forward/backward substitution using full LU factorization in stead of diagonal scaling for 3x3 block

Preconditioning Method for Contact Problems

- Contact Problems for Simulations of Earthquake Simulation Cycle
  - Quasi-Static Stress Accumulation Process at Plate Boundaries
  - Non-Linear Contact Problems, Newton-Raphson Method
  - Constraint Conditions through Augmented Lagrangean Method (ALM：拡大ラグランジェ法): Penalty Terms



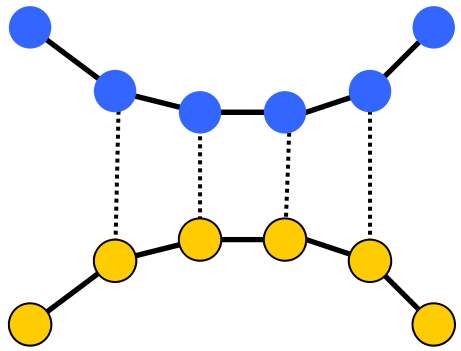




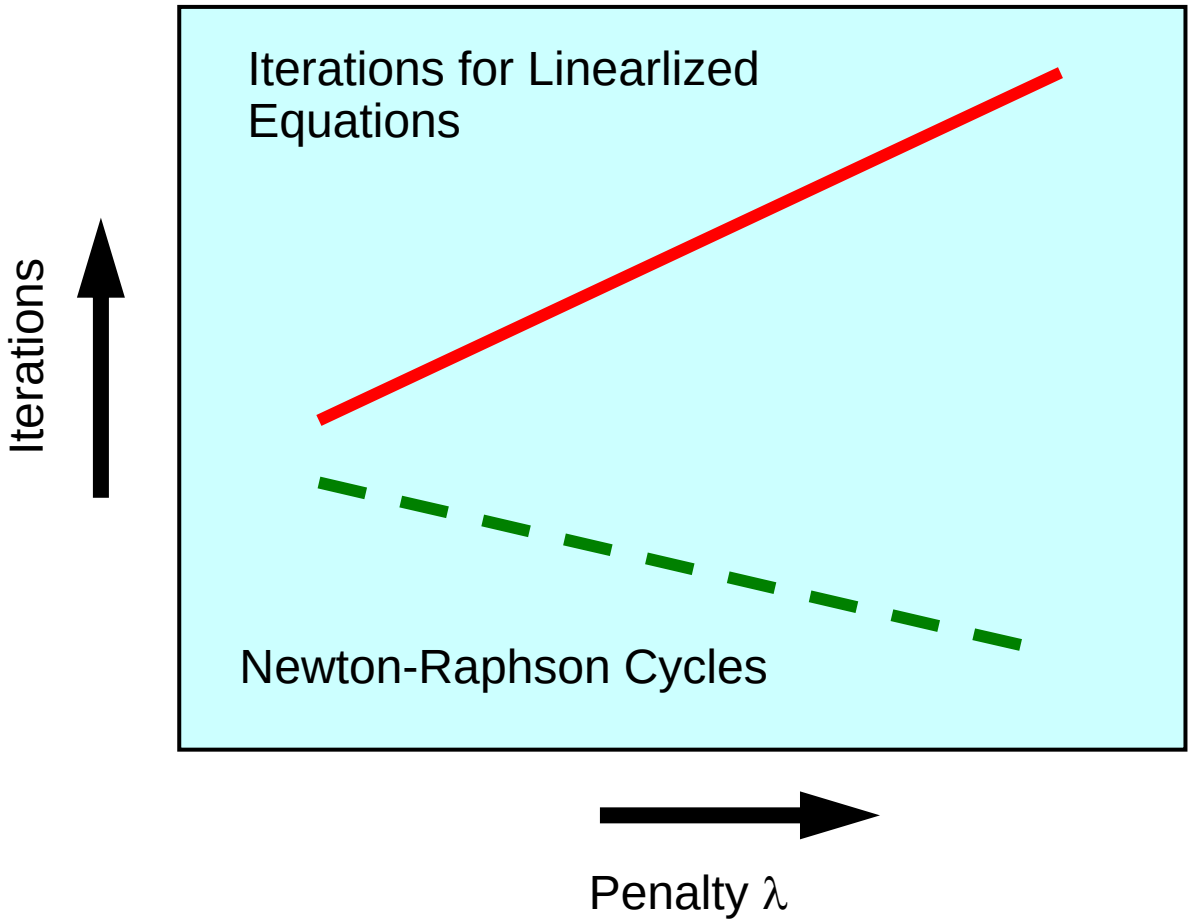
Preconditioning Method for Contact Problems (cont.)

- Assumption
  - Infinitesimal Deformation Theory, Static Contact Condition (contact conditions not changed)
  - No friction: symmetric coefficient matrices
- Special preconditioning method: ***Selective Blocking.***
  - Suitable for 3D contact problems

- Computations
  - Hitachi SR2201: 2001-2002
  - Earth Simulator: 2002-2003
  - IBM SP-3: 2003-2005



Augmented Lagrangean
拡大ラグランジェ法
Newton-Raphson / Iterative Solver



If penalty number becomes larger, number of Newton-Raphson cycles is smaller because of higher accuracy for contact conditions.

But, linearlized equations are worse-conditioned.

Preliminary Results

Elastic Problems with Penalty Constraint

27,888 nodes, 83,664 DOFs, $\varepsilon=10^{-8}$

Single PE case (Xeon 2.8MHz)

GeoFEM's Original Solvers (Scalar Version)

Preconditioning	λ	Iterations	Set-up (sec.)	Solve (sec.)	Set-up+Solve (sec.)	Single Iteration (sec.)	Memory Size (MB)
Diagonal Scaling	10^2	1531	<0.01	75.1	75.1	0.049	119
	10^6	No Conv.	-	-	-	-	
IC(0) (Scalar Type)	10^2	401	0.02	39.2	39.2	0.098	119
	10^6	No Conv.	-	-	-	-	
BIC(0)	10^2	388	0.02	37.4	37.4	0.097	59
	10^6	2590	0.01	252.3	252.3	0.097	
BIC(1)	10^2	77	8.5	11.7	20.2	0.152	176
	10^6	78	8.5	11.8	20.3	0.152	
BIC(2)	10^2	59	16.9	13.9	30.8	0.236	319
	10^6	59	16.9	13.9	30.8	0.236	
SB-BIC(0)	10^0	114	0.10	12.9	13.0	0.113	67
	10^6	114	0.10	12.9	13.0	0.113	

Ill-Conditioned Problems
悪条件問題

- Generally, direct methods have been used for ill-conditioned linear equations.
- But, it is difficult to “parallelize” direct method for large-scale problems
- Robust preconditioning is required

- Remedies
  - Similar to Direct Method with Higher Order of Fill-in’s
  - Blocking
  - Reordering

Higher Order of Fill-in’s

- Closer to Direct Method
- More Expensive (Memory, Computation)

Blocking in F/B Substitution

$$[M] = [\tilde{L}][\tilde{U}] = [\overline{L} + \overline{D}][\overline{D}^{-1}][\overline{D} + \overline{U}] = [\overline{L}\,\overline{D}^{-1} + I][\overline{D} + \overline{U}]$$

Forward Substitution

$$[\overline{L} + \overline{D}]\{y\} = \{r\} \Rightarrow \{y\} = [\overline{D}^{-1}](\{r\} - [\overline{L}]\{y\}) \Rightarrow y_i = \overline{D}_{ii}^{-1}\left(r_i - \sum_{j=1}^{i-1} \overline{L}_{ij} y_j\right)$$

Backward Substitution

$$[I + \overline{D}^{-1}\overline{U}]\{z\} = \{y\} \Rightarrow \{z\} = \{y\} - [\overline{D}^{-1}][\overline{U}]\{z\} \Rightarrow z_i = y_i - \overline{D}_{ii}^{-1}\left[\sum_{j=i+1}^{N} \overline{U}_{ij} z_j\right]$$

- Full LU factorization of 3x3 diagonal block in stead of diagonal scaling for the process of “multiplying D^{-1}”.
  - 3D solid mechanics
  - 3 strongly coupled components on each node
  - Smaller indirect access, more efficient

Results in the Benchmark

27,888 nodes, 83,664 DOFs, $\varepsilon=10^{-8}$

Single PE case (Xeon 2.8MHz)

Effect of Blocking/Fill-in

Preconditioning	λ	Iterations	Set-up (sec.)	Solve (sec.)	Set-up+Solve (sec.)	Single Iteration (sec.)	Memory Size (MB)
Diagonal Scaling	10^2	1531	<0.01	75.1	75.1	0.049	119
	10^6	No Conv.	-	-	-	-	
IC(0) (Scalar Type)	10^2	401	0.02	39.2	39.2	0.098	119
	10^6	No Conv.	-	-	-	-	
BIC(0)	10^2	388	0.02	37.4	37.4	0.097	59
	10^6	2590	0.01	252.3	252.3	0.097	
BIC(1)	10^2	77	8.5	11.7	20.2	0.152	176
	10^6	78	8.5	11.8	20.3	0.152	
BIC(2)	10^2	59	16.9	13.9	30.8	0.236	319
	10^6	59	16.9	13.9	30.8	0.236	
SB-BIC(0)	10^0	114	0.10	12.9	13.0	0.113	67
	10^6	114	0.10	12.9	13.0	0.113	

Blocking and Higher Order of Fill-in's improved robustness.

- Preconditioning
- **Linear Solver in “fem3D”**
- Computation of Stress
- Report #2

SOLVE33（1/4）：INPUT_CNTL

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE *fp_log;
extern void CG_3();
void SOLVE33()
{
        int i,j,k,ii,L;
        KREAL ALU[3][3];
        KREAL PW[3];
        double ALO;

        int  ERROR, ICFLAG=0;
        CHAR_LENGTH BUF;

/**
        +------------+
        | PARAMETERs |
        +------------+
**/
        ITER      = pfemIarray[0];
        METHOD    = pfemIarray[1];
        PRECOND   = pfemIarray[2];
        NSET      = pfemIarray[3];  0
        iterPREmax= pfemIarray[4];  使用せず
        NREST     = pfemIarray[5];  使用せず

        RESID     = pfemRarray[0];
        SIGMA_DIAG= pfemRarray[1];  1.0

        if( iterPREmax  < 1   ) iterPREmax= 1;
        if (iterPREmax  > 4   ) iterPREmax= 4;
```

Control File: INPUT_CNTL

```
#include <stdio.h>
#include <stdlib.h>
#include "pfem_util.h"
/** **/
void INPUT_CNTL()
{
        FILE *fp;
        if( (fp=fopen("INPUT.DAT","r")) == NULL){
                fprintf(stdout,"input file cannot be opened!¥n");
                exit(1);
        }
        fscanf(fp,"%s",fname);
        fscanf(fp,"%d %d",&METHOD,&PRECOND);
        fscanf(fp,"%d",&iterPREmax);
        fscanf(fp,"%d",&ITER);
        fscanf(fp, "%lf %lf", &ELAST, &POISSON);
        fclose(fp);

        if( ( iterPREmax < 1 ) ){
                iterPREmax= 1;
        }
        if( ( iterPREmax > 4 ) ){
                iterPREmax= 4;
        }

        SIGMA_DIAG= 1.0;
        SIGMA     = 0.0;
        RESID     = 1.e-8;
        NSET      = 0;

        pfemRarray[0]= RESID;
        pfemRarray[1]= SIGMA_DIAG;
        pfemRarray[2]= SIGMA;

        pfemIarray[0]= ITER;
        pfemIarray[1]= METHOD;
        pfemIarray[2]= PRECOND;
        pfemIarray[3]= NSET;
        pfemIarray[4]= iterPREmax;
}
```

SOLVE33（2/4）

```
/**
        +-----------+
        | BLOCK LUs |
        +-----------+
**/
        if( ICFLAG == 0 ){
                ALUG  =(KREAL*)allocate_vector(sizeof(KREAL),9*N);
                ICFLAG= 1;
                strcpy(BUF.name,"### LINEAR SOLVER:  3x3 Block" );

                if (METHOD == 1) strcat(BUF.name,"ssCG");
                if (METHOD == 2) strcat(BUF.name,"ssBiCGSTAB");

                if (PRECOND == 0) {
                    strcat(BUF.name,"BILU(0)-no ASDD");
                }

                if (PRECOND != 0) strcat(BUF.name,"Block Scaling");

                        fprintf(stdout,"%s¥n",BUF.name);
                        fprintf(fp_log,"%s¥n",BUF.name);
        }
```

SOLVE33 （3/4）

```
if( NSET == 0 ){
     for(i=0;i<9*N;i++) ALUG[i]=0.0;
          for( ii=0;ii<N;ii++){
               ALU[0][0]= D[9*ii  ]*SIGMA_DIAG;
               ALU[0][1]= D[9*ii+1];
               ALU[0][2]= D[9*ii+2];
               ALU[1][0]= D[9*ii+3];
               ALU[1][1]= D[9*ii+4]*SIGMA_DIAG;
               ALU[1][2]= D[9*ii+5];
               ALU[2][0]= D[9*ii+6];
               ALU[2][1]= D[9*ii+7];
               ALU[2][2]= D[9*ii+8]*SIGMA_DIAG;

               for(k=1;k<=3;k++){
                    L=k;
                    ALO=fabs(ALU[L-1][k-1]);
                    for( i=k+1;i<=3;i++){
                         if( fabs(ALU[i-1][k-1]) > ALO ){
                              L=i;
                              ALO=fabs(ALU[L-1][k-1]);}
                    }
                    ALU[k-1][k-1]= 1.e0/ALU[k-1][k-1];
                    for(i=k+1;i<=3;i++){
                         ALU[i-1][k-1]*=ALU[k-1][k-1];
                              for(j=k+1;j<=3;j++){
                                   PW[j-1]=ALU[i-1][j-1] - ALU[i-1][k-1]*ALU[k-1][j-1];}
                              for(j=k+1;j<=3;j++){
                                   ALU[i-1][j-1]=PW[j-1];}
                    }
               }
               ALUG[9*ii  ]=ALU[0][0];
               ALUG[9*ii+1]=ALU[0][1];
               ALUG[9*ii+2]=ALU[0][2];
               ALUG[9*ii+3]=ALU[1][0];
               ALUG[9*ii+4]=ALU[1][1];
               ALUG[9*ii+5]=ALU[1][2];
               ALUG[9*ii+6]=ALU[2][0];
               ALUG[9*ii+7]=ALU[2][1];
               ALUG[9*ii+8]=ALU[2][2];
     }
}
```

ALUG: full LU factorization of D

SIGMA_DIAG= 1.0 （INPUT_CNTL）

SOLVE33 （3/4）

```
if( NSET == 0 ){
     for(i=0;i<9*N;i++) ALUG[i]=0.0;
          for( ii=0;ii<N;ii++){
               ALU[0][0]= D[9*ii  ]*SIGMA_DIAG;
               ALU[0][1]= D[9*ii+1];
               ALU[0][2]= D[9*ii+2];
               ALU[1][0]= D[9*ii+3];
               ALU[1][1]= D[9*ii+4]*SIGMA_DIAG;
               ALU[1][2]= D[9*ii+5];
               ALU[2][0]= D[9*ii+6];
               ALU[2][1]= D[9*ii+7];
               ALU[2][2]= D[9*ii+8]*SIGMA_DIAG;

               for(k=1;k<=3;k++){
                    L=k;
                    ALO=fabs(ALU[L-1][k-1]);
                    for( i=k+1;i<=3;i++){
                         if( fabs(ALU[i-1][k-1]) > ALO ){
                              L=i;
                              ALO=fabs(ALU[L-1][k-1]);}
                    }
                    ALU[k-1][k-1]= 1.e0/ALU[k-1][k-1];
                    for(i=k+1;i<=3;i++){
                         ALU[i-1][k-1]*=ALU[k-1][k-1];
                              for(j=k+1;j<=3;j++){
                                   PW[j-1]=ALU[i-1][j-1] - ALU[i-1][k-1]*ALU[k-1][j-1];}
                              for(j=k+1;j<=3;j++){
                                   ALU[i-1][j-1]=PW[j-1];}
                    }
               }
               ALUG[9*ii  ]=ALU[0][0];
               ALUG[9*ii+1]=ALU[0][1];
               ALUG[9*ii+2]=ALU[0][2];
               ALUG[9*ii+3]=ALU[1][0];
               ALUG[9*ii+4]=ALU[1][1];
               ALUG[9*ii+5]=ALU[1][2];
               ALUG[9*ii+6]=ALU[2][0];
               ALUG[9*ii+7]=ALU[2][1];
               ALUG[9*ii+8]=ALU[2][2];
     }
}
```

ALUG: full LU factorization of D

SIGMA_DIAG= 1.0 （INPUT_CNTL）

SOLVE33 （3/4）

```
if( NSET == 0 ){
     for(i=0;i<9*N;i++) ALUG[i]=0.0;
          for( ii=0;ii<N;ii++){
               ALU[0][0]= D[9*ii  ]*SIGMA_DIAG;
               ALU[0][1]= D[9*ii+1];
               ALU[0][2]= D[9*ii+2];
               ALU[1][0]= D[9*ii+3];
               ALU[1][1]= D[9*ii+4]*SIGMA_DIAG;
               ALU[1][2]= D[9*ii+5];
               ALU[2][0]= D[9*ii+6];
               ALU[2][1]= D[9*ii+7];
               ALU[2][2]= D[9*ii+8]*SIGMA_DIAG;

               for(k=1;k<=3;k++){
                    L=k;
                    ALO=fabs(ALU[L-1][k-1]);
                    for( i=k+1;i<=3;i++){
                         if( fabs(ALU[i-1][k-1]) > A
                              L=i;
                              ALO=fabs(ALU[L-1][k-1]);
                    }
                    ALU[k-1][k-1]= 1.e0/ALU[k-1][k-1];
                    for(i=k+1;i<=3;i++){
                         ALU[i-1][k-1]*=ALU[k-1][k-1];
                              for(j=k+1;j<=3;j++){
                                   PW[j-1]=ALU[i-1][j-1] - ALU[i-1][k-1]*ALU[k-1][j-1];}
                              for(j=k+1;j<=3;j++){
                                   ALU[i-1][j-1]=PW[j-1];}
                    }
               }
               ALUG[9*ii  ]=ALU[0][0];
               ALUG[9*ii+1]=ALU[0][1];
               ALUG[9*ii+2]=ALU[0][2];
               ALUG[9*ii+3]=ALU[1][0];
               ALUG[9*ii+4]=ALU[1][1];
               ALUG[9*ii+5]=ALU[1][2];
               ALUG[9*ii+6]=ALU[2][0];
               ALUG[9*ii+7]=ALU[2][1];
               ALUG[9*ii+8]=ALU[2][2];
     }
}
```

<u>LU factorization with Full Pivoting</u>

Component with the largest absolute value becomes “pivot”

ALUG：full LU fact. of D

Pivoting

Full LU Factorization

$$\begin{pmatrix} D(9*i) & D(9*i+1) & D(9*i+2) \\ D(9*i+3) & D(9*i+4) & D(9*i+5) \\ D(9*i+6) & D(9*i+7) & D(9*i+8) \end{pmatrix} =$$

$$\begin{pmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ ALUG(9*i+3) & 1 & 0 \\ ALUG(9*i+6) & ALUG(9*i+7) & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix}$$

$$\begin{pmatrix} \underline{ALUG(9*i)} & ALUG(9*i+1) & ALUG(9*i+2) \\ 0 & \underline{ALUG(9*i+4)} & ALUG(9*i+5) \\ 0 & 0 & \underline{ALUG(9*i+8)} \end{pmatrix} = \begin{pmatrix} \underline{1/u_{11}} & u_{12} & u_{13} \\ 0 & \underline{1/u_{22}} & u_{23} \\ 0 & 0 & \underline{1/u_{33}} \end{pmatrix}$$

SOLVE33（4/4）

```
/***
        +-----------------+
        | ITERATIVE solver |
        +-----------------+
***/
        if (METHOD == 1 ) {
                CG_3( N, NP, NPL, NPU, D, AL, indexL, itemL, AU, indexU, itemU,
                        B, X,  ALUG, RESID, ITER, &ERROR,
                        PRECOND, iterPREmax);
        }

        ITERactual= ITER;
}
```

CG_3 (1/2)

```
/***
 *** CG_3
 ***/
#include <stdio.h>
#include <math.h>
#include "precision.h"
#include "allocate.h"
extern FILE *fp_log;
/***
        CG_3 solves the linear system Ax = b with 3*3 block matrix
        using the Conjugate Gradient iterative method with the following
        preconditioners for SMP nodes:
 ***/
void  CG_3(
                KINT N,KINT NP,KINT NPL, KINT NPU,KREAL D[],
                KREAL AL[],KINT INL[], KINT IAL[],
                KREAL AU[],KINT INU[], KINT IAU[],
                KREAL B[],KREAL X[],KREAL ALU[],
                KREAL RESID,KINT ITER, KINT *ERROR,
                KINT  PRECOND, KINT iterPREmax)
{
        int i,j,k;
        int ieL,isL,ieU,isU;
        double X1,X2,X3;
        double WVAL1,WVAL2,WVAL3;
        double SW1,SW2,SW3;
        double WV1,WV2,WV3;
        double BNRM20,BNRM2,DNRM20,DNRM2;
        double S1_TIME,E1_TIME;
        double ALPHA,BETA;
        double C1,C10,RHO,RHO0,RHO1;
        int    iterPRE;
        int    indexA,indexB;

        KREAL **WW;
        KINT R=0,Z=1,Q=1,P=2,ZP=3;
        KINT MAXIT;
        KREAL TOL;

        double COMPtime;
```

Variables/Arrays

Global	I/R	Size	CG_3
N, NP, NPL, NPU	I		N, NP, NPL, NPU
D, B, X	R	[3*N]	D, B, X
AL, AU	R	[9*NPL], [9*NPU]	AL, AU
indexL, indexU	I	[N+1]	INL, INU
itemL, itemU	I	[NPL], [NPU]	IAL, IAU
ALUG	R	[9*N]	ALU
RESID	R		RESID
ITER	I		ITER
PRECOND	I		PRECOND
iterPREmax	I		iterPREmax
	R	[4][3*N]	WW

CG_3 (2/2)

```
/***
        +-------+
        | INIT. |
        +-------+
***/
        ERROR= 0;

        WW=(KREAL**) allocate_matrix(sizeof(KREAL),4,3*N);

        MAXIT  = ITER;
        TOL    = RESID;

        for(i=0;i<3*N;i++){
                X[i]=0.0;
        }
        for(j=0;j<4;j++) for(i=0;i<3*N;i++) WW[j][i]=0.0;
```

```
KINT R =0  WW[0][i]: {r}
KINT Z =1  WW[1][i]: {z}
KINT Q =1  WW[1][i]: {q}
KINT P =2  WW[2][i]: {p}
KINT ZP=3  WW[3][i]
```

Compute $r^{(0)} = b - [A]x^{(0)}$

for i= 1, 2, …

solve $[M]z^{(i-1)} = r^{(i-1)}$

$\rho_{i-1} = r^{(i-1)} z^{(i-1)}$

if i=1

$p^{(1)} = z^{(0)}$

else

$\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$

$p^{(i)} = z^{(i-1)} + \beta_{i-1} p^{(i-1)}$

endif

$q^{(i)} = [A]p^{(i)}$

$\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$

$x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$

$r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$

check convergence |r|

end

Preconditioning: LU-GS (1/3)

```
/**
        +----------------+
        | {z}= [Minv]{r} |
        +----------------+
**/
     if( PRECOND == 0 ){
/**
     Block SSOR
**/
     for( i=0;i<N;i++){
          WW[ZP][3*i  ]=WW[R][3*i  ];
          WW[ZP][3*i+1]=WW[R][3*i+1];
          WW[ZP][3*i+2]=WW[R][3*i+2];
        }

        for( i=0;i<N;i++){
          WW[Z][3*i  ]=0.e0;
          WW[Z][3*i+1]=0.e0;
          WW[Z][3*i+2]=0.e0;
        }
```

Preconditioning: LU-GS (2/3)

```
/**
    FORWARD
**/
 for( i=0;i<N;i++){
          SW1= WW[ZP][3*i  ];
          SW2= WW[ZP][3*i+1];
          SW3= WW[ZP][3*i+2];

          isL=INL[i];
          ieL=INL[i+1];
          for(j=isL;j<ieL;j++){
            k=IAL[j];
            X1= WW[ZP][3*k  ];
            X2= WW[ZP][3*k+1];
            X3= WW[ZP][3*k+2];
            SW1+=  - AL[9*j  ]*X1 - AL[9*j+1]*X2 - AL[9*j+2]*X3;
            SW2+=  - AL[9*j+3]*X1 - AL[9*j+4]*X2 - AL[9*j+5]*X3;
            SW3+=  - AL[9*j+6]*X1 - AL[9*j+7]*X2 - AL[9*j+8]*X3;
          }

          X1= SW1;
          X2= SW2;
          X3= SW3;
          X2= X2 - ALU[9*i+3]*X1;
          X3= X3 - ALU[9*i+6]*X1 - ALU[9*i+7]*X2;
          X3= ALU[9*i+8]*  X3;
          X2= ALU[9*i+4]*( X2 - ALU[9*i+5]*X3 );
          X1= ALU[9*i  ]*( X1 - ALU[9*i+2]*X3 - ALU[9*i+1]*X2);

          WW[ZP][3*i  ]= X1;
          WW[ZP][3*i+1]= X2;
          WW[ZP][3*i+2]= X3;
        }
```

indexL ⇒ INL
itemL ⇒ IAL

$$\left[\overline{L}+\overline{D}\right]\{y\}=\{r\}\Rightarrow\{y\}=\left[\overline{D}^{-1}\right]\left(\{r\}-\left[\overline{L}\right]\{y\}\right)\Rightarrow y_i=\overline{D}_{ii}^{-1}\left(r_i-\sum_{j=1}^{i-1}\overline{L}_{ij}y_j\right)$$

Preconditioning: LU-GS (3/3)

$$\left[I+\overline{D}^{-1}\overline{U}\right]\{z\}=\{y\}\Rightarrow\{z\}=\{y\}-\left[\overline{D}^{-1}\right]\left[\overline{U}\right]\{z\}$$

$$\Rightarrow z_i = y_i - \overline{D}_{ii}^{-1}\left[\sum_{j=i+1}^{N}\overline{U}_{ij}z_j\right]$$

indexU ⇒ INU
itemU ⇒ IAU

```
/**
    BACKWARD
**/
    for(i=N-1;i>=0;i--){
          isU= INU[i];
          ieU= INU[i+1];
          SW1= 0.e0;
          SW2= 0.e0;
          SW3= 0.e0;

          for(j=isU;j<ieU;j++){
            k=IAU[j];

            X1=WW[ZP][3*k  ];
            X2=WW[ZP][3*k+1];
            X3=WW[ZP][3*k+2];
            SW1+=  + AU[9*j  ]*X1 + AU[9*j+1]*X2 + AU[9*j+2]*X3;
            SW2+=  + AU[9*j+3]*X1 + AU[9*j+4]*X2 + AU[9*j+5]*X3;
            SW3+=  + AU[9*j+6]*X1 + AU[9*j+7]*X2 + AU[9*j+8]*X3;
          }

          X1= SW1;
          X2= SW2;
          X3= SW3;
          X2= X2 - ALU[9*i+3]*X1;
          X3= X3 - ALU[9*i+6]*X1 - ALU[9*i+7]*X2;
          X3= ALU[9*i+8]*  X3;
          X2= ALU[9*i+4]*( X2 - ALU[9*i+5]*X3 );
          X1= ALU[9*i  ]*( X1 - ALU[9*i+2]*X3 - ALU[9*i+1]*X2);
          WW[ZP][3*i  ]+= -X1;
          WW[ZP][3*i+1]+= -X2;
          WW[ZP][3*i+2]+= -X3;
    }

         for( i=1;i<=N;i++){
              WW[Z][3*i-3]= WW[ZP][3*i-3];
              WW[Z][3*i-2]= WW[ZP][3*i-2];
              WW[Z][3*i-1]= WW[ZP][3*i-1];
         }
    }
```

Preconditioning: Block Scaling

```
    if (PRECOND != 0 ) {
/**
       Block SCALING
**/
         for(i=0;i<N;i++){
              WW[Z][3*i  ]= WW[R][3*i  ];
              WW[Z][3*i+1]= WW[R][3*i+1];
              WW[Z][3*i+2]= WW[R][3*i+2];
         }

         for(i=0;i<N;i++){
              X1=WW[Z][3*i  ];
              X2=WW[Z][3*i+1];
              X3=WW[Z][3*i+2];
              X2= X2 - ALU[9*i+3]*X1;
              X3= X3 - ALU[9*i+6]*X1 - ALU[9*i+7]*X2;
              X3= ALU[9*i+8]*  X3;
              X2= ALU[9*i+4]*( X2 - ALU[9*i+5]*X3 );
              X1= ALU[9*i  ]*( X1 - ALU[9*i+2]*X3 - ALU[9*i+1]*X2);
              WW[Z][3*i  ]= X1;
              WW[Z][3*i+1]= X2;
              WW[Z][3*i+2]= X3;
         }
    }
```

Forward/Backward Substitution by LU factorization of D (ALU)

Full LU Factorization

$$\begin{pmatrix} D(9*i) & D(9*i+1) & D(9*i+2) \\ D(9*i+3) & D(9*i+4) & D(9*i+5) \\ D(9*i+6) & D(9*i+7) & D(9*i+8) \end{pmatrix} =$$

$$\begin{pmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ ALUG(9*i+3) & 1 & 0 \\ ALUG(9*i+6) & ALUG(9*i+7) & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix}$$

$$\begin{pmatrix} \underline{ALUG(9*i)} & ALUG(9*i+1) & ALUG(9*i+2) \\ 0 & \underline{ALUG(9*i+4)} & ALUG(9*i+5) \\ 0 & 0 & \underline{ALUG(9*i+8)} \end{pmatrix} = \begin{pmatrix} \underline{1/u_{11}} & u_{12} & u_{13} \\ 0 & \underline{1/u_{22}} & u_{23} \\ 0 & 0 & \underline{1/u_{33}} \end{pmatrix}$$

Preconditioning: Block Scaling

```
    if (PRECOND != 0 ) {
/**
      Block SCALING
**/
        for(i=0;i<N;i++){
            WW[Z][3*i  ]= WW[R][3*i  ];
            WW[Z][3*i+1]= WW[R][3*i+1];
            WW[Z][3*i+2]= WW[R][3*i+2];
        }

        for(i=0;i<N;i++){
            X1=WW[Z][3*i  ];
            X2=WW[Z][3*i+1];
            X3=WW[Z][3*i+2];
            X2= X2 - ALU[9*i+3]*X1;
            X3= X3 - ALU[9*i+6]*X1 - ALU[9*i+7]*X2;
            X3= ALU[9*i+8]*  X3;
            X2= ALU[9*i+4]*( X2 - ALU[9*i+5]*X3 );
            X1= ALU[9*i  ]*( X1 - ALU[9*i+2]*X3 - ALU[9*i+1]*X2);
            WW[Z][3*i  ]= X1;
            WW[Z][3*i+1]= X2;
            WW[Z][3*i+2]= X3;
        }
    }
```

Forward/Backward Substitution by LU factorization of D (ALU)

$$\begin{pmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{pmatrix}\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix}\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix}$$

Forward Substitution

```
    if (PRECOND != 0 ) {
/**
      Block SCALING
**/
        for(i=0;i<N;i++){
            WW[Z][3*i  ]= WW[R][3*i  ];
            WW[Z][3*i+1]= WW[R][3*i+1];
            WW[Z][3*i+2]= WW[R][3*i+2];
        }

        for(i=0;i<N;i++){
            X1=WW[Z][3*i  ];
            X2=WW[Z][3*i+1];
            X3=WW[Z][3*i+2];
            X2= X2 - ALU[9*i+3]*X1;
            X3= X3 - ALU[9*i+6]*X1 - ALU[9*i+7]*X2;
            X3= ALU[9*i+8]*  X3;
            X2= ALU[9*i+4]*( X2 - ALU[9*i+5]*X3 );
            X1= ALU[9*i  ]*( X1 - ALU[9*i+2]*X3 - ALU[9*i+1]*X2);
            WW[Z][3*i  ]= X1;
            WW[Z][3*i+1]= X2;
            WW[Z][3*i+2]= X3;
        }
    }
```

$$\begin{pmatrix} 1 & 0 & 0 \\ \mathrm{ALUG}(9*i+3) & 1 & 0 \\ \mathrm{ALUG}(9*i+6) & \mathrm{ALUG}(9*i+7) & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix}$$

$$[L]\{y\} = \{r\}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \rightarrow$$

$$y_1 = r_1$$
$$y_2 = r_2 - l_{21} \times y_1$$
$$y_3 = r_3 - l_{31} \times y_1 - l_{32} \times y_2 \rightarrow$$

$$x_1 = x_1$$
$$x_2 = x_2 - l_{21} \times x_1$$
$$x_3 = x_3 - l_{31} \times x_1 - l_{32} \times x_2$$

Backward Substitution

```
    if (PRECOND != 0 ) {
/**
      Block SCALING
**/
        for(i=0;i<N;i++){
            WW[Z][3*i  ]= WW[R][3*i  ];
            WW[Z][3*i+1]= WW[R][3*i+1];
            WW[Z][3*i+2]= WW[R][3*i+2];
        }

        for(i=0;i<N;i++){
            X1=WW[Z][3*i  ];
            X2=WW[Z][3*i+1];
            X3=WW[Z][3*i+2];
            X2= X2 - ALU[9*i+3]*X1;
            X3= X3 - ALU[9*i+6]*X1 - ALU[9*i+7]*X2;
            X3= ALU[9*i+8]*  X3;
            X2= ALU[9*i+4]*( X2 - ALU[9*i+5]*X3 );
            X1= ALU[9*i  ]*( X1 - ALU[9*i+2]*X3 - ALU[9*i+1]*X2);
            WW[Z][3*i  ]= X1;
            WW[Z][3*i+1]= X2;
            WW[Z][3*i+2]= X3;
        }
    }
```

$$\begin{pmatrix} \text{ALUG}(9*i) & \text{ALUG}(9*i+1) & \text{ALUG}(9*i+2) \\ 0 & \text{ALUG}(9*i+4) & \text{ALUG}(9*i+5) \\ 0 & 0 & \text{ALUG}(9*i+8) \end{pmatrix} = \begin{pmatrix} 1/u_{11} & u_{12} & u_{13} \\ 0 & 1/u_{22} & u_{23} \\ 0 & 0 & 1/u_{33} \end{pmatrix}$$

$$[U]\{z\} = \{y\}$$

$$\begin{pmatrix} u_{11} & u_{12} & u_{12} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

$$z_3 = [1/u_{33}] \times [y_3]$$
$$z_2 = [1/u_{22}] \times [y_2 - u_{23} \times z_3]$$
$$z_1 = [1/u_{11}] \times [y_1 - u_{13} \times z_3 - u_{12} \times z_2]$$

$$x_3 = [1/u_{33}] \times [x_3]$$
$$x_2 = [1/u_{22}] \times [x_2 - u_{23} \times x_3]$$
$$x_1 = [1/u_{11}] \times [x_1 - u_{13} \times x_3 - u_{12} \times x_2]$$

Sparse Matrix-Vector Multiplication

```
/***
        +-------------+
        | {q}= [A]{p} |
        +-------------+
***/
     for( j=0;j<N;j++){
          X1=WW[P][3*j  ];
          X2=WW[P][3*j+1];
          X3=WW[P][3*j+2];

          WVAL1= D[9*j  ]*X1 + D[9*j+1]*X2 + D[9*j+2]*X3;
          WVAL2= D[9*j+3]*X1 + D[9*j+4]*X2 + D[9*j+5]*X3;
          WVAL3= D[9*j+6]*X1 + D[9*j+7]*X2 + D[9*j+8]*X3;
          for(k=INL[j];k<INL[j+1];k++){
            i=IAL[k];
            X1=WW[P][3*i  ];
            X2=WW[P][3*i+1];
            X3=WW[P][3*i+2];
            WVAL1+=   AL[9*k  ]*X1 + AL[9*k+1]*X2 + AL[9*k+2]*X3;
            WVAL2+=   AL[9*k+3]*X1 + AL[9*k+4]*X2 + AL[9*k+5]*X3;
            WVAL3+=   AL[9*k+6]*X1 + AL[9*k+7]*X2 + AL[9*k+8]*X3;
          }
          for(k=INU[j];k<INU[j+1];k++){
            i=IAU[k];
            X1=WW[P][3*i  ];
            X2=WW[P][3*i+1];
            X3=WW[P][3*i+2];
            WVAL1+=   AU[9*k  ]*X1 + AU[9*k+1]*X2 + AU[9*k+2]*X3;
            WVAL2+=   AU[9*k+3]*X1 + AU[9*k+4]*X2 + AU[9*k+5]*X3;
            WVAL3+=   AU[9*k+6]*X1 + AU[9*k+7]*X2 + AU[9*k+8]*X3;
          }
          WW[Q][3*j  ]=WVAL1;
          WW[Q][3*j+1]=WVAL2;
          WW[Q][3*j+2]=WVAL3;
        }
```

DAXPY, Dot Products

```
/***
        +----------------------+
        | {x}= {x} + ALPHA*{p} |
        | {r}= {r} - ALPHA*{q} |
        +----------------------+
***/
     for(i=0;i<N;i++){
          X[3*i  ]+=ALPHA *WW[P][3*i  ];
          X[3*i+1]+=ALPHA *WW[P][3*i+1];
          X[3*i+2]+=ALPHA *WW[P][3*i+2];
          WW[R][3*i  ][R]+= -ALPHA *WW[Q][3*i  ];
          WW[R][3*i+1][R]+= -ALPHA *WW[Q][3*i+1];
          WW[R][3*i+2][R]+= -ALPHA *WW[Q][3*i+2];
     }

     DNRM20= 0.e0;
     for(i=0;i<N;i++){
          DNRM20+= WW[R][3*I  ]*WW[R][3*I  ] +
                   WW[R][3*i+1]*WW[R][3*i+1] +
                   WW[R][3*i+2]*WW[R][3*i+2];
     }

     DNRM2= DNRM20;
     RESID= sqrt(DNRM2/BNRM2);
```

DAXPY, Dot Products

```
/***
        +----------------------+
        | {x}= {x} + ALPHA*{p} |
        | {r}= {r} - ALPHA*{q} |
        +----------------------+
***/
     for(i=0;i<N;i++){
          X[3*i  ]+=ALPHA *WW[3*i  ][P];
          X[3*i+1]+=ALPHA *WW[3*i+1][P];
          X[3*i+2]+=ALPHA *WW[3*i+2][P];
          WW[3*i  ][R]+= -ALPHA *WW[3*i  ][Q];
          WW[3*i+1][R]+= -ALPHA *WW[3*i+1][Q];
          WW[3*i+2][R]+= -ALPHA *WW[3*i+2][Q];
     }

     DNRM20= 0.e0;
     for(i=0;i<N;i++){
          DNRM20+= WW[R][3*I  ]*WW[R][3*I  ] +
                   WW[R][3*i+1]*WW[R][3*i+1] +
                   WW[R][3*i+2]*WW[R][3*i+2];
     }

     DNRM2= DNRM20;
     RESID= sqrt(DNRM2/BNRM2);
```

- Preconditioning
- Linear Solver in “fem3D”
- **Computation of Stress**
- Report #2

Stress

- So far, “displacement” at each node has been computed.
- “Stress” is important from the engineering point of view !!
  - “stress” is calculated by “strain”, which is derivative of “displacement” (or rate of displacement)

Strain-Stress Relationship

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{Bmatrix} = \underbrace{\boxed{\frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}(1-2\nu) & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2}(1-2\nu) & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2}(1-2\nu) \end{bmatrix}}}_{[D]} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{Bmatrix}$$

$$\{\sigma\} = [D]\{\varepsilon\}$$

Strain-Stress Relationship

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{Bmatrix} = \begin{bmatrix} \text{valX} & \text{valA} & \text{valA} & 0 & 0 & 0 \\ \text{valA} & \text{valX} & \text{valA} & 0 & 0 & 0 \\ \text{valA} & \text{valA} & \text{valX} & 0 & 0 & 0 \\ 0 & 0 & 0 & \text{valB} & 0 & 0 \\ 0 & 0 & 0 & 0 & \text{valB} & 0 \\ 0 & 0 & 0 & 0 & 0 & \text{valB} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{Bmatrix}$$

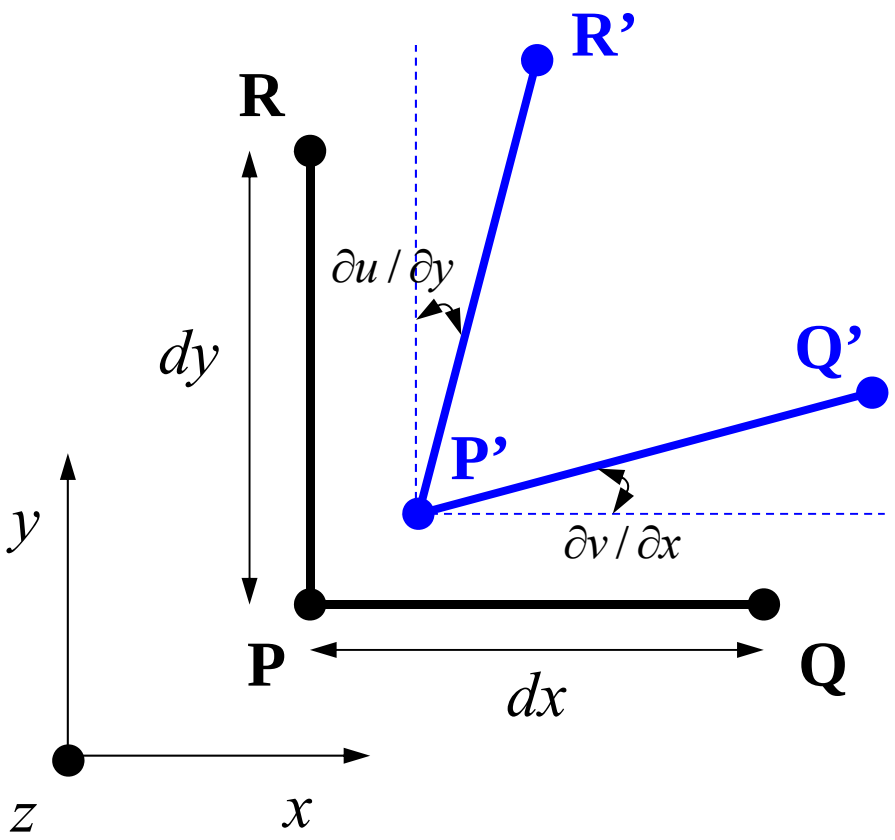
$$[D]$$

$$\{\sigma\} = [D]\{\varepsilon\}$$

Normal Strain - Displacement

- PQ⇒P'Q'

$$\varepsilon_x = \frac{\left\{\left(x + dx + u + \frac{\partial u}{\partial x} dx\right) - (x + u)\right\} - dx}{dx} = \frac{\partial u}{\partial x}$$

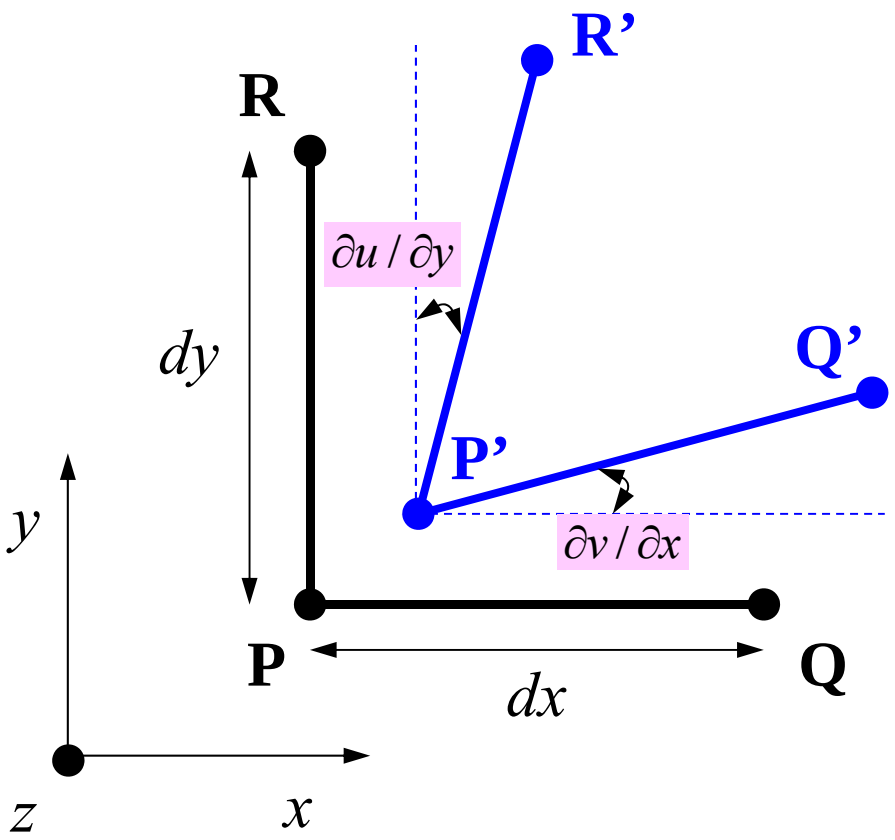


$$\varepsilon_x = \frac{\partial u}{\partial x}$$

$$\varepsilon_y = \frac{\partial v}{\partial y}$$

$$\varepsilon_z = \frac{\partial w}{\partial z}$$

Shear Strain - Displacement



$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}$$

$$\gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}$$

$$\gamma_{zx} = \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}$$

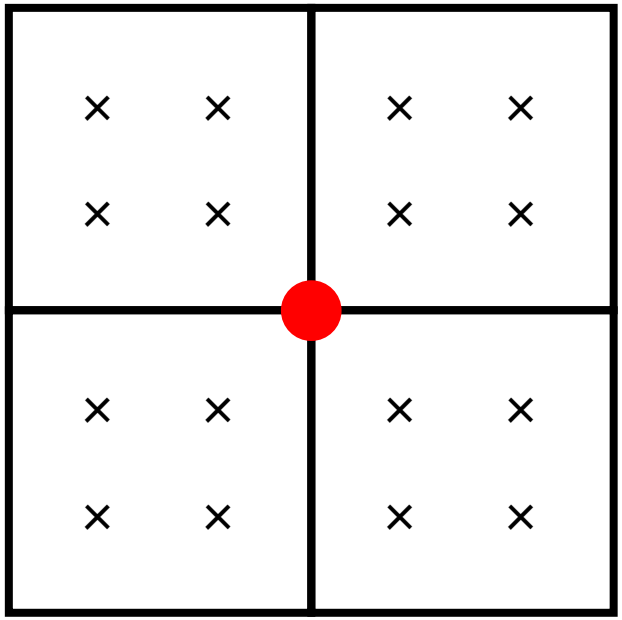
Computation of Stress Components

$$\sigma_x = \frac{E}{(1+\nu)(1-2\nu)}\left[(1-\nu)\varepsilon_x + \nu\varepsilon_y + \nu\varepsilon_z\right]$$

$$= \frac{E}{(1+\nu)(1-2\nu)}\left[(1-\nu)\frac{\partial u}{\partial x} + \nu\frac{\partial v}{\partial y} + \nu\frac{\partial w}{\partial z}\right]$$

Stress

- So far, “displacement” at each node has been computed.
- “Stress” is important from the engineering point of view !!
  - “stress” is calculated by “strain”, which is derivative of “displacement”
- Accurate stress components are calculated at Gussian Quad. Points.
- Stress at nodes (vertices)
  - Averaged Value



Computation of Stress Components

$$\sigma_x = \frac{E}{(1+\nu)(1-2\nu)}\left[(1-\nu)\varepsilon_x + \nu\varepsilon_y + \nu\varepsilon_z\right]$$

$$= \frac{E}{(1+\nu)(1-2\nu)}\left[(1-\nu)\frac{\partial u}{\partial x} + \nu\frac{\partial v}{\partial y} + \nu\frac{\partial w}{\partial z}\right]$$

- Galerkin Method: Ave. Elem. Stress X Volume

$$\int_V [N]^T \sigma_x dV = \int_V [N]^T \left\{\frac{E}{(1+\nu)(1-2\nu)}\left[(1-\nu)u_{,x} + \nu v_{,y} + \nu w_{,z}\right]\right\} dV$$

$$= \int_V [N]^T \left\{\frac{E}{(1+\nu)(1-2\nu)}\left((1-\nu)[N_{,x}]\{U\} + \nu[N_{,y}]\{V\} + \nu[N_{,z}]\{W\}\right)\right\} dV$$

1D: Elem.-by-Elem. Integration: $\{f\}$

$$N_i = \left(\frac{X_j - x}{L}\right), \quad N_j = \left(\frac{x - X_i}{L}\right) \qquad \frac{dN_i}{dx} = \left(\frac{-1}{L}\right), \quad \frac{dN_j}{dx} = \left(\frac{1}{L}\right)$$

$$\int_V X[N]^T dV = XA\int_0^L \begin{bmatrix} 1 - x/L \\ x/L \end{bmatrix} dx = \frac{XAL}{2}\begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

Body Force

A : Sectional Area

L : Length of Element

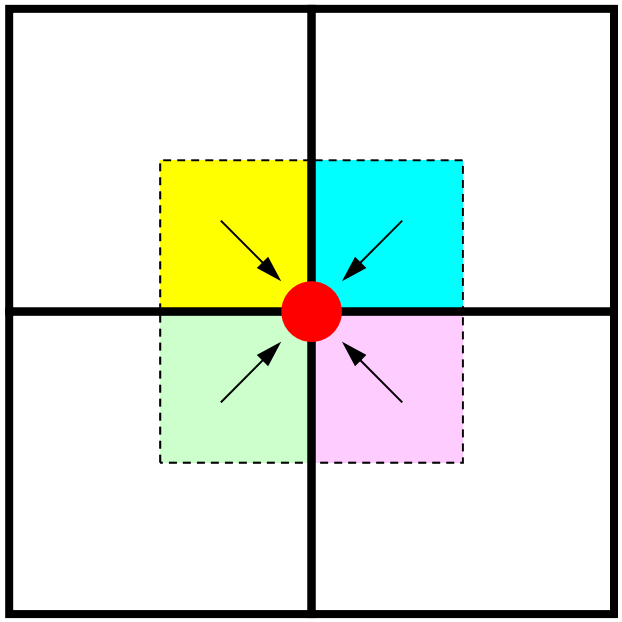
2D: Stress Components at Node

$$\int_V [N]^T \sigma_x dV$$

"Volume of Contribution X Stress" at each node from surrounding elements

$$\int_V [N]^T dV$$

"Volume Contribution" at each node



$$\therefore \quad \sigma_x = \frac{\sum \int_V [N]^T \sigma_x dV}{\sum \int_V [N]^T dV}$$

Average Stress at Each Node

RECOVER_STRESS: Stress (1/4)

```
#include <math.h>
#include "pfem_util.h"
#include "allocate.h"
extern   void JACOBI();
void RECOVER_STRESS()
{
     int  i,k,kk,icel;
     int  ie,je,ip,jp;
     int  ipn,jpn,kpn;
     int  iiS,iiE;
     double   RB;
     double   UUi,VVi,WWi,UUj,VVj,WWj;
     double   valX,valA,valB,E0,POI0,VOL,coef;
     int  in1,in2,in3,in4,in5,in6,in7,in8;
     double   X1,X2,X3,X4,X5,X6,X7,X8;
     double   Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8;
     double   Z1,Z2,Z3,Z4,Z5,Z6,Z7,Z8;
     double   SHi,SHj;
     double   EPS_xx,EPS_yy,EPS_zz,GAM_xy,GAM_xz,GAM_yz;

     KINT nodLOCAL[8];

     SIGMA_N=(KREAL*)allocate_vector(sizeof(KREAL),3*N);
     TAU_N  =(KREAL*)allocate_vector(sizeof(KREAL),3*N);

     for(i=0;i<3*N;i++) SIGMA_N[i]=0.0;
     for(i=0;i<3*N;i++) TAU_N[i]=0.0;
     for(i=0;i<3*N;i++) B[i]=0.0;

     for( icel=0;icel<  ICELTOT;icel++){
          E0   =    ELAST;
          POI0=     POISSON;

          valA=     POI0 /     (1.e0-POI0);
          valB=     (1.e0-2.e0*POI0)/(2.e0*(1.e0-POI0));
          valX=     E0*  (1.e0-POI0)/((1.e0+POI0)*(1.e0-2.e0*POI0));

          valA=     valA *    valX;
          valB=     valB *    valX;
```

RECOVER_STRESS: Stress (2/4)

```
          in1= ICELNOD[icel][0];
          in2= ICELNOD[icel][1];
          in3= ICELNOD[icel][2];
          in4= ICELNOD[icel][3];
          in5= ICELNOD[icel][4];
          in6= ICELNOD[icel][5];
          in7= ICELNOD[icel][6];
          in8= ICELNOD[icel][7];
         nodLOCAL[0]=  in1;
         nodLOCAL[1]=  in2;
         nodLOCAL[2]=  in3;
         nodLOCAL[3]=  in4;
         nodLOCAL[4]=  in5;
         nodLOCAL[5]=  in6;
         nodLOCAL[6]=  in7;
         nodLOCAL[7]=  in8;
         X1=  XYZ[in1-1][0];
         X2=  XYZ[in2-1][0];
 (中略)
         X7=  XYZ[in7-1][0];
         X8=  XYZ[in8-1][0];
         Y1=  XYZ[in1-1][1];
         Y2=  XYZ[in2-1][1];
 (中略)
         Y7=  XYZ[in7-1][1];
         Y8=  XYZ[in8-1][1];
         Z1=  XYZ[in1-1][2];
         Z2=  XYZ[in2-1][2];
 (中略)
         Z7=  XYZ[in7-1][2];
         Z8=  XYZ[in8-1][2];

/**
    JACOBIAN &    inv-JACOBIAN
**/
         JACOBI    (DETJ, PNQ, PNE, PNT, PNX, PNY, PNZ,
                    X1, X2, X3, X4, X5, X6, X7, X8,
                    Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8,
                    Z1, Z2, Z3, Z4, Z5, Z6, Z7, Z8          );
```

RECOVER_STRESS: Stress (3/4)

```
/**
    MATRIX
**/
        for(ie=0;ie<8;ie++){
            ip= nodLOCAL[ie];
            for(je=0;je<8;je++){
                jp= nodLOCAL[je];
                UUj= X[3*jp-3];
                VVj= X[3*jp-2];
                WWj= X[3*jp-1];
                UUi= X[3*ip-3];
                VVi= X[3*ip-2];
                WWi= X[3*ip-1];

                EPS_xx=  0.e0;
                EPS_yy=  0.e0;
                EPS_zz=  0.e0;
                GAM_xy=  0.e0;
                GAM_xz=  0.e0;
                GAM_yz=  0.e0;
                GAM_xy=  0.e0;
                GAM_xz=  0.e0;
                GAM_yz=  0.e0;

                VOL  =   0.e0;
                for( ipn=0;ipn<2;ipn++){
                    for( jpn=0;jpn<2;jpn++){
                        for( kpn=0;kpn<2;kpn++){
                            coef= fabs(DETJ[ipn][jpn][kpn])*WEI[ipn]*WEI[jpn]*WEI[kpn];
                            SHi= SHAPE[ipn][jpn][kpn][ie]          *          coef;

                            EPS_xx+= SHi*PNX[ipn][jpn][kpn][je];
                            EPS_yy+= SHi*PNY[ipn][jpn][kpn][je];
                            EPS_zz+= SHi*PNZ[ipn][jpn][kpn][je];
                            GAM_xy+= SHi*PNX[ipn][jpn][kpn][je] * VVj
                                 +   SHi*PNY[ipn][jpn][kpn][je] * UUj;
                            GAM_xz+= SHi*PNX[ipn][jpn][kpn][je] * WWj
                                 +   SHi*PNZ[ipn][jpn][kpn][je] * UUj;
                            GAM_yz+= SHi*PNY[ipn][jpn][kpn][je] * WWj
                                 +   SHi*PNZ[ipn][jpn][kpn][je] * VVj;
                            VOL  =   VOL + SHi; }}}
```

Displacement at each node

RECOVER_STRESS: Stress (3/4)

```
/**
    MATRIX
**/
        for(ie=0;ie<8;ie++){
            ip= nodLOCAL[ie];
            for(je=0;je<8;je++){
                jp= nodLOCAL[je];
                UUj= X[3*jp-3];
                VVj= X[3*jp-2];
                WWj= X[3*jp-1];
                UUi= X[3*ip-3];
                VVi= X[3*ip-2];
                WWi= X[3*ip-1];

                EPS_xx=  0.e0;
                EPS_yy=  0.e0;
                EPS_zz=  0.e0;
                GAM_xy=  0.e0;
                GAM_xz=  0.e0;
                GAM_yz=  0.e0;
                GAM_xy=  0.e0;
                GAM_xz=  0.e0;
                GAM_yz=  0.e0;

                VOL  =   0.e0;
                for( ipn=0;ipn<2;ipn++){
                    for( jpn=0;jpn<2;jpn++){
                        for( kpn=0;kpn<2;kpn++){
                            coef= fabs(DETJ[ipn][jpn][kpn])*WEI[ipn]*WEI[jpn]*WEI[kpn];
                            SHi= SHAPE[ipn][jpn][kpn][ie]          *          coef;

                            EPS_xx+= SHi*PNX[ipn][jpn][kpn][je];
                            EPS_yy+= SHi*PNY[ipn][jpn][kpn][je];
                            EPS_zz+= SHi*PNZ[ipn][jpn][kpn][je];
                            GAM_xy+= SHi*PNX[ipn][jpn][kpn][je] * VVj
                                 +   SHi*PNY[ipn][jpn][kpn][je] * UUj;
                            GAM_xz+= SHi*PNX[ipn][jpn][kpn][je] * WWj
                                 +   SHi*PNZ[ipn][jpn][kpn][je] * UUj;
                            GAM_yz+= SHi*PNY[ipn][jpn][kpn][je] * WWj
                                 +   SHi*PNZ[ipn][jpn][kpn][je] * VVj;
                            VOL  =   VOL + SHi; }}}
```

$$u_{,x} = \lfloor N_{,x} \rfloor \{U\}, \quad u_{,y} = \lfloor N_{,y} \rfloor \{U\}, \quad u_{,z} = \lfloor N_{,z} \rfloor \{U\}$$

$$v_{,x} = \lfloor N_{,x} \rfloor \{V\}, \quad v_{,y} = \lfloor N_{,y} \rfloor \{V\}$$

$$w_{,x} = \lfloor N_{,x} \rfloor \{W\}, \quad w_{,z} = \lfloor N_{,z} \rfloor \{W\}$$

RECOVER_STRESS: Stress (4/4)

```
                    EPS_xx= EPS_xx * UUj;
                    EPS_yy= EPS_yy * VVj;
                    EPS_zz= EPS_zz * WWj;

                    SIGMA_N[3*ip-3]+= valX*EPS_xx+ valA*EPS_yy + valA*EPS_zz;
                    SIGMA_N[3*ip-2]+= valA*EPS_xx+ valX*EPS_yy + valA*EPS_zz;
                    SIGMA_N[3*ip-1]+= valA*EPS_xx+ valA*EPS_yy + valX*EPS_zz;
                    TAU_N[3*ip-3]+= GAM_xy*valB;
                    TAU_N[3*ip-2]+= GAM_xz*valB;
                    TAU_N[3*ip-1]+= GAM_yz*valB;
                    if (ip==jp) B[ip-1]+=VOL;
               }
          }
     }

/****
     NODAL     VALUE
***/

     for(i=0;i<N;i++){
          RB=1.0e0/B[i];

          SIGMA_N[3*I  ]*=RB;
          SIGMA_N[3*i+1]*=RB;
          SIGMA_N[3*i+2]*=RB;

          TAU_N[3*i ]*=RB;
          TAU_N[3*i+1]*=RB;
          TAU_N[3*i+2]*=RB;

          }
     }
```

$$u_{,x} = \lfloor N_{,x} \rfloor \{U\}, \quad u_{,y} = \lfloor N_{,y} \rfloor \{U\}, \quad u_{,z} = \lfloor N_{,z} \rfloor \{U\}$$

$$v_{,x} = \lfloor N_{,x} \rfloor \{V\}, \quad v_{,y} = \lfloor N_{,y} \rfloor \{V\}$$

$$w_{,x} = \lfloor N_{,x} \rfloor \{W\}, \quad w_{,z} = \lfloor N_{,z} \rfloor \{W\}$$

RECOVER_STRESS: Stress (4/4)

```
                    EPS_xx= EPS_xx * UUj;
                    EPS_yy= EPS_yy * VVj;
                    EPS_zz= EPS_zz * WWj;

                    SIGMA_N[3*ip-3]+= valX*EPS_xx+ valA*EPS_yy + valA*EPS_zz;
                    SIGMA_N[3*ip-2]+= valA*EPS_xx+ valX*EPS_yy + valA*EPS_zz;
                    SIGMA_N[3*ip-1]+= valA*EPS_xx+ valA*EPS_yy + valX*EPS_zz;
                    TAU_N[3*ip-3]+= GAM_xy*valB;
                    TAU_N[3*ip-2]+= GAM_xz*valB;
                    TAU_N[3*ip-1]+= GAM_yz*valB;
                    if (ip==jp) B[ip-1]+=VOL;
                }
            }
        }

/****
    NODAL     VALUE
***/

    for(i=0;i<N;i++){
        RB=1.0e0/B[i];

        SIGMA_N[3*I  ]*=RB;
        SIGMA_N[3*i+1]*=RB;
        SIGMA_N[3*i+2]*=RB;

        TAU_N[3*i ]*=RB;
        TAU_N[3*i+1]*=RB;
        TAU_N[3*i+2]*=RB;

        }
    }
```

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{Bmatrix} = \begin{bmatrix} \mathrm{valX} & \mathrm{valA} & \mathrm{valA} & 0 & 0 & 0 \\ \mathrm{valA} & \mathrm{valX} & \mathrm{valA} & 0 & 0 & 0 \\ \mathrm{valA} & \mathrm{valA} & \mathrm{valX} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{valB} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathrm{valB} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathrm{valB} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{Bmatrix}$$

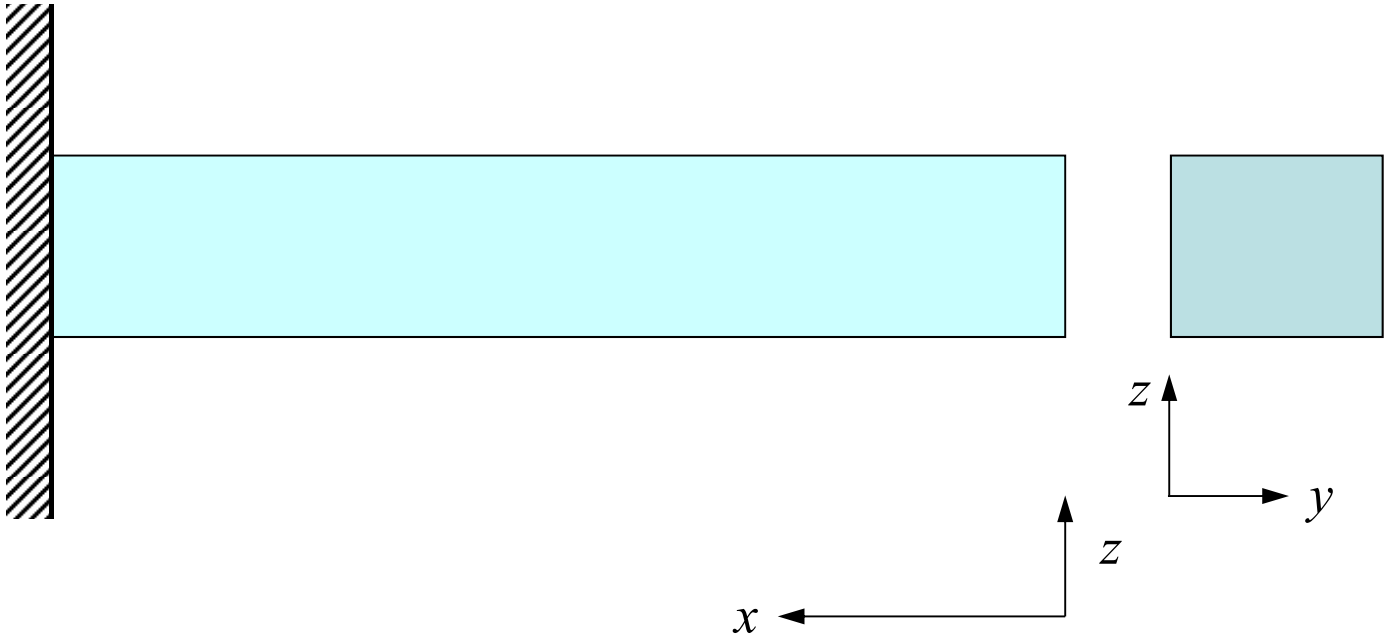
RECOVER_STRESS: Stress (4/4)

```
                    EPS_xx= EPS_xx * UUj;
                    EPS_yy= EPS_yy * VVj;
                    EPS_zz= EPS_zz * WWj;

                    SIGMA_N[3*ip-3]+= valX*EPS_xx+ valA*EPS_yy + valA*EPS_zz;
                    SIGMA_N[3*ip-2]+= valA*EPS_xx+ valX*EPS_yy + valA*EPS_zz;
                    SIGMA_N[3*ip-1]+= valA*EPS_xx+ valA*EPS_yy + valX*EPS_zz;
                    TAU_N[3*ip-3]+= GAM_xy*valB;
                    TAU_N[3*ip-2]+= GAM_xz*valB;
                    TAU_N[3*ip-1]+= GAM_yz*valB;
                    if (ip==jp) B[ip-1]+=VOL;
                }
            }
        }

/****
    NODAL     VALUE
***/

    for(i=0;i<N;i++){
        RB=1.0e0/B[i];

        SIGMA_N[3*I  ]*=RB;
        SIGMA_N[3*i+1]*=RB;
        SIGMA_N[3*i+2]*=RB;

        TAU_N[3*i ]*=RB;
        TAU_N[3*i+1]*=RB;
        TAU_N[3*i+2]*=RB;

        }
    }
```

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{Bmatrix} = \begin{bmatrix} \mathrm{valX} & \mathrm{valA} & \mathrm{valA} & 0 & 0 & 0 \\ \mathrm{valA} & \mathrm{valX} & \mathrm{valA} & 0 & 0 & 0 \\ \mathrm{valA} & \mathrm{valA} & \mathrm{valX} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{valB} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathrm{valB} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathrm{valB} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{Bmatrix}$$

- Preconditioning
- Linear Solver in “fem3D”
- Computation of Stress
- **Report #2**

Report #2 (1/2)



- Implement and evaluate a 3D-linear-elastic-code solving the following problem:
  - Cantilevered Beam with Rect. Section
    - E=1.00, ρ (density)=0.025, ν=0.30
  - u=v=w=0@x=Xmax=L
  - Set initial value of "NU" and NL" in mat_con0 as "5"

Report #2 (2/2)

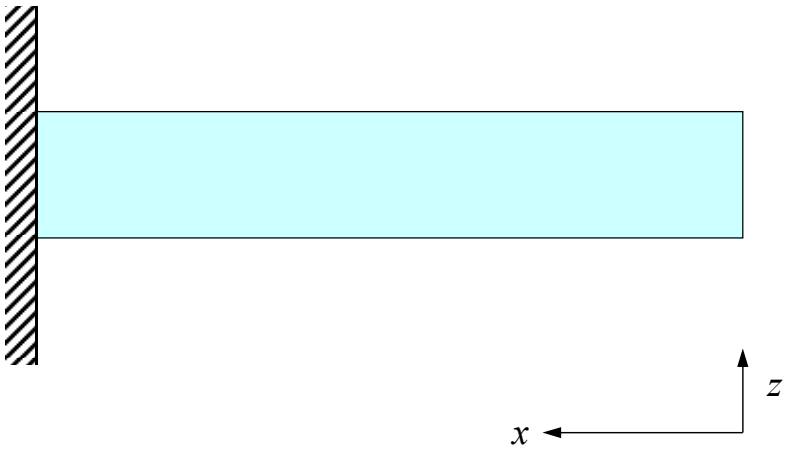
- Modify mesh generation program
- Modify “fem3d” (Boundary Conditions, Body Force)
- Evaluation (Comparison with Analytical Solution, Effect of Mesh)
- Apply various numbers of Gaussian quad. points
  - n=2（fem3d），n=1，n=3
- Try ILU(0) (or IC(0)) and GS as preconditioners (if you have time)
- Report
  - Due on August 18th (M), 2014 at 17:00
  - Documents
    - Report (Outline, Results, Discussions) (less than total 10 pages)
    - List of Source Code

Analytical Solution

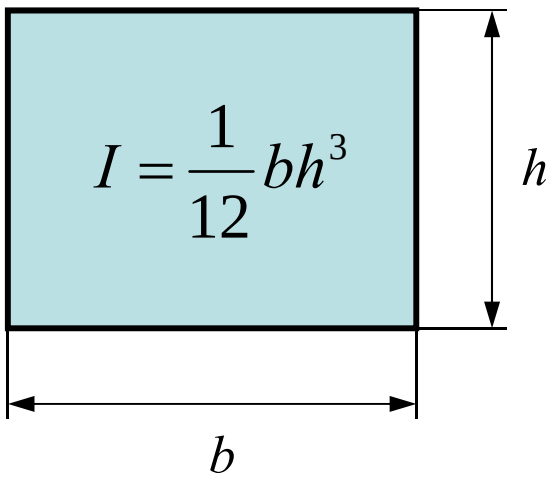
if “L” is sufficiently Large

$$z = -\frac{W}{24EI}(x-L)^2 \cdot \left(x^2 + 2Lx + 3L^2\right)$$

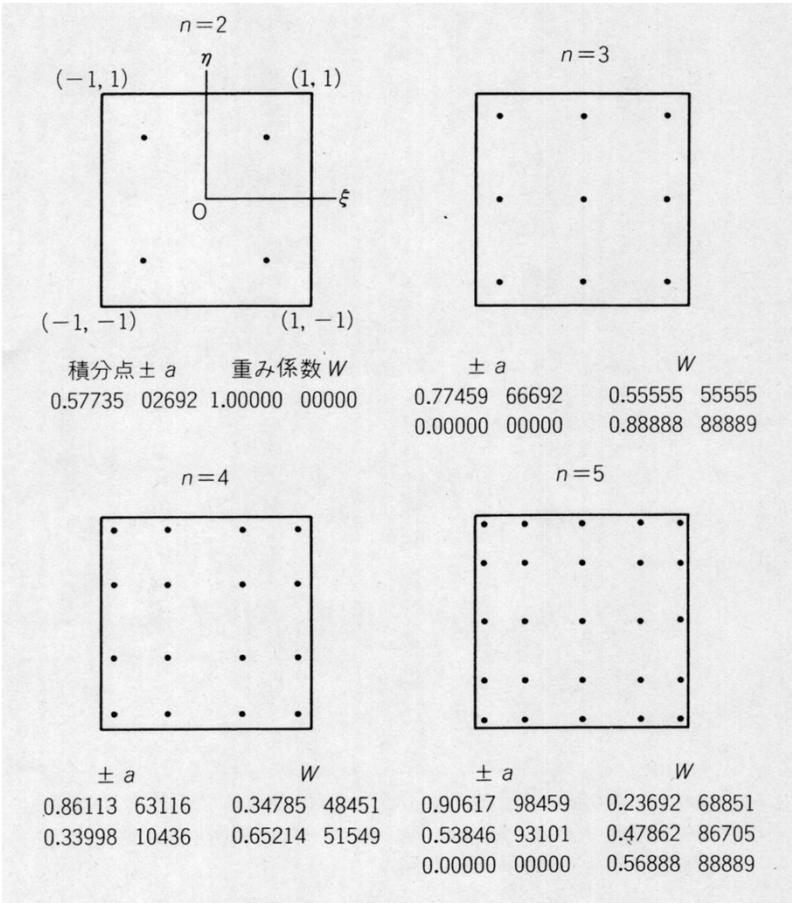
$$z_{\max} = -\frac{WL^4}{8EI} \quad where \quad \frac{L}{h} > 4$$



- z disp. in z-direction
- L Length of Beam
- W Density per Length
- E Young’s Modulus
- I Geometrical Moment of Inertia (for rectangular section)



Gaussian Quadrature



<u>n=1</u>

Quadrature Point: 0.000
Weighting Factor: 2.000

At Center of the Element

Body Force in Z-Direction

$$\frac{\partial \tau_{zx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \sigma_z}{\partial z} + Z = 0$$

$$\tau_{zx,x} + \tau_{xy,y} + \sigma_{z,z} + Z = 0$$

Galerkin Method

$$\int [N]^T \{\tau_{zx,x} + \tau_{xy,y} + \sigma_{z,z} + Z\} dV = 0$$

$$\int [N]^T \{\tau_{zx,x}\} dV + \int [N]^T \{\tau_{xy,y}\} dV + \int [N]^T \{\sigma_{z,z}\} dV + \underline{\int [N]^T \{Z\} dV} = 0$$