

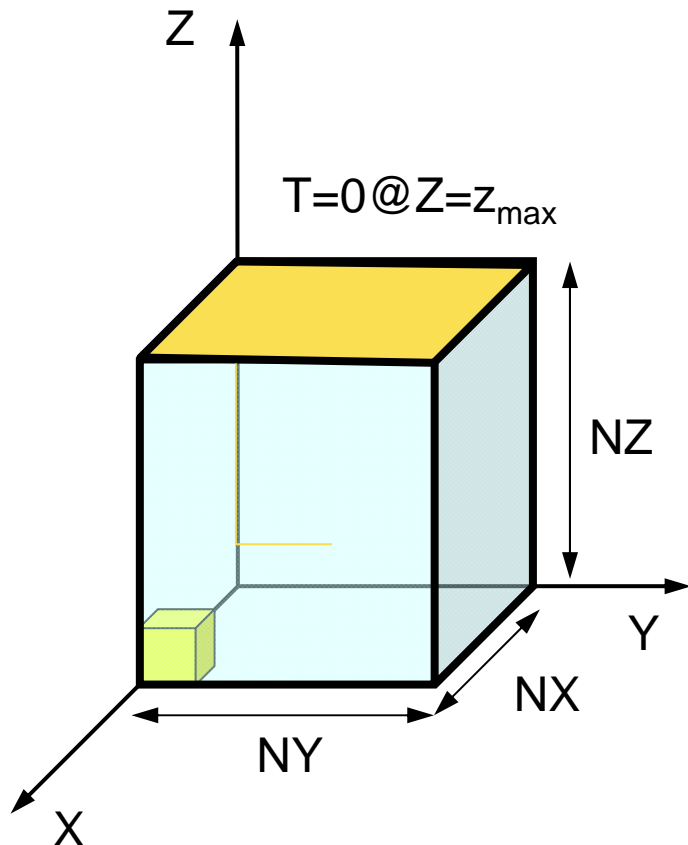
# **3D-FEM in Fortran: Steady State Heat Conduction**

Kengo Nakajima  
Information Technology Center

Programming for Parallel Computing (616-2057)  
Seminar on Advanced Computing (616-4009)

# 3D Steady-State Heat Conduction

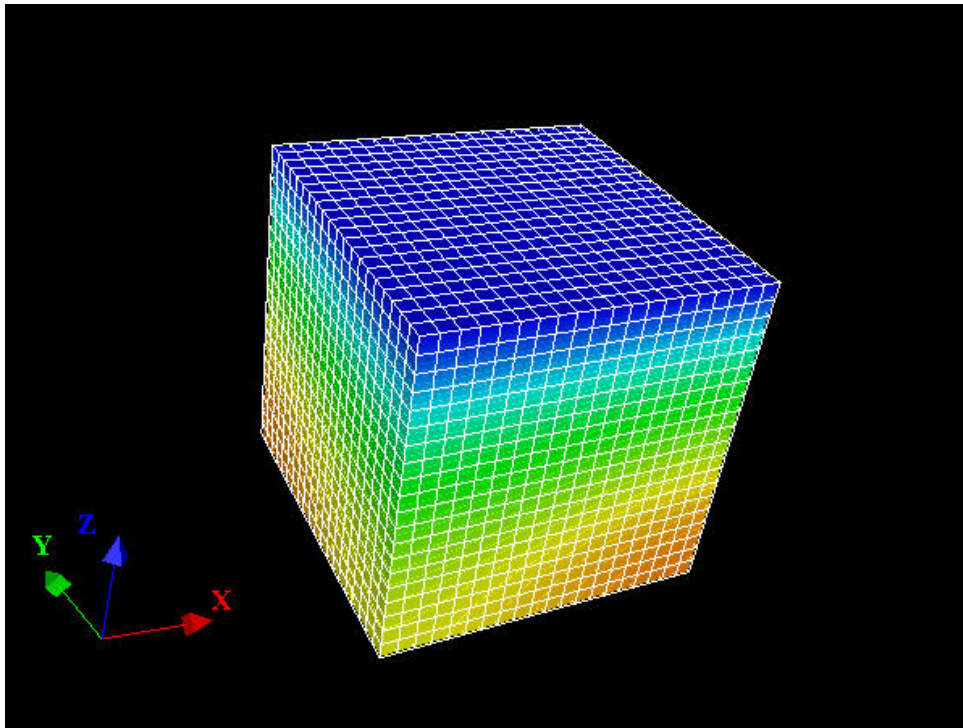
$$\frac{\partial}{\partial x} \left( \lambda \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left( \lambda \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left( \lambda \frac{\partial T}{\partial z} \right) + \dot{Q}(x, y, z) = 0$$



- Heat Generation
- Uniform thermal conductivity  $\lambda$
- HEX meshes
  - 1x1x1 cubes
  - NX, NY, NZ cubes in each direction
- Boundary Conditions
  - T=0@Z=z<sub>max</sub>
- Heat Gen. Rate is a function of location (cell center:  $x_c, y_c$ )
  - $\dot{Q}(x, y, z) = QVOL|x_c + y_c|$

# 3D Steady-State Heat Conduction

$$\frac{\partial}{\partial x} \left( \lambda \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left( \lambda \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left( \lambda \frac{\partial T}{\partial z} \right) + \dot{Q}(x, y, z) = 0$$



- Higher temperature at nodes far from the origin.
- Heat Gen. Rate is a function of location (cell center:  $x_c, y_c$ )

$$\dot{Q}(x, y, z) = |x_c + y_c|$$

movie

# Finite-Element Procedures

- Governing Equations
- Galerkin Method: Weak Form
- Element-by-Element Integration
  - Element Matrix
- Global Matrix
- Boundary Conditions
- Linear Solver

# FEM Procedures: Program

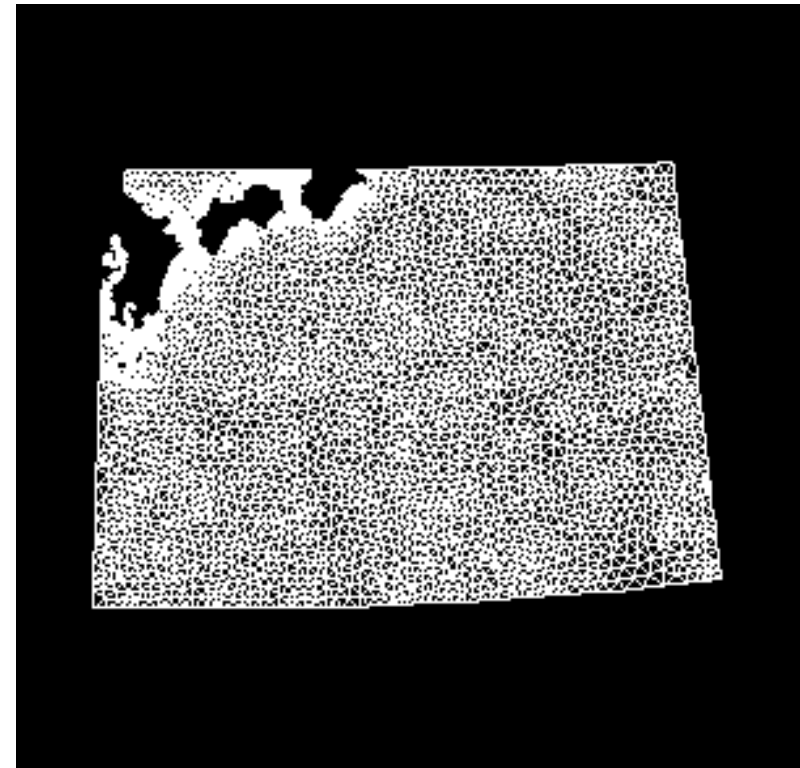
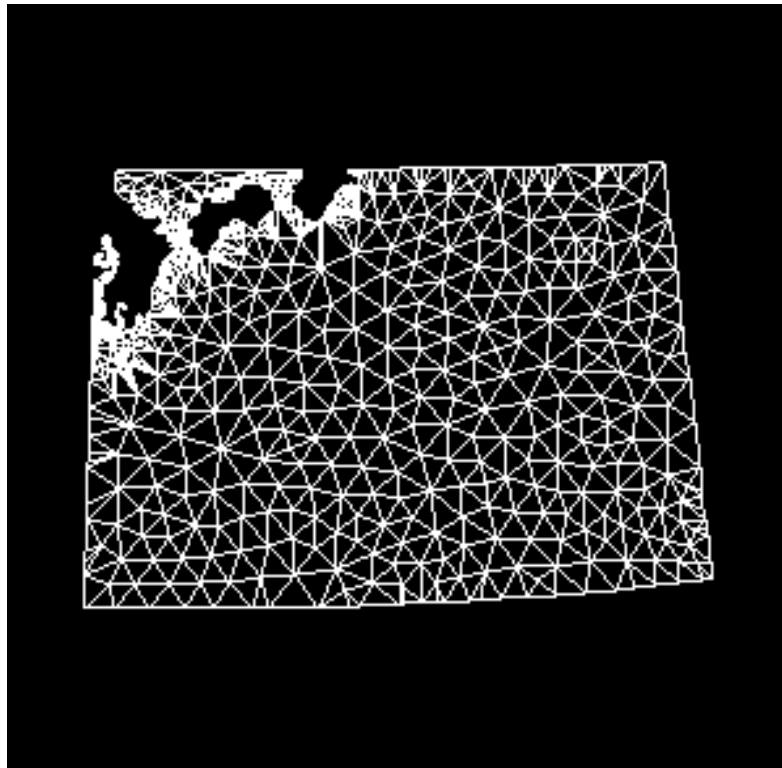
- Initialization
  - Control Data
  - Node, Connectivity of Elements (N: Node#, NE: Elem#)
  - Initialization of Arrays (Global/Element Matrices)
  - Element-Global Matrix Mapping (Index, Item)
- Generation of Matrix
  - Element-by-Element Operations (do icel= 1, NE)
    - Element matrices
    - Accumulation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

- Formulation of 3D Element
- 3D Heat Equations
  - Galerkin Method
  - Element Matrices
- Running the Code
- Data Structure
- Overview of the Program

# Extension to 2D Prob.: Triangles

## 三角形要素

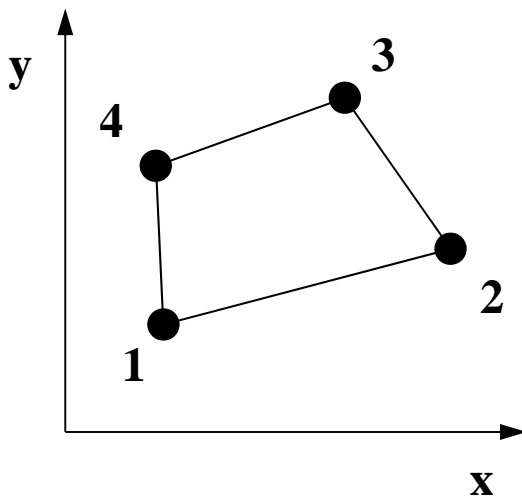
- Triangles can handle arbitrarily shaped object
- “Linear” triangular elements provide low accuracy, therefore they are not used in practical applications.



# Extension to 2D Prob.: Quadrilaterals

## 四角形要素

- Formulation of quad. elements is possible if same shape functions in 1D elements are applied along X- and Y- axis.
  - More accurate than triangles
- Each edge must be “parallel” with X- and Y- axis.
  - Similar to FDM

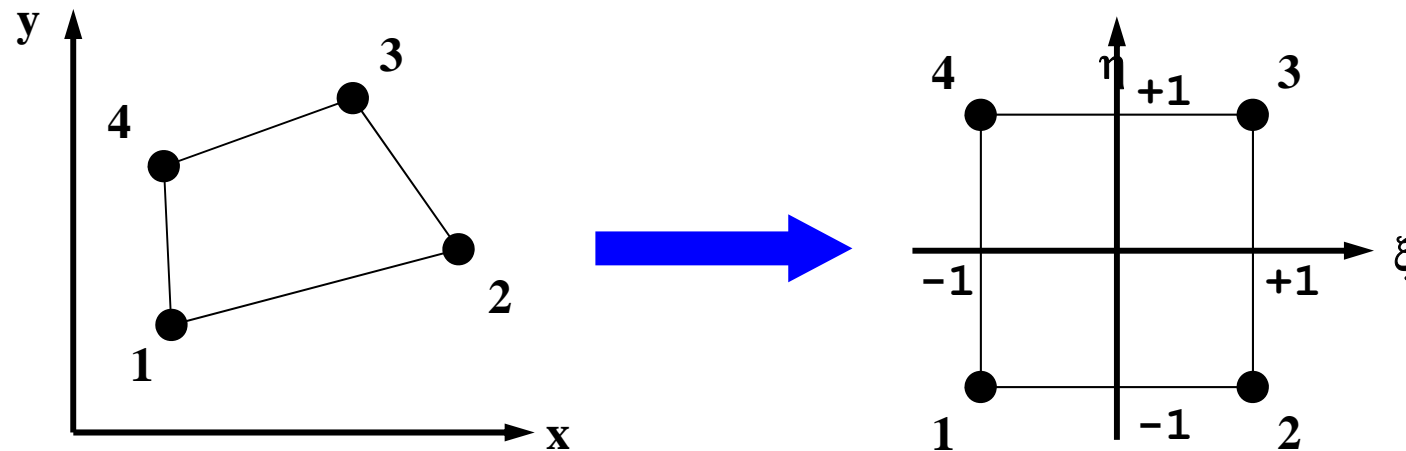


- This type of elements cannot be considered.



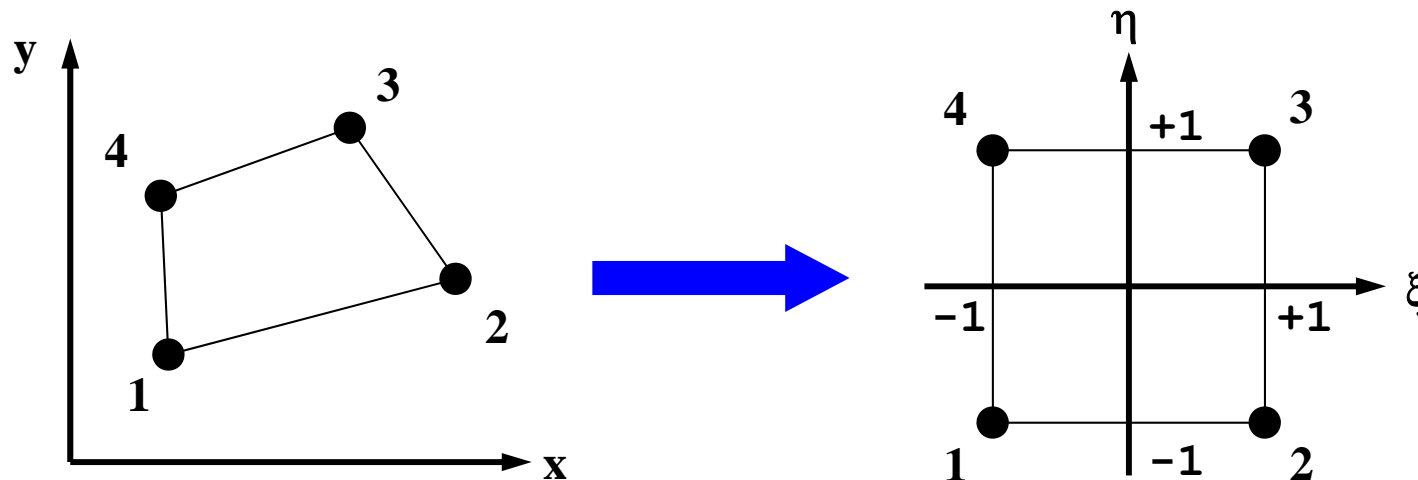
# Isoparametric Element (1/3)

- Each element is mapped to square element  $[\pm 1, \pm 1]$  on natural/local coordinate  $(\xi, \eta)$



- Components of global coordinate system of each node  $(x, y)$  for certain kinds of elements are defined by shape functions  $[N]$  on natural/local coordinate system, where shape functions  $[N]$  are also used for interpolation of dependent variables.

# Isoparametric Element (2/3)

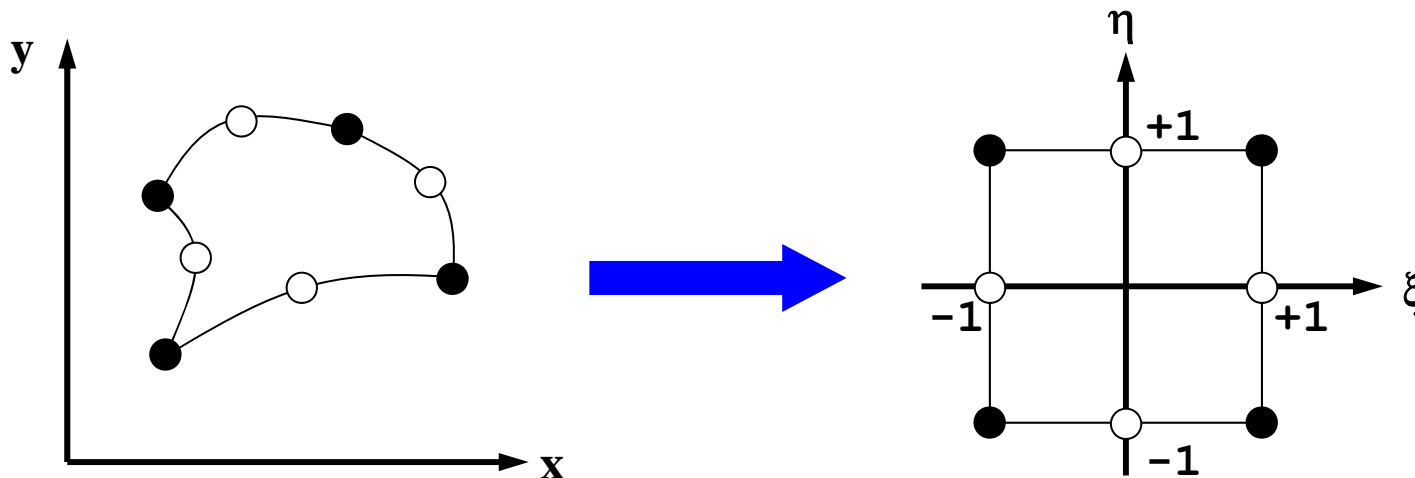


- Coordinate of each node:  $(x_1, y_1)$ ,  $(x_2, y_2)$ ,  $(x_3, y_3)$ ,  $(x_4, y_4)$
- Temperature at each node:  $T_1$ ,  $T_2$ ,  $T_3$ ,  $T_4$

$$T = \sum_{i=1}^4 N_i(\xi, \eta) \cdot T_i$$

$$x = \sum_{i=1}^4 N_i(\xi, \eta) \cdot x_i, \quad y = \sum_{i=1}^4 N_i(\xi, \eta) \cdot y_i$$

# Isoparametric Element (3/3)



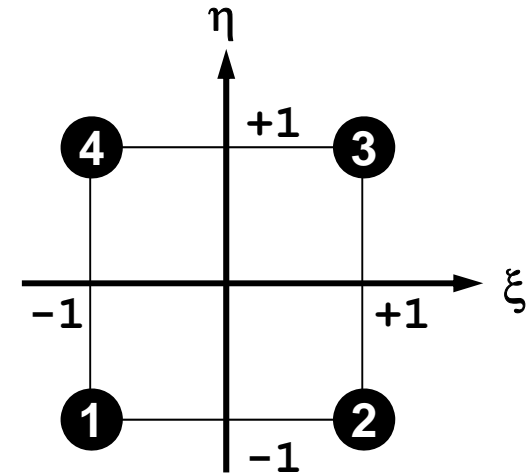
- Higher-order shape function can handle curved lines/surfaces.
- “Natural” coordinate system

Sub-Parametric  
Super-Parametric

# Shape Fn's on 2D Natural Coord. (1/3)

- Polynomial shape functions on squares of natural coordinate:

$$T = \alpha_1 + \alpha_2 \xi + \alpha_3 \eta + \alpha_4 \xi \eta$$



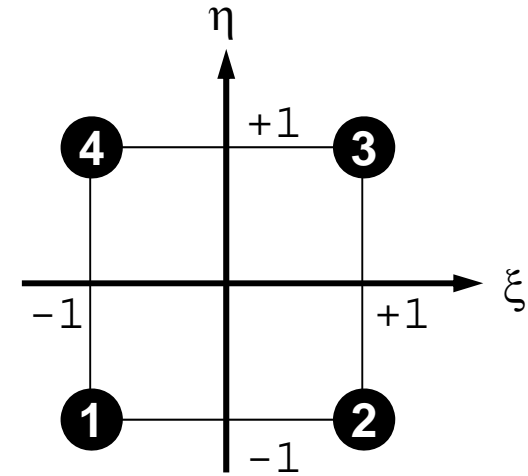
- Coefficients are calculated as follows:

$$\alpha_1 = \frac{T_1 + T_2 + T_3 + T_4}{4}, \quad \alpha_2 = \frac{-T_1 + T_2 + T_3 - T_4}{4},$$

$$\alpha_3 = \frac{-T_1 - T_2 + T_3 + T_4}{4}, \quad \alpha_4 = \frac{T_1 - T_2 + T_3 - T_4}{4}$$

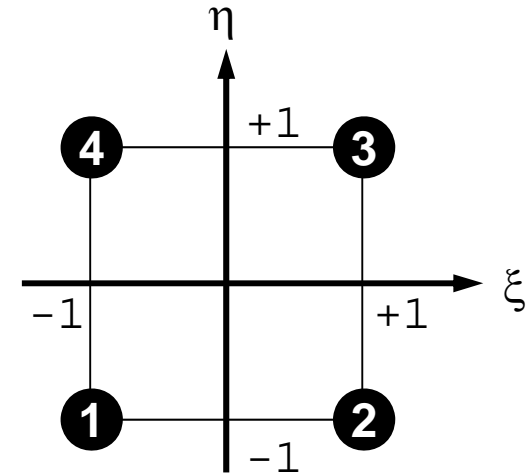
# Shape Fn's on 2D Natural Coord. (2/3)

$$\begin{aligned}
 T &= \alpha_1 + \alpha_2 \xi + \alpha_3 \eta + \alpha_4 \xi \eta \\
 &= \frac{T_1 + T_2 + T_3 + T_4}{4} + \frac{-T_1 + T_2 + T_3 - T_4}{4} \xi + \\
 &\quad \frac{-T_1 - T_2 + T_3 + T_4}{4} \eta + \frac{T_1 - T_2 + T_3 - T_4}{4} \xi \eta \\
 &= \frac{1}{4} (1 - \xi - \eta + \xi \eta) T_1 + \frac{1}{4} (1 + \xi - \eta - \xi \eta) T_2 + \\
 &\quad \frac{1}{4} (1 + \xi + \eta + \xi \eta) T_3 + \frac{1}{4} (1 - \xi + \eta - \xi \eta) T_4 \\
 &= \frac{1}{4} (1 - \xi)(1 - \eta) T_1 + \frac{1}{4} (1 + \xi)(1 - \eta) T_2 + \\
 &\quad \frac{1}{4} (1 + \xi)(1 + \eta) T_3 + \frac{1}{4} (1 - \xi)(1 + \eta) T_4
 \end{aligned}$$



# Shape Fn's on 2D Natural Coord. (2/3)

$$\begin{aligned}
 T &= \alpha_1 + \alpha_2 \xi + \alpha_3 \eta + \alpha_4 \xi \eta \\
 &= \frac{T_1 + T_2 + T_3 + T_4}{4} + \frac{-T_1 + T_2 + T_3 - T_4}{4} \xi + \\
 &\quad \frac{-T_1 - T_2 + T_3 + T_4}{4} \eta + \frac{T_1 - T_2 + T_3 - T_4}{4} \xi \eta \\
 &= \frac{1}{4} (1 - \xi - \eta + \xi \eta) T_1 + \frac{1}{4} (1 + \xi - \eta - \xi \eta) T_2 + \\
 &\quad \frac{1}{4} (1 + \xi + \eta + \xi \eta) T_3 + \frac{1}{4} (1 - \xi + \eta - \xi \eta) T_4 \\
 &= \boxed{\frac{1}{4} (1 - \xi)(1 - \eta)} T_1 + \boxed{\frac{1}{4} (1 + \xi)(1 - \eta)} T_2 + \\
 &\quad \boxed{\frac{1}{4} (1 + \xi)(1 + \eta)} T_3 + \boxed{\frac{1}{4} (1 - \xi)(1 + \eta)} T_4 \\
 &= N_1 T_1 + N_2 T_2 + N_3 T_3 + N_4 T_4
 \end{aligned}$$



# Shape Fn's on 2D Natural Coord. (3/3)

- $T$  is defined as follows according to  $T_i$  :

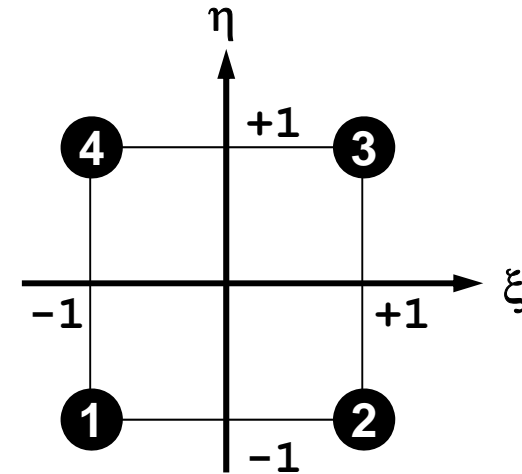
$$T = N_1 T_1 + N_2 T_2 + N_3 T_3 + N_4 T_4$$

- Shape functions  $N_i$  :

$$N_1(\xi, \eta) = \frac{1}{4}(1 - \xi)(1 - \eta), \quad N_2(\xi, \eta) = \frac{1}{4}(1 + \xi)(1 - \eta),$$

$$N_3(\xi, \eta) = \frac{1}{4}(1 + \xi)(1 + \eta), \quad N_4(\xi, \eta) = \frac{1}{4}(1 - \xi)(1 + \eta)$$

- Also known as “bi-linear” interpolation
- Calculate  $N_i$  at each node



# Extension to 3D Problems

- Tetrahedron/Tetrahedra (四面体) : Triangles in 2D
  - can handle arbitrary shape objects
  - Linear elements are generally less accurate, not practical
  - Higher-order tetrahedral elements are widely used.
- In this class, “tri-linear” hexahedral elements (isoparametric) are used (六面体要素)



# Shape Fn's: 3D Natural/Local Coord.

$$N_1(\xi, \eta, \zeta) = \frac{1}{8}(1-\xi)(1-\eta)(1-\zeta) \quad N_5(\xi, \eta, \zeta) = \frac{1}{8}(1-\xi)(1-\eta)(1+\zeta)$$

$$N_2(\xi, \eta, \zeta) = \frac{1}{8}(1+\xi)(1-\eta)(1-\zeta) \quad N_6(\xi, \eta, \zeta) = \frac{1}{8}(1+\xi)(1-\eta)(1+\zeta)$$

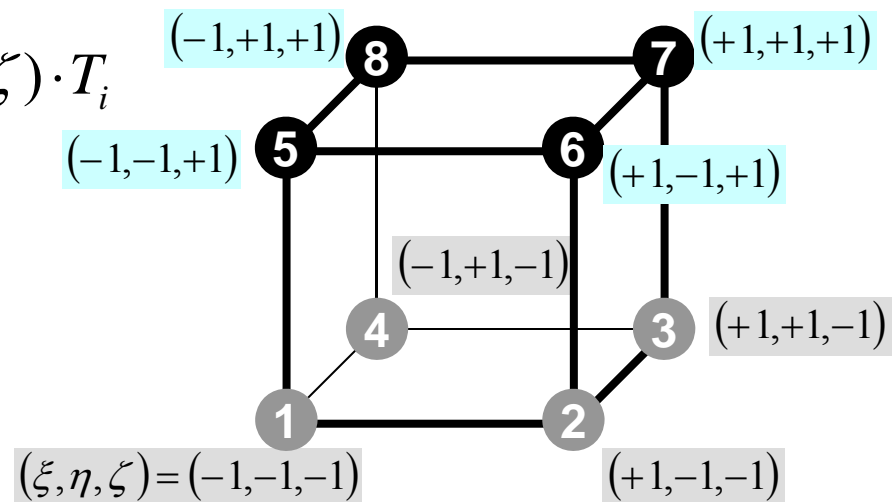
$$N_3(\xi, \eta, \zeta) = \frac{1}{8}(1+\xi)(1+\eta)(1-\zeta) \quad N_7(\xi, \eta, \zeta) = \frac{1}{8}(1+\xi)(1+\eta)(1+\zeta)$$

$$N_4(\xi, \eta, \zeta) = \frac{1}{8}(1-\xi)(1+\eta)(1-\zeta) \quad N_8(\xi, \eta, \zeta) = \frac{1}{8}(1-\xi)(1+\eta)(1+\zeta)$$

$$x = \sum_{i=1}^8 N_i(\xi, \eta, \zeta) \cdot x_i, \quad T = \sum_{i=1}^8 N_i(\xi, \eta, \zeta) \cdot T_i$$

$$y = \sum_{i=1}^8 N_i(\xi, \eta, \zeta) \cdot y_i$$

$$z = \sum_{i=1}^8 N_i(\xi, \eta, \zeta) \cdot z_i$$



- Formulation of 3D Element
- **3D Heat Equations**
  - **Galerkin Method**
  - **Element Matrices**
- Running the Code
- Data Structure
- Overview of the Program

# Galerkin Method (1/3)

- Governing Equation for 3D Steady State Heat Conduction Problems (uniform  $\lambda$ ):

$$\left( \lambda \frac{\partial^2 T}{\partial x^2} \right) + \left( \lambda \frac{\partial^2 T}{\partial y^2} \right) + \left( \lambda \frac{\partial^2 T}{\partial z^2} \right) + \dot{Q} = 0$$

$$T = [N]\{\phi\} \quad \text{Distribution of temperature in each element (matrix form), } \phi: \text{ Temperature at each node}$$

- Following integral equation is obtained at each element by Galerkin method, where  $[N]$ 's are also weighting functions:

$$\int_V [N]^T \left\{ \lambda \left( \frac{\partial^2 T}{\partial x^2} \right) + \lambda \left( \frac{\partial^2 T}{\partial y^2} \right) + \lambda \left( \frac{\partial^2 T}{\partial z^2} \right) + \dot{Q} \right\} dV = 0$$

# Galerkin Method (2/3)

- Green's Theorem (3D)

$$\int_V A \left( \frac{\partial^2 B}{\partial x^2} + \frac{\partial^2 B}{\partial y^2} + \frac{\partial^2 B}{\partial z^2} \right) dV = \int_S A \frac{\partial B}{\partial n} dS - \int_V \left( \frac{\partial A}{\partial x} \frac{\partial B}{\partial x} + \frac{\partial A}{\partial y} \frac{\partial B}{\partial y} + \frac{\partial A}{\partial z} \frac{\partial B}{\partial z} \right) dV$$

- Apply this to the 1st 3-parts of the equation with 2<sup>nd</sup>-order diff. (surface integration terms are ignored):

$$\begin{aligned} & \int_V [N]^T \{ \lambda(T_{,xx}) + \lambda(T_{,yy}) + \lambda(T_{,zz}) \} dV \\ &= - \int_V \{ \lambda([N_{,x}]^T T_{,x}) + \lambda([N_{,y}]^T T_{,y}) + \lambda([N_{,z}]^T T_{,z}) \} dV \end{aligned}$$

- Consider the following terms:

$$T = [N]\{\phi\}, \quad T_{,x} = [N_{,x}]\{\phi\}, \quad T_{,y} = [N_{,y}]\{\phi\}, \quad T_{,z} = [N_{,z}]\{\phi\}$$

## Galerkin Method (3/3)

- Finally, following equation is obtained by considering heat generation term  $\dot{Q}$  :

$$-\int_V \left\{ \lambda ([N_{,x}]^T [N_{,x}]) + \lambda ([N_{,y}]^T [N_{,y}]) + \lambda ([N_{,z}]^T [N_{,z}]) \right\} dV \cdot \{\phi\} + \int_V \dot{Q} [N] dV = 0$$

- This is called “weak form (弱形式)”. Original PDE consists of terms with 2<sup>nd</sup>-order diff., but this “weak form” only includes 1<sup>st</sup>-order diff by Green’s theorem.
  - Requirements for shape functions are “weaker” in “weak form”. Linear functions can describe effects of 2<sup>nd</sup>-order differentiation.
  - Same as 1D problem

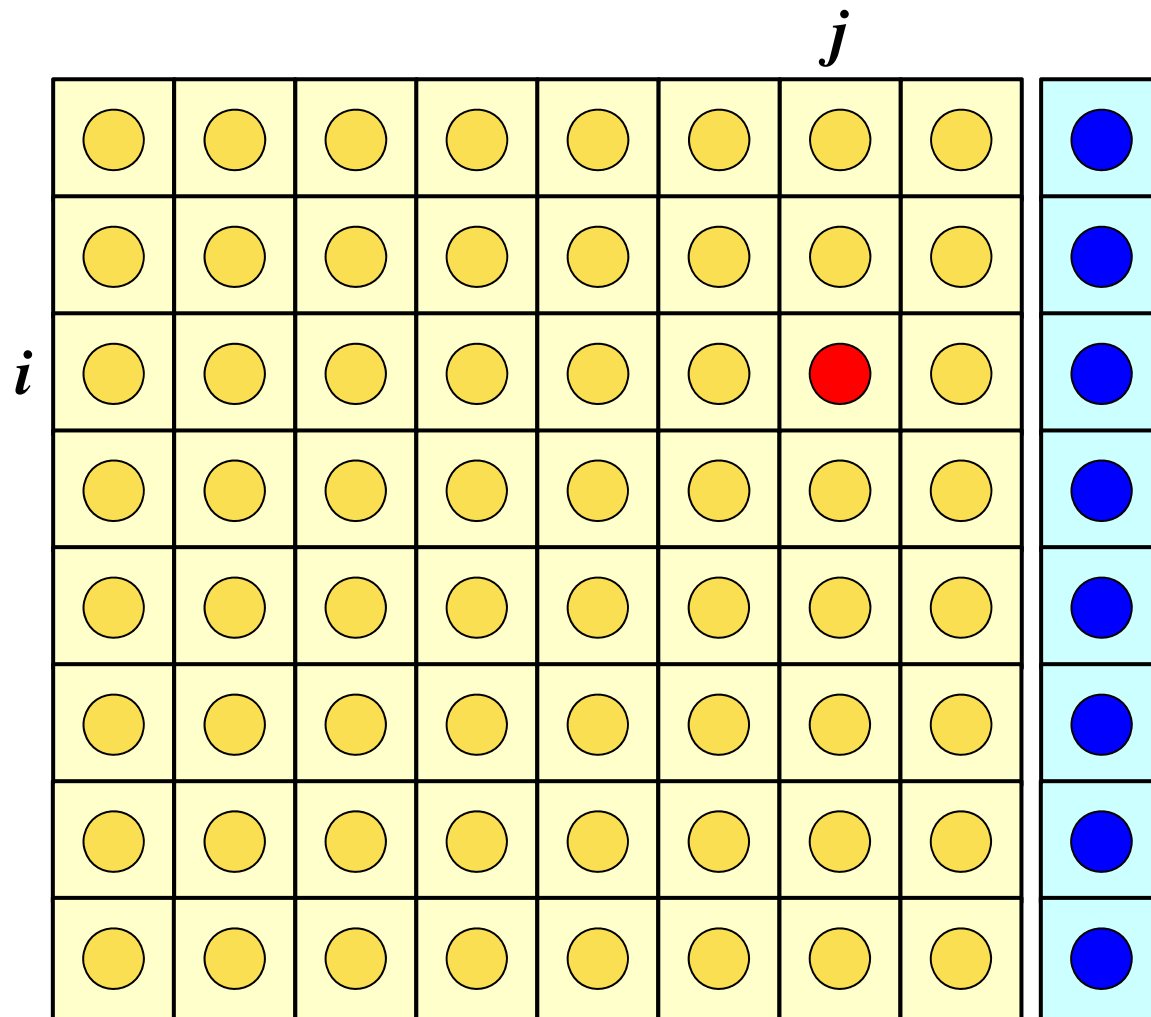
Weak Form with B.C.: on each elem.

$$[k]^{(e)} \{\phi\}^{(e)} = \{f\}^{(e)}$$

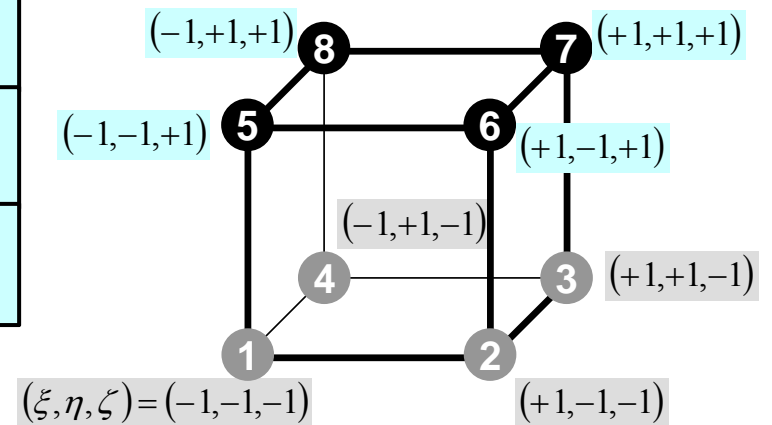
$$[k]^{(e)} = \int_V \lambda \left( [N_{,x}]^T [N_{,x}] \right) dV + \int_V \lambda \left( [N_{,y}]^T [N_{,y}] \right) dV \\ + \int_V \lambda \left( [N_{,z}]^T [N_{,z}] \right) dV$$

$$\{f\}^{(e)} = \int_V \dot{Q} [N]^T dV$$

# Element Matrix: 8x8

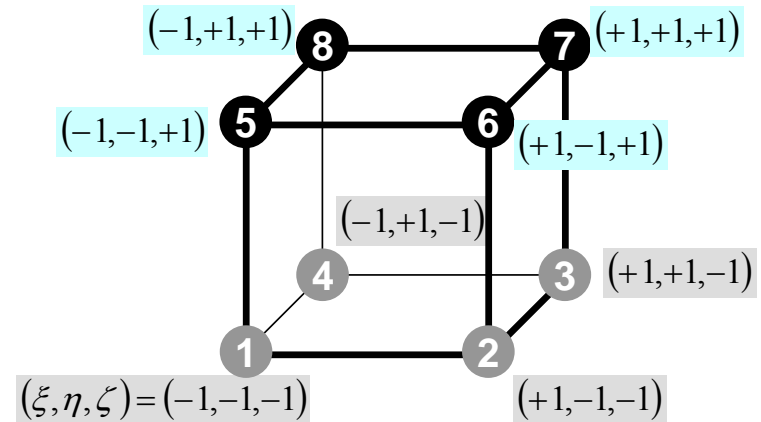


$$[k_{ij}] \quad (i, j = 1 \dots 8)$$



# Element Matrix: $k_{ij}$

$$[k_{ij}] \quad (i, j = 1 \dots 8)$$



$$[k]^{(e)} = \int_V \lambda ([N_{,x}]^T [N_{,x}]) dV + \int_V \lambda ([N_{,y}]^T [N_{,y}]) dV + \int_V \lambda ([N_{,z}]^T [N_{,z}]) dV$$



$$k_{ij} = - \int_V \{ \lambda \cdot N_{i,x} \cdot N_{j,x} + \lambda \cdot N_{i,y} \cdot N_{j,y} + \lambda \cdot N_{i,z} \cdot N_{j,z} \} dV$$

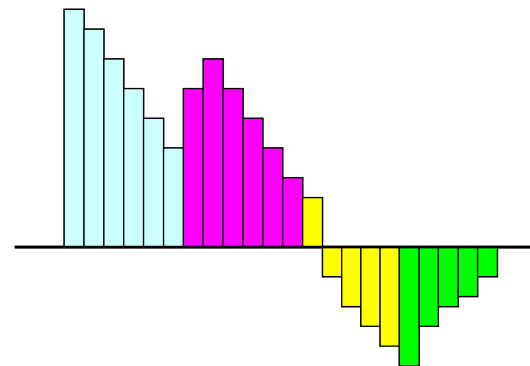


# Next Stage: Integration

# Methods for Numerical Integration

- Trapezoidal Rule
- Simpson's Rule
- Gaussian Quadrature (or Gauss-Legendre)
  - accurate
  
- Values of functions at finite numbers of sample points are utilized:

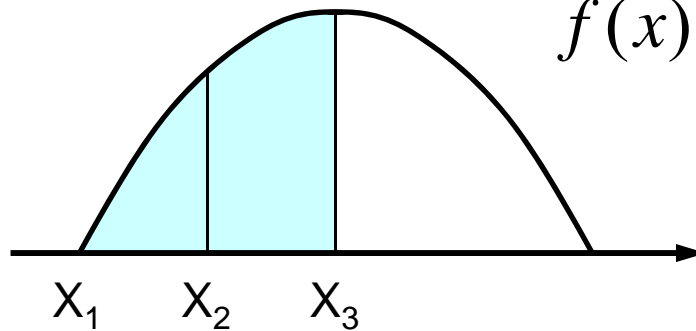
$$\int_{X_1}^{X_2} f(x) dx \Rightarrow \sum_{k=1}^m [w_k \cdot f(x_k)]$$



# Gaussian Quadrature in 1D

more accurate than Simpson's rule

**Simpson's**

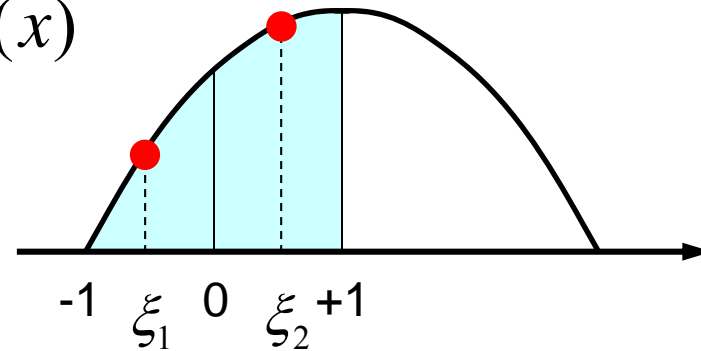


$$X_1 = 0, \quad X_2 = \frac{\pi}{4}, \quad X_3 = \frac{\pi}{2}$$

$$h = X_2 - X_1 = X_3 - X_2 = \frac{\pi}{4}$$

$$S = \frac{h}{3} [f(X_1) + 4f(X_2) + f(X_3)] = 1.0023$$

**Gauss**



$$\xi_1, \xi_2 = \pm 0.5773502692$$

$$S = \int_0^{\pi/2} f(x) dx = \int_{-1}^{+1} f(\xi) h d\xi$$

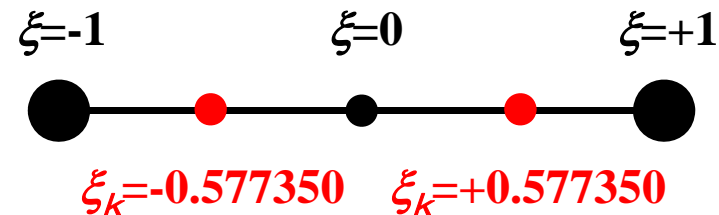
$$\cong h \sum_{k=1}^2 W_k \cdot f(\xi_k) = 0.99847$$

# Gaussian Quadrature

## ガウスの積分公式

- On normalized “natural (or local)” coordinate system  $[-1,+1]$  (自然座標系, 局所座標系)
- Can approximate up to  $(2m-1)$ -th order of functions by  $m$  quadrature points ( $m=2$  is enough for quadratic shape functions).

$$\int_{-1}^{+1} f(\xi) d\xi = \sum_{k=1}^m [w_k \cdot f(\xi_k)]$$



$$m = 1 \quad \xi_k = 0.00, w_k = 2.00$$

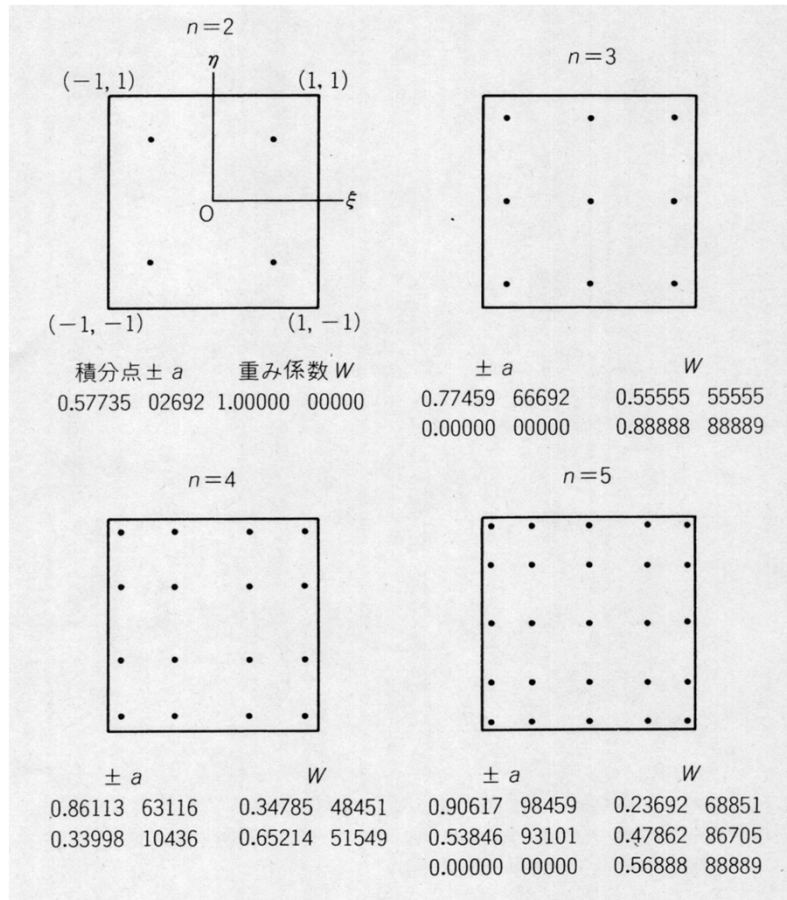
$$m = 2 \quad \xi_k = \pm 0.577350, w_k = 1.00$$

$$m = 3 \quad \xi_k = 0.00, w_k = 8/9$$

$$\xi_k = \pm 0.774597, w_k = 5/9$$

# Gaussian Quadrature

can be easily extended to 2D & 3D



$$I = \int_{-1}^{+1} \int_{-1}^{+1} f(\xi, \eta) d\xi d\eta$$

$$= \sum_{i=1}^m \sum_{j=1}^n [W_i \cdot W_j \cdot f(\xi_i, \eta_j)]$$

$m, n$ : number of quadrature points in  $\xi, \eta$ -direction

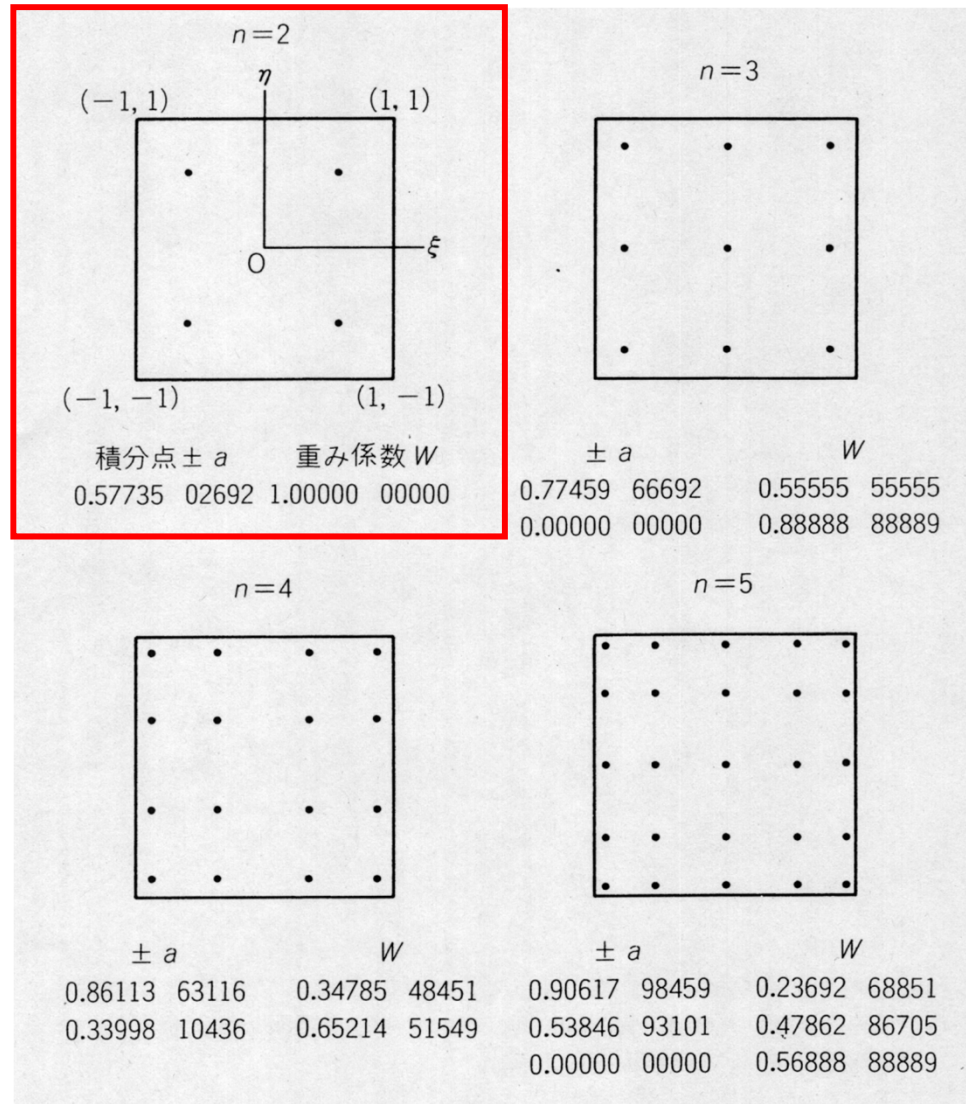
$(\xi_i, \eta_j)$ : Coordinates of Quad's

$W_i, W_j$ : Weighting Factor

# Gaussian Quadrature

## ガウスの積分公式

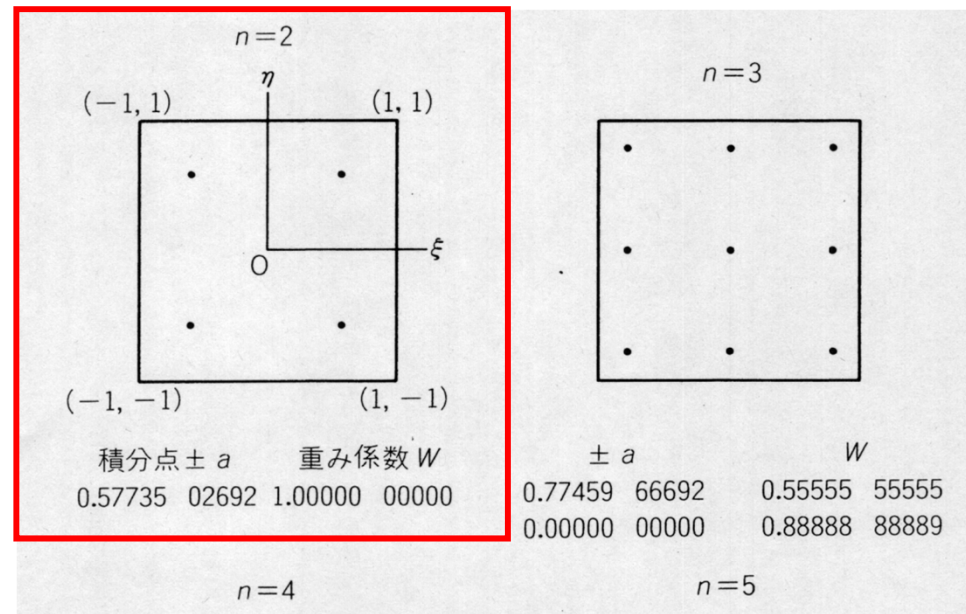
This configuration is widely used. In 2D problem, integration is done using values of “f” at 4 quad. points.



# Gaussian Quadrature

## ガウスの積分公式

This configuration is widely used. In 2D problem, integration is done using values of “f” at 4 quad. points.



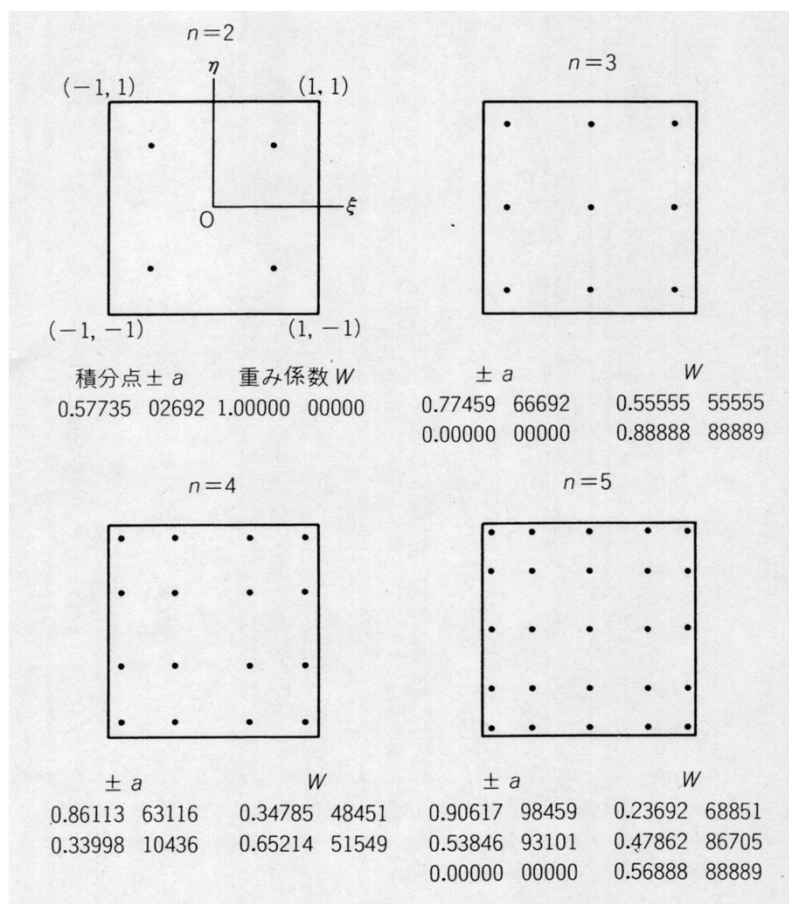
$$I = \int_{-1}^{+1} \int_{-1}^{+1} f(\xi, \eta) d\xi d\eta = \sum_{i=1}^m \sum_{j=1}^n [W_i \cdot W_j \cdot f(\xi_i, \eta_j)]$$

$$= 1.0 \times 1.0 \times f(-0.57735, -0.57735) + 1.0 \times 1.0 \times f(-0.57735, +0.57735) \\ + 1.0 \times 1.0 \times f(+0.57735, +0.57735) + 1.0 \times 1.0 \times f(+0.57735, -0.57735)$$

0.33998 10436 0.05214 51549 0.33846 55101 0.47802 00703  
0.00000 00000 0.56888 88889

# Next Stage: Integration

- 3D Natural/Local Coordinate  $(\xi, \eta, \zeta)$ :
  - Gaussian Quadrature



$$I = \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} f(\xi, \eta, \zeta) d\xi d\eta d\zeta$$

$$= \sum_{i=1}^L \sum_{j=1}^M \sum_{k=1}^N [W_i \cdot W_j \cdot W_k \cdot f(\xi_i, \eta_j, \zeta_k)]$$

$L, M, N$ : number of quadrature points in  $\xi, \eta, \zeta$ -direction

$(\xi_i, \eta_j, \zeta_k)$  : Coordinates of Quad's

$W_i, W_j, W_k$  : Weighting Factor



# Partial Diff. on Natural Coord. (1/4)

- According to formulae:

$$\frac{\partial N_i(\xi, \eta, \zeta)}{\partial \xi} = \frac{\partial N_i}{\partial x} \frac{\partial x}{\partial \xi} + \frac{\partial N_i}{\partial y} \frac{\partial y}{\partial \xi} + \frac{\partial N_i}{\partial z} \frac{\partial z}{\partial \xi}$$

$$\frac{\partial N_i(\xi, \eta, \zeta)}{\partial \eta} = \frac{\partial N_i}{\partial x} \frac{\partial x}{\partial \eta} + \frac{\partial N_i}{\partial y} \frac{\partial y}{\partial \eta} + \frac{\partial N_i}{\partial z} \frac{\partial z}{\partial \eta}$$

$$\frac{\partial N_i(\xi, \eta, \zeta)}{\partial \zeta} = \frac{\partial N_i}{\partial x} \frac{\partial x}{\partial \zeta} + \frac{\partial N_i}{\partial y} \frac{\partial y}{\partial \zeta} + \frac{\partial N_i}{\partial z} \frac{\partial z}{\partial \zeta}$$

$\left[ \frac{\partial N_i}{\partial \xi}, \frac{\partial N_i}{\partial \eta}, \frac{\partial N_i}{\partial \zeta} \right]$  can be easily derived according to definitions.

$\left[ \frac{\partial N_i}{\partial x}, \frac{\partial N_i}{\partial y}, \frac{\partial N_i}{\partial z} \right]$  are required for computations.

# Partial Diff. on Natural Coord. (2/4)

- In matrix form:

$$\left\{ \begin{array}{c} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \zeta} \end{array} \right\} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \left\{ \begin{array}{c} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{array} \right\} = [J] \left\{ \begin{array}{c} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{array} \right\}$$

$[J]$  : Jacobi matrix, Jacobian

# Partial Diff. on Natural Coord. (3/4)

- Components of Jacobian:

$$J_{11} = \frac{\partial x}{\partial \xi} = \frac{\partial}{\partial \xi} \left( \sum_{i=1}^8 N_i x_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \xi} x_i, \quad J_{12} = \frac{\partial y}{\partial \xi} = \frac{\partial}{\partial \xi} \left( \sum_{i=1}^8 N_i y_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \xi} y_i,$$

$$J_{13} = \frac{\partial z}{\partial \xi} = \frac{\partial}{\partial \xi} \left( \sum_{i=1}^8 N_i z_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \xi} z_i$$

$$J_{21} = \frac{\partial x}{\partial \eta} = \frac{\partial}{\partial \eta} \left( \sum_{i=1}^8 N_i x_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \eta} x_i, \quad J_{22} = \frac{\partial y}{\partial \eta} = \frac{\partial}{\partial \eta} \left( \sum_{i=1}^8 N_i y_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \eta} y_i,$$

$$J_{23} = \frac{\partial z}{\partial \eta} = \frac{\partial}{\partial \eta} \left( \sum_{i=1}^8 N_i z_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \eta} z_i$$

$$J_{31} = \frac{\partial x}{\partial \zeta} = \frac{\partial}{\partial \zeta} \left( \sum_{i=1}^8 N_i x_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \zeta} x_i, \quad J_{32} = \frac{\partial y}{\partial \zeta} = \frac{\partial}{\partial \zeta} \left( \sum_{i=1}^8 N_i y_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \zeta} y_i,$$

$$J_{33} = \frac{\partial z}{\partial \zeta} = \frac{\partial}{\partial \zeta} \left( \sum_{i=1}^8 N_i z_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \zeta} z_i$$

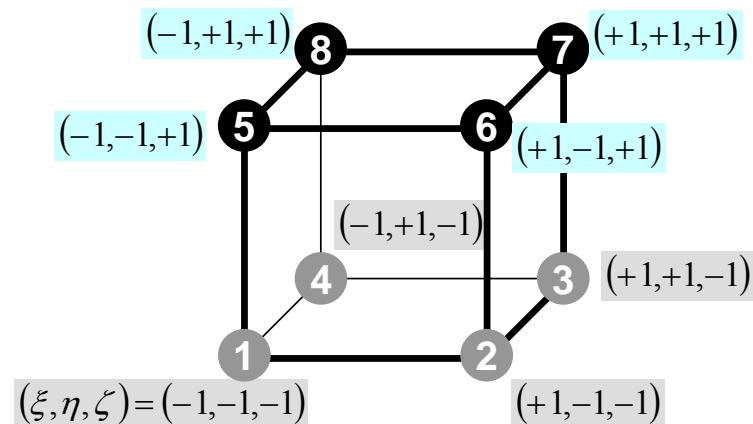
# Partial Diff. on Natural Coord. (4/4)

- Partial differentiation on global coordinate system is introduced as follows (with inverse of Jacobian matrix ( $3 \times 3$ ))

$$\begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix}^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \zeta} \end{Bmatrix} = [J]^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \zeta} \end{Bmatrix}$$

# Integration on Element

$$[k_{ij}] \quad (i, j = 1 \dots 8)$$

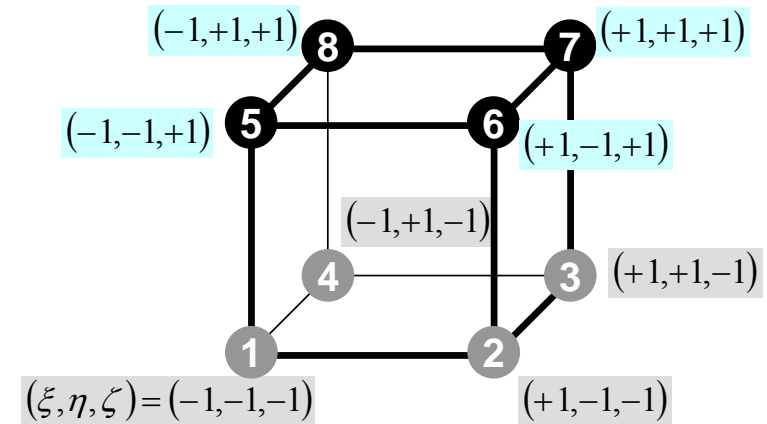


$$k_{ij} = - \int_V \left\{ \lambda \cdot N_{i,x} \cdot N_{j,x} + \lambda \cdot N_{i,y} \cdot N_{j,y} + \lambda \cdot N_{i,z} \cdot N_{j,z} \right\} dV$$

$$= - \int_V \left\{ \lambda \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \lambda \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + \lambda \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right\} dV$$

# Integration on Natural Coord.

$$[k_{ij}] \quad (i, j = 1 \dots 8)$$



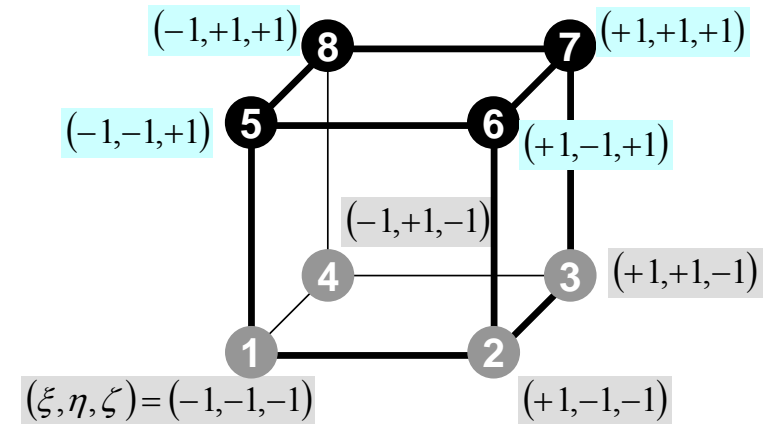
$$- \int_V \left\{ \lambda \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \lambda \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + \lambda \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right\} dV =$$

$$- \iiint \left\{ \lambda \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \lambda \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + \lambda \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right\} dx dy dz =$$

$$- \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} \left\{ \lambda \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \lambda \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + \lambda \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right\} \det |J| d\xi d\eta d\zeta$$

# Gaussian Quadrature

$$[k_{ij}] \quad (i, j = 1 \dots 8)$$



$$- \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} \left\{ \lambda \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \lambda \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + \lambda \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right\} \det |J| d\xi d\eta d\zeta$$

$$I = \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} f(\xi, \eta, \zeta) d\xi d\eta d\zeta$$

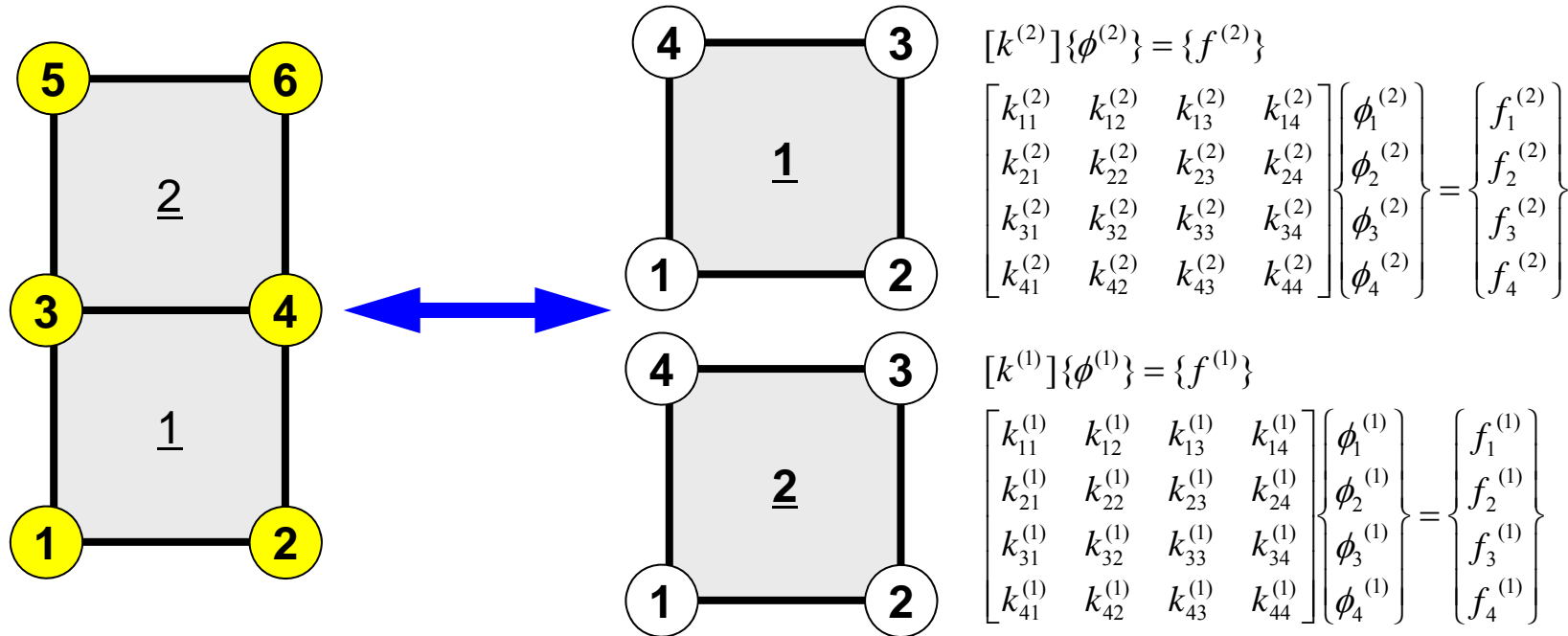
$$= \sum_{i=1}^L \sum_{j=1}^M \sum_{k=1}^N [w_i \cdot w_j \cdot w_k \cdot f(\xi_i, \eta_j, \zeta_k)]$$

# Remaining Procedures

- Element matrices have been formed.
- Accumulation to Global Matrix
- Implementation of Boundary Conditions
- Solving Linear Equations
- Details of implementation will be discussed in classes later than next week through explanation of programs



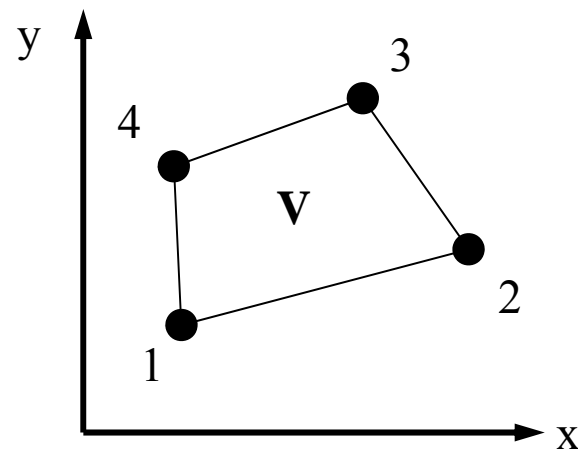
# Accumulation: Local -> Global Matrices



- Formulation of 3D Element
- 3D Heat Equations
  - Galerkin Method
  - Element Matrices
  - **Exercise**
- Running the Code
- Data Structure
- Overview of the Program

# Exercise

- Develop a program and calculate area of the following quadrilateral using Gaussian Quadrature.



1: (1.0, 1.0)  
2: (4.0, 2.0)  
3: (3.0, 5.0)  
4: (2.0, 4.0)

$$I = \int_V dV = \int_{-1}^{+1} \int_{-1}^{+1} \det|J| d\xi d\zeta$$

# Tips (1/2)

- Calculate Jacobian
- Apply Gaussian Quadrature (n=2)

$$I = \int_{-1}^{+1} \int_{-1}^{+1} f(\xi, \eta) d\xi d\eta = \sum_{i=1}^m \sum_{j=1}^n [W_i \cdot W_j \cdot f(\xi_i, \eta_j)]$$

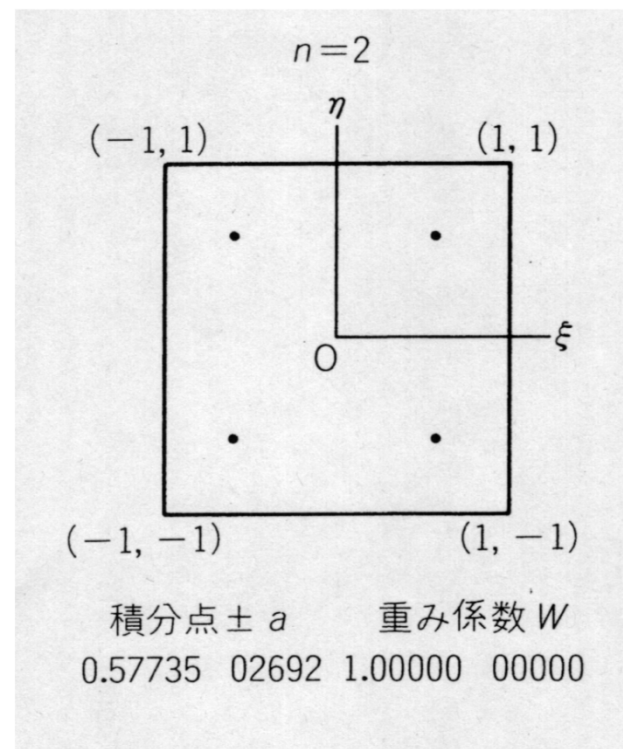
```

implicit REAL*8 (A-H,O-Z)
real*8 W(2)
real*8 POI(2)

W(1)= 1.0d0
W(2)= 1.0d0
POI(1)= -0.5773502692d0
POI(2)= +0.5773502692d0

SUM= 0.d0
do jp= 1, 2
do ip= 1, 2
    FC = F(POI(ip), POI(jp))
    SUM= SUM + W(ip)*W(jp)*FC
enddo
enddo

```



# Tips (2/2)

$$[J] = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix}, \quad \det|J| = \frac{\partial x}{\partial \xi} \cdot \frac{\partial y}{\partial \eta} - \frac{\partial y}{\partial \xi} \cdot \frac{\partial x}{\partial \eta}$$

$$\frac{\partial x}{\partial \xi} = \frac{\partial}{\partial \xi} \left( \sum_{i=1}^4 N_i x_i \right) = \sum_{i=1}^4 \frac{\partial N_i}{\partial \xi} x_i, \quad \frac{\partial y}{\partial \xi} = \frac{\partial}{\partial \xi} \left( \sum_{i=1}^4 N_i y_i \right) = \sum_{i=1}^4 \frac{\partial N_i}{\partial \xi} y_i,$$

$$\frac{\partial x}{\partial \eta} = \frac{\partial}{\partial \eta} \left( \sum_{i=1}^4 N_i x_i \right) = \sum_{i=1}^4 \frac{\partial N_i}{\partial \eta} x_i, \quad \frac{\partial y}{\partial \eta} = \frac{\partial}{\partial \eta} \left( \sum_{i=1}^4 N_i y_i \right) = \sum_{i=1}^4 \frac{\partial N_i}{\partial \eta} y_i$$

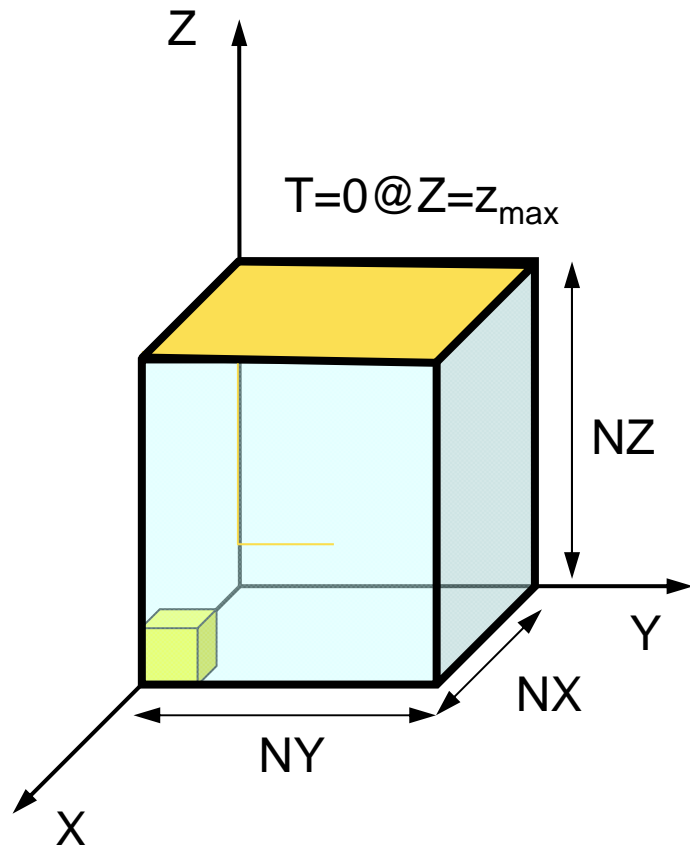
$$N_1(\xi, \eta) = \frac{1}{4}(1 - \xi)(1 - \eta), \quad N_2(\xi, \eta) = \frac{1}{4}(1 + \xi)(1 - \eta),$$

$$N_3(\xi, \eta) = \frac{1}{4}(1 + \xi)(1 + \eta), \quad N_4(\xi, \eta) = \frac{1}{4}(1 - \xi)(1 + \eta)$$

- Formulation of 3D Element
- 3D Heat Equations
  - Galerkin Method
  - Element Matrices
- **Running the Code**
- Data Structure
- Overview of the Program

# 3D Steady-State Heat Conduction

$$\frac{\partial}{\partial x} \left( \lambda \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left( \lambda \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left( \lambda \frac{\partial T}{\partial z} \right) + \dot{Q}(x, y, z) = 0$$



- Heat Generation
- Uniform thermal conductivity  $\lambda$
- HEX meshes
  - 1x1x1 cubes
  - NX, NY, NZ cubes in each direction
- Boundary Conditions
  - T=0@Z=z<sub>max</sub>
- Heat Gen. Rate is a function of location (cell center:  $x_c, y_c$ )
  - $\dot{Q}(x, y, z) = QVOL|x_c + y_c|$

# Copy/Installation

## 3D-FEM Code

```
>$ cd <$E-TOP>
>$ cp /home03/skengon/Documents/class_eps/F/fem3d.tar .
>$ cp /home03/skengon/Documents/class_eps/C/fem3d.tar .
>$ tar xvf fem3d.tar
>$ cd fem3d
>$ ls
    run src
```

## Install

```
>$ cd <$E-TOP>/fem3d/src
>$ make
>$ ls ../run/sol
    sol
```

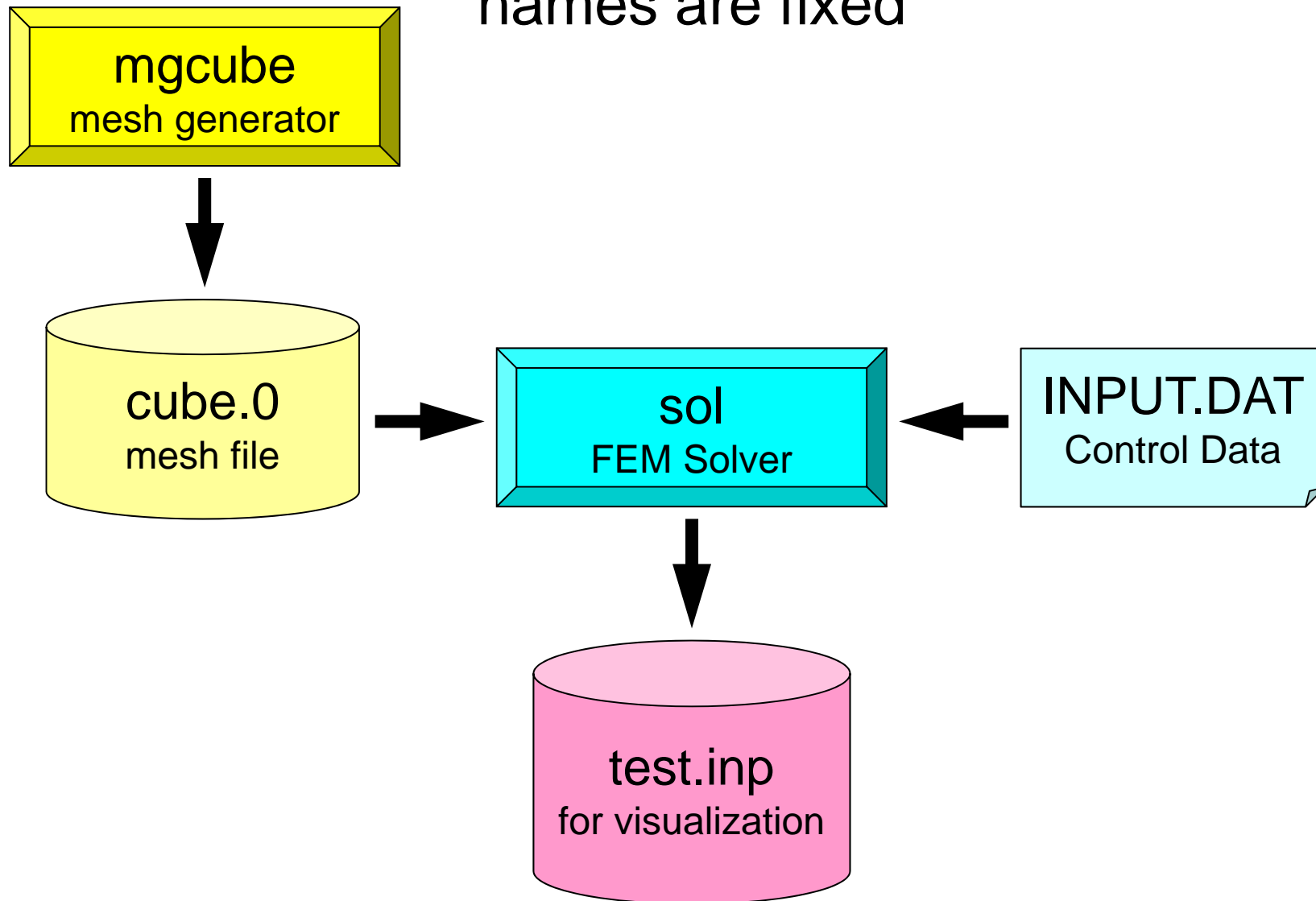
## Install of Mesh Generator

```
>$ cd <$E-TOP>/fem3d/run
>$ g95 -O3 mgcube.f -o mgcube
```

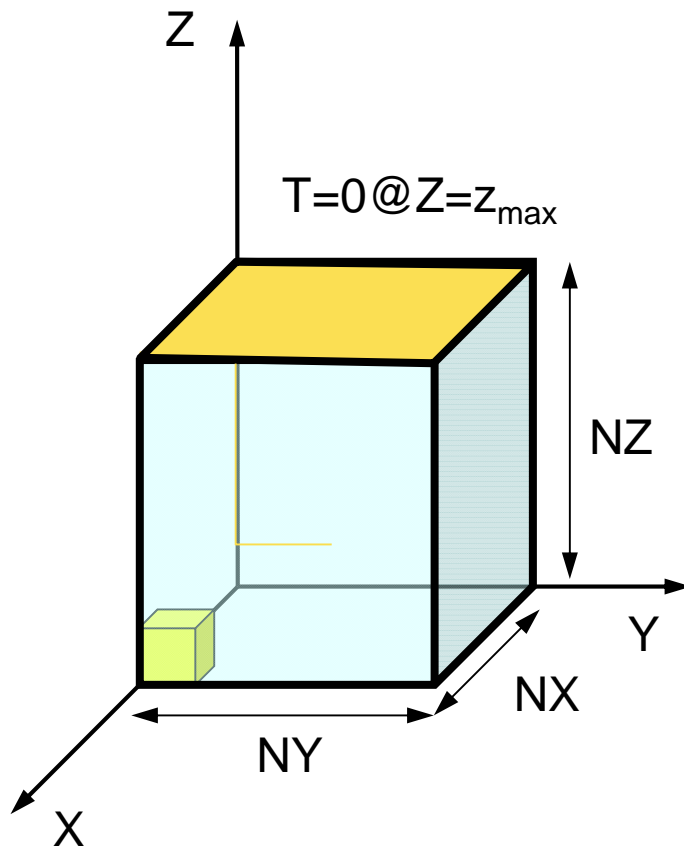


# Operations

Starting from Grid Generation to Computation, File-  
names are fixed



# Mesh Generation



```
>$ cd <$fem1>/fem3d/run
```

```
>$ ./mgcube
```

```
NX, NY, NZ
```

← Number of  
Elem's in each  
direction

```
20,20,20
```

← example

```
>$ ls cube.0
```

confirmation

```
cube.0
```

# Control File: INPUT.DAT

## INPUT.DAT

```

cube.0      fname
2000        ITER
1.0 1.0     COND, QVOL
1.0e-08     RESID
  
```

- fname : Name of Mesh File
- ITER : Max. Iterations for CG
- COND : Thermal Conductivity
- QVOL : Heat Generation Rate
- RESID : Criteria for Convergence of CG

$$\frac{\partial}{\partial x} \left( \lambda \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left( \lambda \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left( \lambda \frac{\partial T}{\partial z} \right) + \dot{Q}(x, y, z) = 0$$

$$\dot{Q}(x, y, z) = QVOL |x_c + y_c|$$

# Running

```
>$ cd <$E-TOP>/fem3d/run  
>$ ./sol
```

```
>$ ls test.inp          Confirmation  
    test.inp
```

# ParaView

- <http://www.paraview.org/>
- Opening files
- Displaying figures
- Saving image files
  - <http://nkl.cc.u-tokyo.ac.jp/class/HowtouseParaViewE.pdf>
  - <http://nkl.cc.u-tokyo.ac.jp/class/HowtouseParaViewJ.pdf>

# UCD Format (1/3)

## Unstructured Cell Data

要素の種類

点

線

三角形

四角形

四面体

角錐

三角柱

六面体

二次要素

線2

三角形2

四角形2

四面体2

角錐2

三角柱2

六面体2

キーワード

pt

line

tri

quad

tet

pyr

prism

hex

line2

tri2

quad2

tet2

pyr2

prism2

hex2

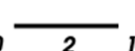
点



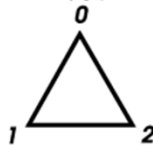
線



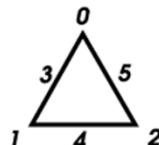
線2



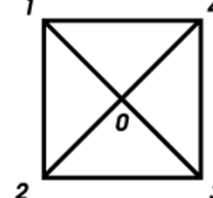
三角形



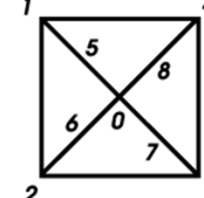
三角形2



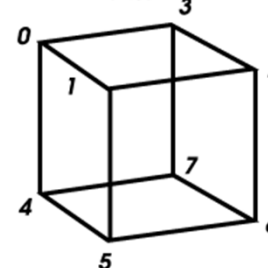
四角錐



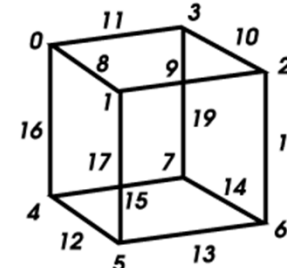
四角錐2



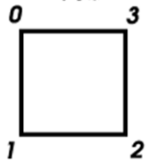
六面体



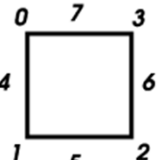
六面体2



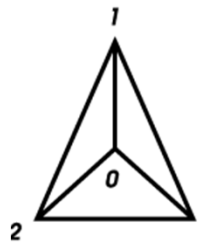
四角形



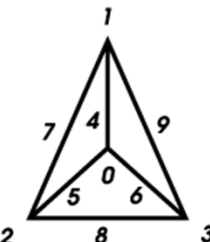
四角形2



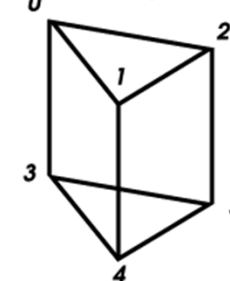
三角錐



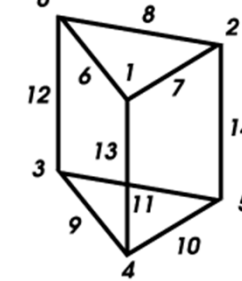
三角錐2



三角柱



三角柱2



# UCD Format (2/3)

- Originally for AVS, microAVS
- Extension of the UCD file is “inp”
- There are two types of formats. Only old type can be read by ParaView.

# UCD Format (3/3): Old Format

(全節点数) (全要素数) (各節点のデータ数) (各要素のデータ数) (モデルのデータ数)

(節点番号1) (X座標) (Y座標) (Z座標)

(節点番号2) (X座標) (Y座標) (Z座標)

·  
·  
·

(要素番号1) (材料番号) (要素の種類) (要素を構成する節点のつながり)

(要素番号2) (材料番号) (要素の種類) (要素を構成する節点のつながり)

·  
·  
·

(節点のデータ成分数) (成分1の構成数) (成分2の構成数) ... (各成分の構成数)

(節点データ成分1のラベル), (単位)

(節点データ成分2のラベル), (単位)

·  
·  
·

(各節点データ成分のラベル), (単位)

(節点番号1) (節点データ1) (節点データ2) .....

(節点番号2) (節点データ1) (節点データ2) .....

·  
·  
·

(要素のデータ成分数) (成分1の構成数) (成分2の構成数) ... (各成分の構成数)

(要素データ成分1のラベル), (単位)

(要素データ成分2のラベル), (単位)

·  
·  
·

(各要素データ成分のラベル), (単位)

(要素番号1) (要素データ1) (要素データ2) .....

(要素番号2) (要素データ1) (要素データ2) .....

·  
·  
·



- Formulation of 3D Element
- 3D Heat Equations
  - Galerkin Method
  - Element Matrices
- Running the Code
- **Data Structure**
- Overview of the Program

# Overview of Mesh File: cube.0

numbering starts from “1”

- Nodes
  - Node # (How many nodes ?)
  - Node ID, Coordinates
- Elements
  - Element #
  - Element Type
  - Element ID, Material ID, Connectivity
- Node Groups
  - Group #
  - Node # in each group
  - Group Name
  - Nodes in each group

# Mesh Generator: mgcube.f (1/5)

```

implicit REAL*8 (A-H,0-Z)
real(kind=8), dimension(:, :), allocatable :: X, Y
real(kind=8), dimension(:, :), allocatable :: XO, YO
character(len=80) :: GRIDFILE, HHH
integer , dimension(:, :), allocatable :: IW

!C
!C +-----+
!C | INIT. |
!C +-----+
!C==

write (*,*) 'NX, NY, NZ'
read (*,*) NX, NY, NZ

NXP1= NX + 1
NYP1= NY + 1
NZP1= NZ + 1

DX= 1.d0

INODTOT= NXP1*NYP1*NZP1
ICELTOT= NX *NY *NZ
IBNODTOT= NXP1*NYP1

allocate (IW(INODTOT, 4))
IW= 0

```

Node number in X-direction  
Node number in Y-direction  
Node number in Z-direction

Total number of Nodes  
Total number of Elements  
Number of nodes in XY-plane

# Mesh Generator: mgcube.f (2/5)

```

icou= 0
ib = 1
do k= 1, NZP1
do j= 1, NYP1
i= 1
icou= icou + 1
ii = (k-1)*IBNODTOT + (j-1)*NXP1 + i
IW(icou, ib)= ii
enddo
enddo

```

```

icou= 0
ib = 2
do k= 1, NZP1
j= 1
do i= 1, NXP1
icou= icou + 1
ii = (k-1)*IBNODTOT + (j-1)*NXP1 + i
IW(icou, ib)= ii
enddo
enddo

```

```

icou= 0
ib = 3
k= 1
do j= 1, NYP1
do i= 1, NXP1
icou= icou + 1
ii = (k-1)*IBNODTOT + (j-1)*NXP1 + i
IW(icou, ib)= ii
enddo
enddo

```

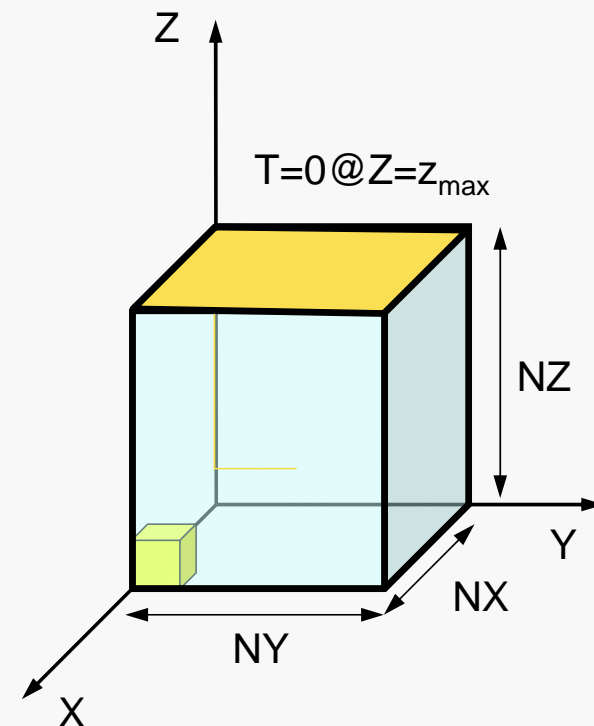
```

icou= 0
ib = 4
k= NZP1
do j= 1, NYP1
do i= 1, NXP1
icou= icou + 1
ii = (k-1)*IBNODTOT + (j-1)*NXP1 + i
IW(icou, ib)= ii
enddo
enddo

```

```
!C===
```

Nodes on  $X=X_{\min}$  surface  
are stored in  $IW(ib,1)$  where  
 $ib= 1, NYP1*NZP1$



# Mesh Generator: mgcube.f (2/5)

```

icou= 0
ib = 1
do k= 1, NZP1
do j= 1, NYP1
i= 1
icou= icou + 1
ii = (k-1)*IBNODTOT + (j-1)*NXP1 + i
IW(icou, ib)= ii
enddo
enddo

```

```

icou= 0
ib = 2
do k= 1, NZP1
j= 1
do i= 1, NXP1
icou= icou + 1
ii = (k-1)*IBNODTOT + (j-1)*NXP1 + i
IW(icou, ib)= ii
enddo
enddo

```

```

icou= 0
ib = 3
k= 1
do j= 1, NYP1
do i= 1, NXP1
icou= icou + 1
ii = (k-1)*IBNODTOT + (j-1)*NXP1 + i
IW(icou, ib)= ii
enddo
enddo

```

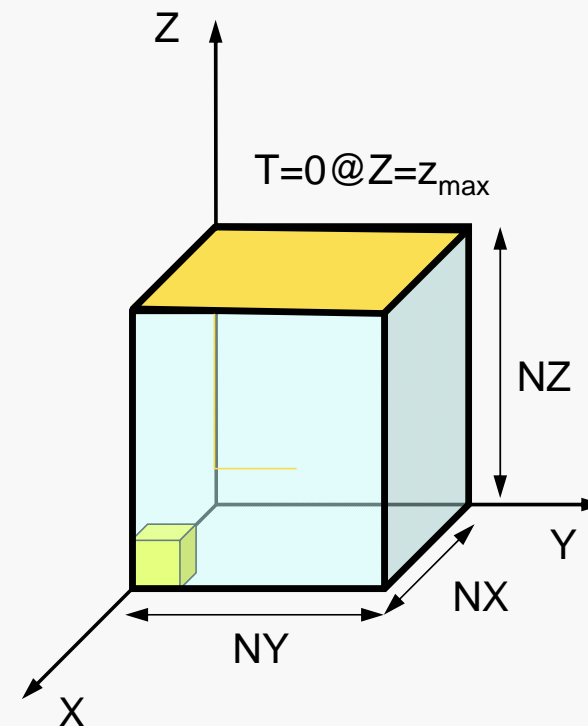
```

icou= 0
ib = 4
k= NZP1
do j= 1, NYP1
do i= 1, NXP1
icou= icou + 1
ii = (k-1)*IBNODTOT + (j-1)*NXP1 + i
IW(icou, ib)= ii
enddo
enddo

```

```
!C===
```

Nodes on  $X=X_{\min}$  surface  
are stored in  $IW(ib,1)$  where  
 $ib= 1, NYP1*NZP1$



# Mesh Generator: mgcube.f (2/5)

```

icou= 0
ib = 1
do k= 1, NZP1
do j= 1, NYP1
i= 1
icou= icou + 1
ii = (k-1)*IBNODTOT + (j-1)*NXP1 + i
IW(icou, ib)= ii
enddo
enddo

```

```

icou= 0
ib = 2
do k= 1, NZP1
j= 1
do i= 1, NXP1
icou= icou + 1
ii = (k-1)*IBNODTOT + (j-1)*NXP1 + i
IW(icou, ib)= ii
enddo
enddo

```

```

icou= 0
ib = 3
k= 1
do j= 1, NYP1
do i= 1, NXP1
icou= icou + 1
ii = (k-1)*IBNODTOT + (j-1)*NXP1 + i
IW(icou, ib)= ii
enddo
enddo

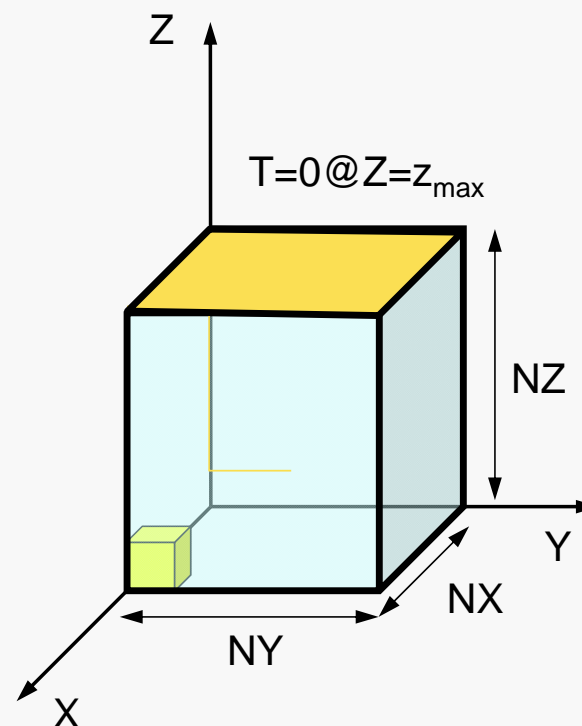
```

```

icou= 0
ib = 4
k= NZP1
do j= 1, NYP1
do i= 1, NXP1
icou= icou + 1
ii = (k-1)*IBNODTOT + (j-1)*NXP1 + i
IW(icou, ib)= ii
enddo
enddo

```

```
!C===
```



Nodes on  $Z=Z_{min}$  surface are stored in  $IW(ib,3)$  where  $ib= 1, NXP1*NYP1$

Nodes on  $Z=Z_{max}$  surface are stored in  $IW(ib,4)$  where  $ib= 1, NXP1*NYP1$

# Mesh Generator: mgcube.f (3/5)

```

|C
|C +-----+
|C | GeoFEM data |
|C +-----+
|C===
      NN= 0
      write (*,*) 'GeoFEM gridfile name ?'
      GRIDFILE= 'cube.0'

      open (12, file= GRIDFILE, status='unknown', form='formatted')
      write(12, '(10i10)') INODTOT

      icou= 0
      do k= 1, NZP1
      do j= 1, NYP1
      do i= 1, NXP1
        XX= dfloat(i-1)*DX
        YY= dfloat(j-1)*DX
        ZZ= dfloat(k-1)*DX

        icou= icou + 1
        write (12, '(i10,3(1pe16.6))') icou, XX, YY, ZZ
      enddo
      enddo
      enddo

      write(12, '(i10)') ICELTOT

      IELMTYPL= 361
      write(12, '(10i10)') (IELMTYPL, i=1, ICELTOT)

```

Node #  
Node ID, Coordinates

Element #  
Element Type (not in use)





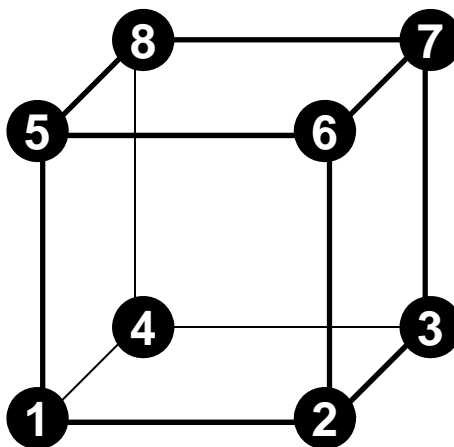
# Mesh Generator: mgcube.f (4/5)

```

icou= 0
imat= 1
do k= 1, NZ
do j= 1, NY
do i= 1, NX
  icou= icou + 1
  in1 = (k-1)*IBNODTOT + (j-1)*NXP1 + i
  in2 = in1 + 1
  in3 = in2 + NXP1
  in4 = in3 - 1
  in5 = in1 + IBNODTOT
  in6 = in2 + IBNODTOT
  in7 = in3 + IBNODTOT
  in8 = in4 + IBNODTOT
  write (12, '(10i10)') icou, imat, in1, in2, in3, in4,
    & in5, in6, in7, in8
enddo
enddo
enddo

```

& imat: Material ID (=1)



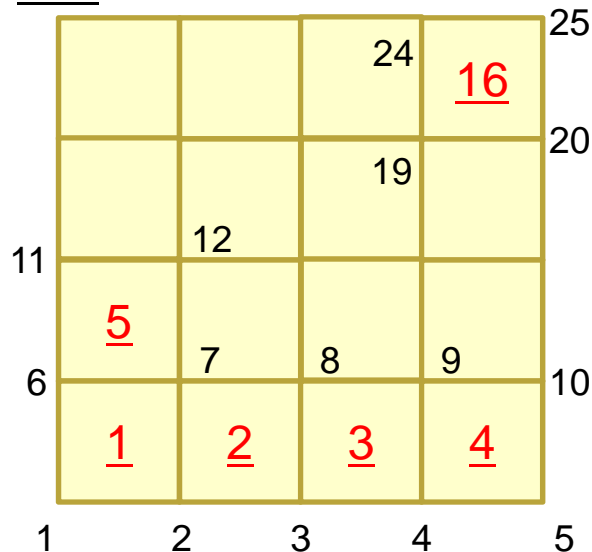
# Example of “cube.0” (NX=NY=NZ=4) Element Connectivity

1	1	1	2	7	6	26	27	32	31
2	1	2	3	8	7	27	28	33	32
3	1	3	4	9	8	28	29	34	33
4	1	4	5	10	9	29	30	35	34
5	1	6	7	12	11	31	32	37	36
6	1	7	8	13	12	32	33	38	37
7	1	8	9	14	13	33	34	39	38
8	1	9	10	15	14	34	35	40	39
9	1	11	12	17	16	36	37	42	41
10	1	12	13	18	17	37	38	43	42
11	1	13	14	19	18	38	39	44	43
12	1	14	15	20	19	39	40	45	44
13	1	16	17	22	21	41	42	47	46
(...)									
42	1	62	63	68	67	87	88	93	92
43	1	63	64	69	68	88	89	94	93
44	1	64	65	70	69	89	90	95	94
45	1	66	67	72	71	91	92	97	96
46	1	67	68	73	72	92	93	98	97
47	1	68	69	74	73	93	94	99	98
48	1	69	70	75	74	94	95	100	99
49	1	76	77	82	81	101	102	107	106
50	1	77	78	83	82	102	103	108	107
51	1	78	79	84	83	103	104	109	108
52	1	79	80	85	84	104	105	110	109
53	1	81	82	87	86	106	107	112	111
54	1	82	83	88	87	107	108	113	112
55	1	83	84	89	88	108	109	114	113
56	1	84	85	90	89	109	110	115	114
57	1	86	87	92	91	111	112	117	116
58	1	87	88	93	92	112	113	118	117
59	1	88	89	94	93	113	114	119	118
60	1	89	90	95	94	114	115	120	119
61	1	91	92	97	96	116	117	122	121
62	1	92	93	98	97	117	118	123	122
63	1	93	94	99	98	118	119	124	123
64	1	94	95	100	99	119	120	125	124

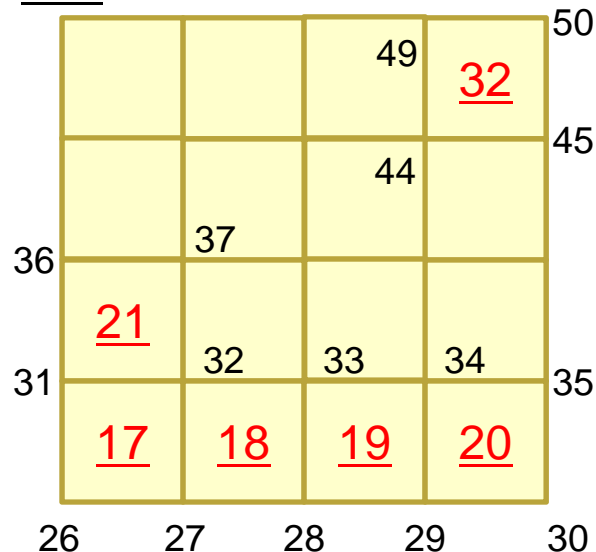
$NX=NY=NZ=4$ ,  $NXP1=NYP1=NZP1=5$

$ICELTOT=64$ ,  $INODDTOT=125$ ,  $IBNODDTOT=25$

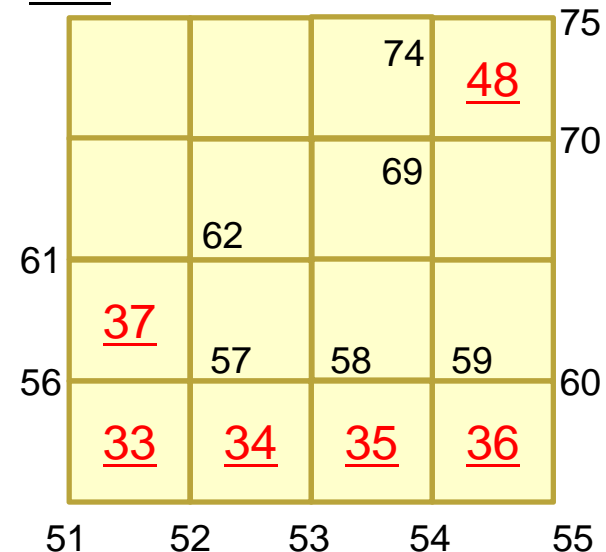
**k=1**



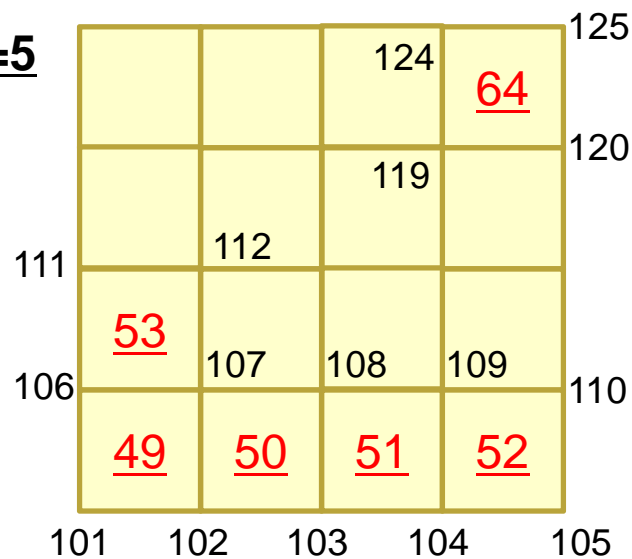
**k=2**



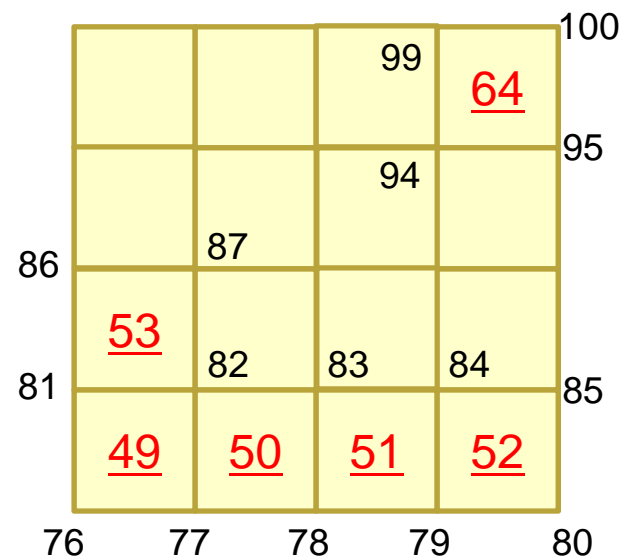
**k=3**



**k=5**



**k=4**



# Mesh Generator: mgcube.f (5/5)

```
IGTOT= 4
```

```
IBT1= NYP1*NZP1
IBT2= NXP1*NZP1 + IBT1
IBT3= NXP1*NYP1 + IBT2
IBT4= NXP1*NYP1 + IBT3
```

```
write (12, '(10i10)') IGTOT
write (12, '(10i10)') IBT1, IBT2, IBT3, IBT4
```

```
HHH= 'Xmin'
write (12, '(a80)') HHH
write (12, '(10i10)') (IW(ii,1), ii=1, NYP1*NZP1)
HHH= 'Ymin'
write (12, '(a80)') HHH
write (12, '(10i10)') (IW(ii,2), ii=1, NXP1*NZP1)
HHH= 'Zmin'
write (12, '(a80)') HHH
write (12, '(10i10)') (IW(ii,3), ii=1, NXP1*NYP1)
HHH= 'Zmax'
write (12, '(a80)') HHH
write (12, '(10i10)') (IW(ii,4), ii=1, NXP1*NYP1)
```

```
(以下略) deallocate (IW)
close (12)
!C===
stop
end
```

IGTOT	Group # (Xmin, Ymin, Zmin, Zmax)
IBTx	Accumulated #

# Example of “cube.0” (NX=NY=NZ=4) Group Info.

	4									
Xmin	25	50	75	100						
	1	6	11	16	21	26	31	36	41	46
	51	56	61	66	71	76	81	86	91	96
Ymin	101	106	111	116	121					
	1	2	3	4	5	26	27	28	29	30
	51	52	53	54	55	76	77	78	79	80
Zmin	101	102	103	104	105					
	1	2	3	4	5	6	7	8	9	10
	11	12	13	14	15	16	17	18	19	20
Zmax	21	22	23	24	25					
	101	102	103	104	105	106	107	108	109	110
	111	112	113	114	115	116	117	118	119	120
	121	122	123	124	125					

(not in use after this point)

# Mesh Generation

- Big Technical & Research Issue
  - Complicated Geometry
  - Large Scale
- Parallelization is difficult
- Commercial Mesh Generator
  - FEMAP
    - Interface to CAD Data Format



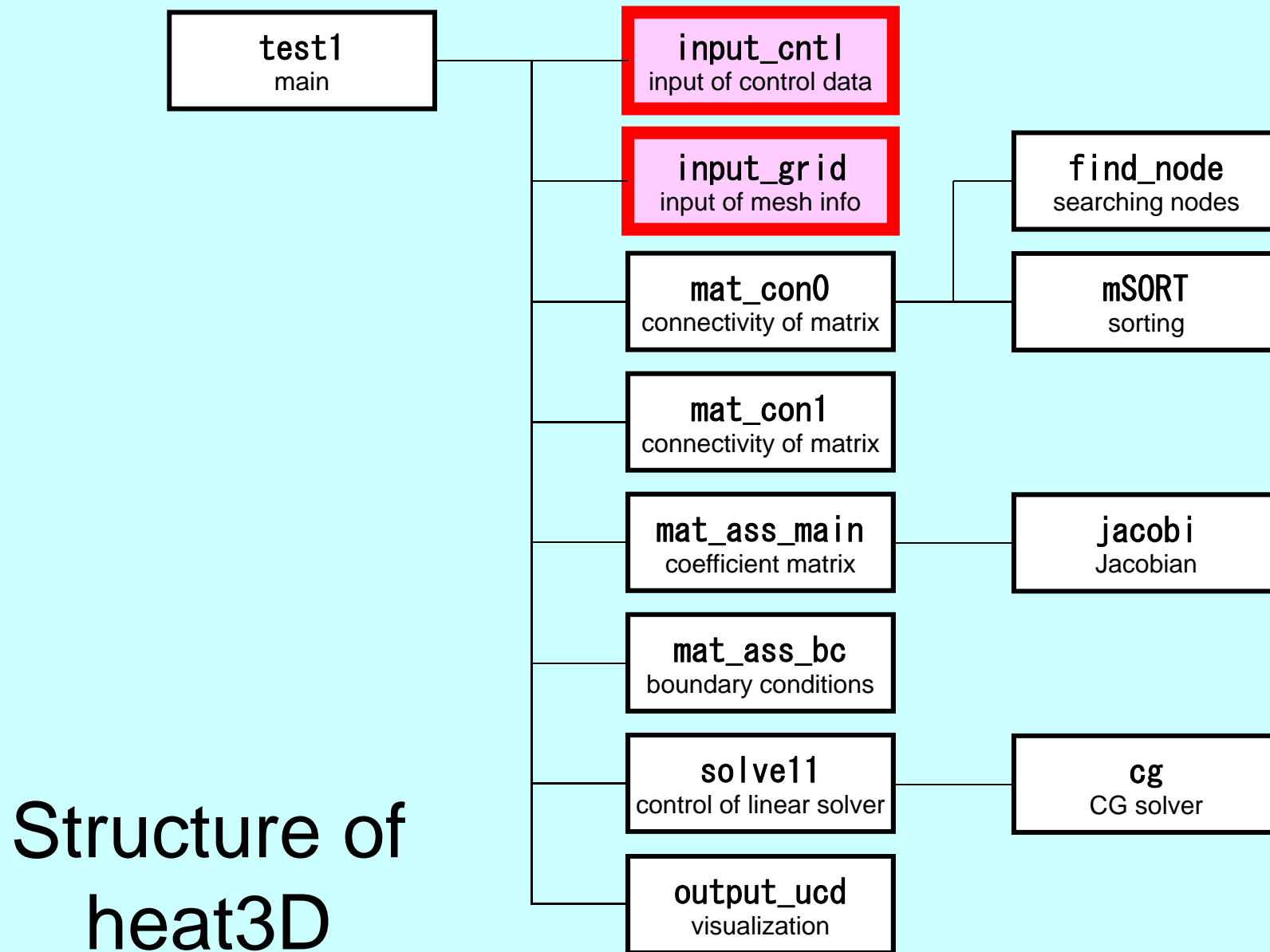
movie

- Formulation of 3D Element
- 3D Heat Equations
  - Galerkin Method
  - Element Matrices
- Running the Code
- Data Structure
- **Overview of the Program**

# 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





# Main Part

```
program heat3D
  use solver11
  use pfem_util

  implicit REAL*8 (A-H, O-Z)

  call INPUT_CNTL
  call INPUT_GRID

  call MAT_CONO
  call MAT_CON1

  call MAT_ASS_MAIN
  call MAT_ASS_BC

  call SOLVE11

  call OUTPUT_UCD

end program heat3D
```

# Global Variables: pfem\_util.f (1/3)

Name	Type	Size	I/O	Definition
<b>fname</b>	C	(80)	I	Name of mesh file
<b>N, NP</b>	I		I	# Node
<b>ICELTOT</b>	I		I	# Element
<b>NODGRPtot</b>	I		I	# Node Group
<b>XYZ</b>	R	(N, 3)	I	Node Coordinates
<b>ICELNOD</b>	I	(ICELTOT, 8)	I	Element Connectivity
<b>NODGRP_INDEX</b>	I	(0:NODGRPtot)	I	# Node in each Node Group
<b>NODGRP_ITEM</b>	I	(NODGRP_INDEX(NODGRPtot))	I	Node ID in each Node Group
<b>NODGRP_NAME</b>	C80	(NODGRP_INDEX(NODGRPtot))	I	Name of NodeGroup
<b>NLU</b>	I		O	# Non-Zero Off-Diagonals at each node
<b>NPLU</b>	I		O	# Non-Zero Off-Diagonals
<b>D</b>	R	(N)	O	Diagonal Block of Global Matrix
<b>B, X</b>	R	(N)	O	RHS, Unknown Vector

# Global Variables: pfem\_util.h (2/3)

Name	Type	Size	I/O	Definition
<b>AMAT</b>	R	(NPLU)	○	Non-Zero Off-Diagonal Components of Global Matrix
<b>index</b>	I	(0:N)	○	# Non-Zero Off-Diagonal Components
<b>item</b>	I	(NPLU)	○	Column ID of Non-Zero Off-Diagonal Components
<b>INLU</b>	I	(N)	○	Number of Non-Zero Off-Diagonal Components at Each Node
<b>IALU</b>	I	(N,NLU)	○	Column ID of Non-Zero Off-Diagonal Components at Each Node
<b>IWKX</b>	I	(N,2)	○	Work Arrays
<b>ITER, ITERactual</b>	I		I	Number of CG Iterations (MAX, Actual)
<b>RESID</b>	R		I	Convergence Criteria (fixed as 1.e-8)
<b>pfemIarray</b>	I	(100)	○	Integer Parameter Array
<b>pfemRarray</b>	R	(100)	○	Real Parameter Array

# Global Variables: pfem\_util.h (3/3)

Name	Type	Size	I/O	Definition
<b>O8th</b>	R		I	= 0.125
<b>PNQ, PNE, PNT</b>	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
<b>POS, WEI</b>	R	(2)	O	Coordinates, Weighting Factor at each Gaussian Quad. Point
<b>NCOL1, NCOL2</b>	I	(100)	O	Work arrays for sorting
<b>SHAPE</b>	R	(2, 2, 2, 8)	O	$N_i (i=1 \sim 8)$ at each Gaussian Quad Point
<b>PNX, PNY, PNZ</b>	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
<b>DETJ</b>	R	(2, 2, 2)	O	Determinant of Jacobian Matrix at each Gaussian Quad. Point
<b>COND, QVOL</b>	R		I	Thermal Conductivity, Heat Generation Rate

$$\frac{\partial}{\partial x} \left( \lambda \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left( \lambda \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left( \lambda \frac{\partial T}{\partial z} \right) + \dot{Q}(x, y, z) = 0$$

$$\dot{Q}(x, y, z) = QVOL |x_c + y_c|$$

# INPUT\_CNTL: Control Data

```
subroutine INPUT_CNTL
use pfem_util

implicit REAL*8 (A-H, O-Z)

open (11, file= 'INPUT.DAT', status='unknown')
read (11, '(a80)') fname
read (11, *) ITER
read (11, *) COND, QVOL
read (11, *) RESID
close (11)

pfemIarray(1)= ITER
pfemRarray(1)= RESID

return
end
```

## INPUT.DAT

cube . 0	fname
2000	ITER
1.0 1.0	COND, QVOL
1.0e-08	RESID

# INPUT\_GRID (1/2)

```
!C***  
!C*** INPUT_GRID  
!C***  
!C  
  subroutine INPUT_GRID  
  use pfem_util  
  implicit REAL*8 (A-H, O-Z)  
  
  open (11, file= fname, status= 'unknown', form= 'formatted')  
  
!C  
!C--- NODE  
  read (11,*) N  
  NP= N  
  
  allocate (XYZ(N, 3))  
  XYZ= 0. d0  
  do i= 1, N  
    read (11,*) ii, (XYZ(i, kk), kk=1, 3)  
  enddo
```

# allocate, deallocate (C)

```

#include <stdio.h>
#include <stdlib.h>
void* allocate_vector(int size, int m)
{
    void *a;
    if ( ( a=(void *)malloc( m * size ) ) == NULL ) {
        fprintf(stdout, "Error:Memory does not enough! in vector %n");
        exit(1);
    }
    return a;
}

void deallocate_vector(void *a)
{
    free( a );
}

void** allocate_matrix(int size, int m, int n)
{
    void **aa;
    int i;
    if ( ( aa=(void **)malloc( m * sizeof(void*) ) ) == NULL ) {
        fprintf(stdout, "Error:Memory does not enough! aa in matrix %n");
        exit(1);
    }
    if ( ( aa[0]=(void *)malloc( m * n * size ) ) == NULL ) {
        fprintf(stdout, "Error:Memory does not enough! in matrix %n");
        exit(1);
    }
    for(i=1; i<m; i++) aa[i]=(char*)aa[i-1]+size*n;
    return aa;
}

void deallocate_matrix(void **aa)
{
    free( aa );
}

```

Same interface with FORTRAN



# INPUT\_GRID (1/2)

```

!C
!C-- ELEMENT
  read (11,*) ICELTOT
  allocate (ICELNOD(ICELTOT,8))
  read (11,'(10i10)') (NTYPE, i= 1, ICELTOT)

  do icel= 1, ICELTOT
    read (11,'(10i10,2i5,8i8)') ii, IMAT, (ICELNOD(icel,k), k=1,8)
  enddo

!C
!C-- NODE grp. info.
  read (11,'(10i10)') NODGRPtot
  allocate (NODGRP_INDEX(0:NODGRPtot), NODGRP_NAME(NODGRPtot))
  NODGRP_INDEX= 0

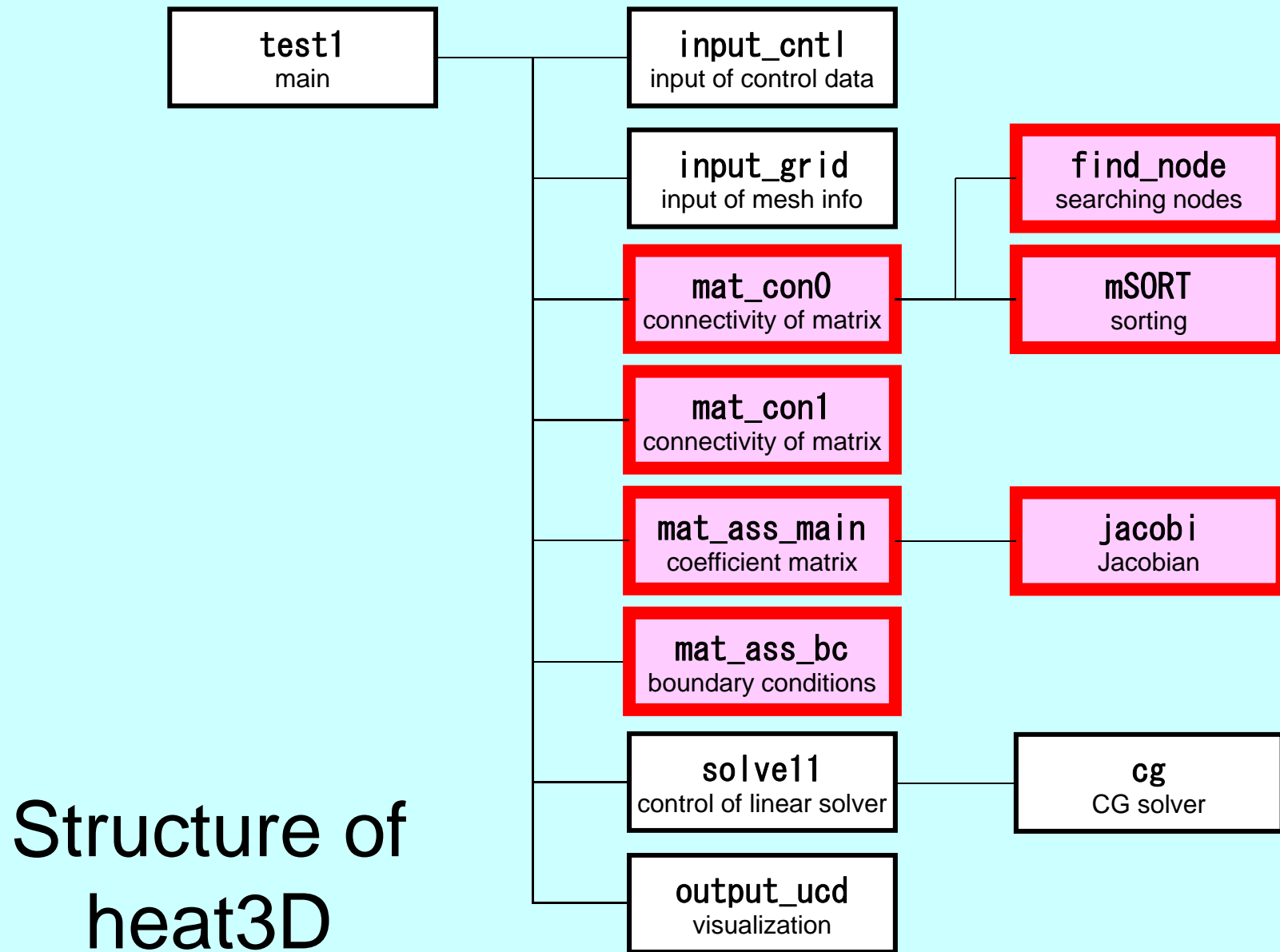
  read (11,'(10i10)') (NODGRP_INDEX(i), i= 1, NODGRPtot)
  nn= NODGRP_INDEX(NODGRPtot)
  allocate (NODGRP_ITEM(nn))

  do k= 1, NODGRPtot
    iS= NODGRP_INDEX(k-1) + 1
    iE= NODGRP_INDEX(k)
    read (11,'(a80)') NODGRP_NAME(k)
    nn= iE - iS + 1
    if (nn.ne.0) then
      read (11,'(10i10)') (NODGRP_ITEM(kk),kk=iS, iE)
    endif
  enddo

  close (11)

  return
end

```



# Global Variables: pfem\_util.f (1/3)

Name	Type	Size	I/O	Definition
<b>fname</b>	C	(80)	I	Name of mesh file
<b>N, NP</b>	I		I	# Node
<b>ICELTOT</b>	I		I	# Element
<b>NODGRPtot</b>	I		I	# Node Group
<b>XYZ</b>	R	(N, 3)	I	Node Coordinates
<b>ICELNOD</b>	I	(ICELTOT, 8)	I	Element Connectivity
<b>NODGRP_INDEX</b>	I	(0:NODGRPtot)	I	# Node in each Node Group
<b>NODGRP_ITEM</b>	I	(NODGRP_INDEX(NODGRPtot))	I	Node ID in each Node Group
<b>NODGRP_NAME</b>	C80	(NODGRP_INDEX(NODGRPtot))	I	Name of NodeGroup
<b>NLU</b>	I		O	# Non-Zero Off-Diagonals at each node
<b>NPLU</b>	I		O	# Non-Zero Off-Diagonals
<b>D</b>	R	(N)	O	Diagonal Block of Global Matrix
<b>B, X</b>	R	(N)	O	RHS, Unknown Vector

# Global Variables: pfem\_util.h (2/3)

Name	Type	Size	I/O	Definition
<b>AMAT</b>	R	(NPLU)	O	Non-Zero Off-Diagonal Components of Global Matrix
<b>index</b>	I	(0:N)	O	# Non-Zero Off-Diagonal Components
<b>item</b>	I	(NPLU)	O	Column ID of Non-Zero Off-Diagonal Components
<b>INLU</b>	I	(N)	O	Number of Non-Zero Off-Diagonal Components at Each Node
<b>IALU</b>	I	(N,NLU)	O	Column ID of Non-Zero Off-Diagonal Components at Each Node
<b>IWKX</b>	I	(N,2)	O	Work Arrays
<b>ITER, ITERactual</b>	I		I	Number of CG Iterations (MAX, Actual)
<b>RESID</b>	R		I	Convergence Criteria (fixed as 1.e-8)
<b>pfemIarray</b>	I	(100)	O	Integer Parameter Array
<b>pfemRarray</b>	R	(100)	O	Real Parameter Array

# Global Variables: pfem\_util.h (3/3)

Name	Type	Size	I/O	Definition
<b>O8th</b>	R		I	= 0.125
<b>PNQ, PNE, PNT</b>	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
<b>POS, WEI</b>	R	(2)	O	Coordinates, Weighting Factor at each Gaussian Quad. Point
<b>NCOL1, NCOL2</b>	I	(100)	O	Work arrays for sorting
<b>SHAPE</b>	R	(2, 2, 2, 8)	O	$N_i (i=1 \sim 8)$ at each Gaussian Quad Point
<b>PNX, PNY, PNZ</b>	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
<b>DETJ</b>	R	(2, 2, 2)	O	Determinant of Jacobian Matrix at each Gaussian Quad. Point
<b>COND, QVOL</b>	R		I	Thermal Conductivity, Heat Generation Rate

$$\frac{\partial}{\partial x} \left( \lambda \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left( \lambda \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left( \lambda \frac{\partial T}{\partial z} \right) + \dot{Q}(x, y, z) = 0$$

$$\dot{Q}(x, y, z) = QVOL |x_c + y_c|$$

# Towards Matrix Assembling

- In 1D, it was easy to obtain information related to index and item.
  - 2 non-zero off-diagonals for each node
  - ID of non-zero off-diagonal :  $i+1$ ,  $i-1$ , where “i” is node ID
- In 3D, situation is more complicated:
  - Number of non-zero off-diagonal components is between 7 and 26 for the current target problem
  - More complicated for real problems.
  - Generally, there are no information related to number of non-zero off-diagonal components beforehand.



movie

# Towards Matrix Assembling

- In 1D, it was easy to obtain information related to index and item.
  - 2 non-zero off-diagonals for each node
  - ID of non-zero off-diagonal :  $i+1$ ,  $i-1$ , where “i” is node ID
- In 3D, situation is more complicated:
  - Number of non-zero off-diagonal components is between 7 and 26 for the current target problem
  - More complicated for real problems.
  - Generally, there are no information related to number of non-zero off-diagonal components beforehand.
- **Count number of non-zero off-diagonals using arrays:  $INLU[N]$ ,  $IALU[N][NLU]$**

# Main Part

```

program heat3D
  use solver11
  use pfem_util

  implicit REAL*8(A-H, O-Z)

  call INPUT_CNTL
  call INPUT_GRID

  call MAT_CON0
  call MAT_CON1

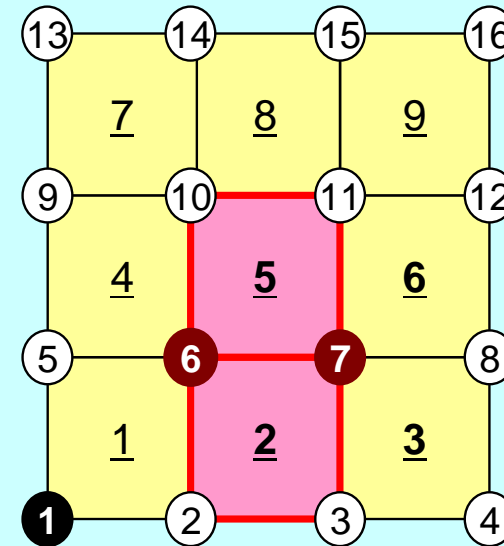
  call MAT_ASS_MAIN
  call MAT_ASS_BC

  call SOLVE11

  call OUTPUT_UCD

end program heat3D

```



MAT\_CON0: generates INU, IALU

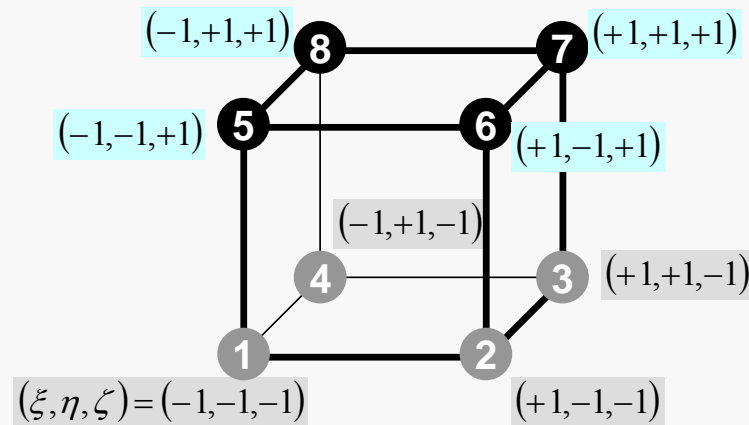
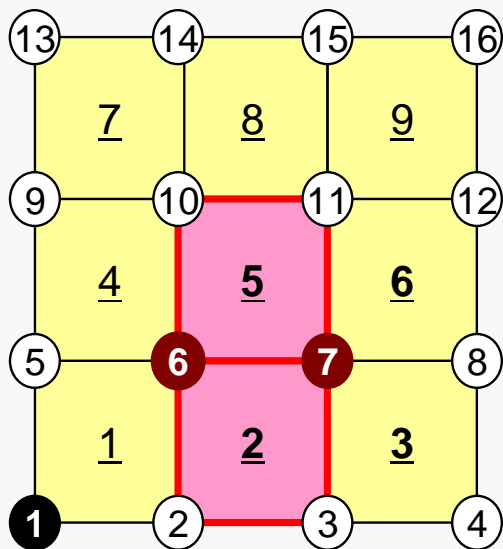
MAT\_CON1: generates index, item

Node ID starting from "1"



# MAT\_CON0: Overview

```
do icel= 1, ICELTOT
  generate INLU, IALU
  according to 8 nodes of hex. elements
  (FIND_NODE)
enddo
```



# Generating Connectivity of Matrix MAT\_CONO (1/4)

```
!C
!C***
!C*** MAT_CONO
!C***
!C
  subroutine MAT_CONO
  use pfem_util
  implicit REAL*8 (A-H, O-Z)

  NLU= 26

  allocate (INLU(N), IALU(N, NLU))

  INLU= 0
  IALU= 0
```

## NLU:

Number of maximum number of connected nodes to each node (number of upper/lower non-zero off-diagonal nodes)

In the current problem, geometry is rather simple. Therefore we can specify NLU in this way.

If it's not clear ->  
Try more flexible implementation

# Generating Connectivity of Matrix MAT\_CON0 (1/4)

```
!C
!C***
!C*** MAT_CON0
!C***
!C
subroutine MAT_CON0
use pfem_util
implicit REAL*8 (A-H, O-Z)

NLU= 26

allocate (INLU(N), IALU(N, NLU))

INLU= 0
IALU= 0
```

Array	Size	Description
INLU	(N)	Number of connected nodes to each node (lower/upper)
IALU	(N, NLU)	Corresponding connected node ID (column ID)

# Generating Connectivity of Matrix MAT\_CON0 (2/4): Starting from 1

```

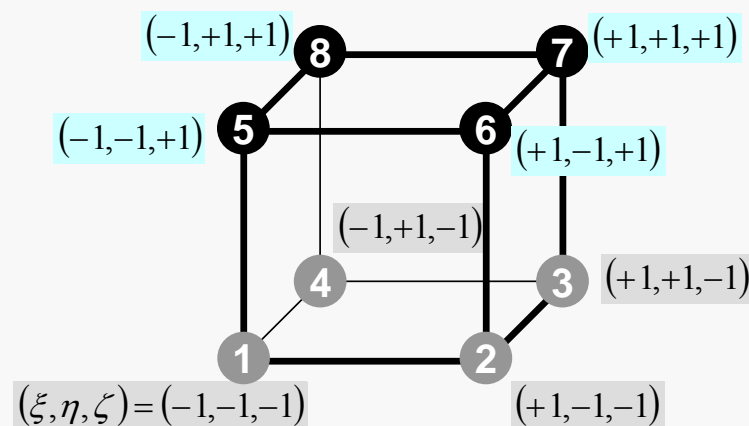
do icel= 1, ICELTOT
  in1= ICELNOD(icel, 1)
  in2= ICELNOD(icel, 2)
  in3= ICELNOD(icel, 3)
  in4= ICELNOD(icel, 4)
  in5= ICELNOD(icel, 5)
  in6= ICELNOD(icel, 6)
  in7= ICELNOD(icel, 7)
  in8= ICELNOD(icel, 8)

  call FIND_TS_NODE (in1, in2)
  call FIND_TS_NODE (in1, in3)
  call FIND_TS_NODE (in1, in4)
  call FIND_TS_NODE (in1, in5)
  call FIND_TS_NODE (in1, in6)
  call FIND_TS_NODE (in1, in7)
  call FIND_TS_NODE (in1, in8)

  call FIND_TS_NODE (in2, in1)
  call FIND_TS_NODE (in2, in3)
  call FIND_TS_NODE (in2, in4)
  call FIND_TS_NODE (in2, in5)
  call FIND_TS_NODE (in2, in6)
  call FIND_TS_NODE (in2, in7)
  call FIND_TS_NODE (in2, in8)

  call FIND_TS_NODE (in3, in1)
  call FIND_TS_NODE (in3, in2)
  call FIND_TS_NODE (in3, in4)
  call FIND_TS_NODE (in3, in5)
  call FIND_TS_NODE (in3, in6)
  call FIND_TS_NODE (in3, in7)
  call FIND_TS_NODE (in3, in8)

```



# FIND\_TS\_NODE: Search Connectivity

## INLU,IALU: Automatic Search

```

!C
!C***
!C*** FIND_TS_NODE
!C***
!C
      subroutine FIND_TS_NODE (ip1, ip2)
         do kk= 1, INLU(ip1)
            if (ip2. eq. IALU(ip1, kk)) return
         enddo
         icou= INLU(ip1) + 1
         IALU(ip1, icou)= ip2
         INLU(ip1      )= icou
         return
      end subroutine FIND_TS_NODE

```

Array	Size	Description
INLU	(N)	Number of connected nodes to each node (lower/upper)
IALU	(N, NLU)	Corresponding connected node ID (column ID)

# FIND\_TS\_NODE: Search Connectivity

## INLU,IALU: Automatic Search

```

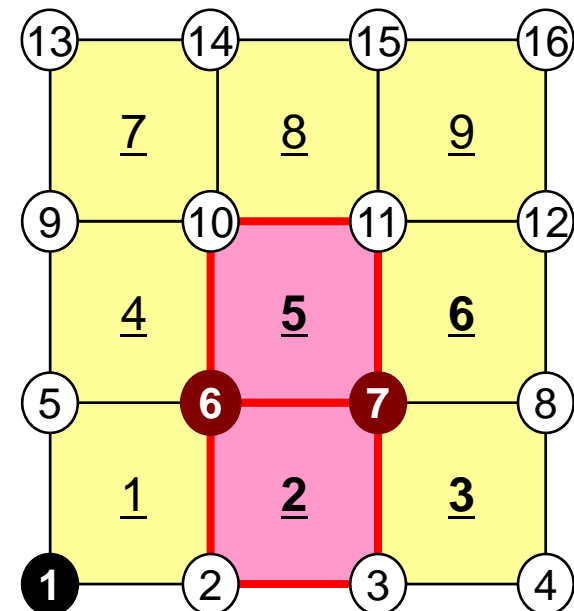
!C
!C***
!C*** FIND_TS_NODE
!C***
!C
      subroutine FIND_TS_NODE (ip1, ip2)

         do kk= 1, INLU(ip1)
            if (ip2. eq. IALU(ip1, kk)) return
         enddo
         icou= INLU(ip1) + 1
         IALU(ip1, icou)= ip2
         INLU(ip1      )= icou
         return

      end subroutine FIND_TS_NODE

```

If the target node is already included in IALU, proceed to next pair of nodes



# FIND\_TS\_NODE: Search Connectivity

## INLU,IALU: Automatic Search

```

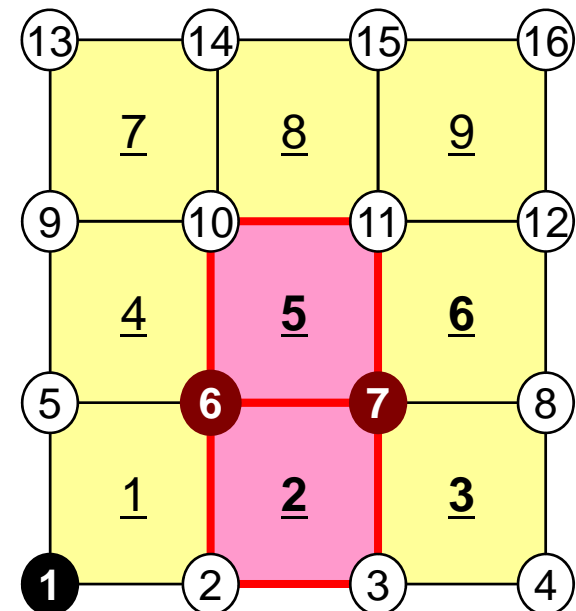
!C
!C***
!C*** FIND_TS_NODE
!C***
!C
      subroutine FIND_TS_NODE (ip1, ip2)

         do kk= 1, INLU(ip1)
            if (ip2. eq. IALU(ip1, kk)) return
         enddo
         icou= INLU(ip1) + 1
         IALU(ip1, icou)= ip2
         INLU(ip1      )= icou
         return

      end subroutine FIND_TS_NODE

```

If the target node is NOT included in IALU, store the node in IALU, and add 1 to INLU.



# Generating Connectivity of Matrix MAT\_CON0 (3/4)

```

call FIND_TS_NODE (in4, in1)
call FIND_TS_NODE (in4, in2)
call FIND_TS_NODE (in4, in3)
call FIND_TS_NODE (in4, in5)
call FIND_TS_NODE (in4, in6)
call FIND_TS_NODE (in4, in7)
call FIND_TS_NODE (in4, in8)

```

```

call FIND_TS_NODE (in5, in1)
call FIND_TS_NODE (in5, in2)
call FIND_TS_NODE (in5, in3)
call FIND_TS_NODE (in5, in4)
call FIND_TS_NODE (in5, in6)
call FIND_TS_NODE (in5, in7)
call FIND_TS_NODE (in5, in8)

```

```

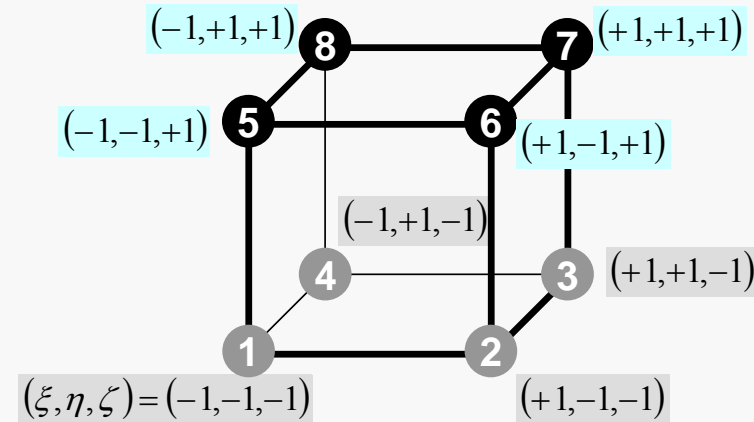
call FIND_TS_NODE (in6, in1)
call FIND_TS_NODE (in6, in2)
call FIND_TS_NODE (in6, in3)
call FIND_TS_NODE (in6, in4)
call FIND_TS_NODE (in6, in5)
call FIND_TS_NODE (in6, in7)
call FIND_TS_NODE (in6, in8)

```

```

call FIND_TS_NODE (in7, in1)
call FIND_TS_NODE (in7, in2)
call FIND_TS_NODE (in7, in3)
call FIND_TS_NODE (in7, in4)
call FIND_TS_NODE (in7, in5)
call FIND_TS_NODE (in7, in6)
call FIND_TS_NODE (in7, in8)

```





# Generating Connectivity of Matrix MAT\_CON0 (4/4)

```
call FIND_TS_NODE (in8, in1)
call FIND_TS_NODE (in8, in2)
call FIND_TS_NODE (in8, in3)
call FIND_TS_NODE (in8, in4)
call FIND_TS_NODE (in8, in5)
call FIND_TS_NODE (in8, in6)
call FIND_TS_NODE (in8, in7)
enddo

do in= 1, N
  NN= INLU(in)
  do k= 1, NN
    NCOL1(k)= IALU(in, k)
  enddo
  call mSORT (NCOL1, NCOL2, NN)
  do k= NN, 1, -1
    IALU(in, NN-k+1)= NCOL1(NCOL2(k))
  enddo
enddo
```

Sort IALU(i,k) in ascending order by  
“bubble” sorting for less than 100  
components.

# MAT\_CON1: CRS format

```

!C
!C***
!C*** MAT_CON1
!C***
!C
  subroutine MAT_CON1
  use pfem_util
  implicit REAL*8 (A-H, O-Z)

  allocate (index(0:N))
  index= 0

  do i= 1, N
    index(i)= index(i-1) + INLU(i)
  enddo

  NPLU= index(N)

  allocate (item(NPLU))

  do i= 1, N
    do k= 1, INLU(i)
      kk = k + index(i-1)
      item(kk)= IALU(i, k)
    enddo
  enddo

  deallocate (INLU, IALU)

  end subroutine MAT_CON1

```

C

$$\text{index}[i + 1] = \sum_{k=0}^i \text{INLU}[k]$$

$$\text{index}[0] = 0$$

FORTRAN

$$\text{index}(i) = \sum_{k=1}^i \text{INLU}(k)$$

$$\text{index}(0) = 0$$

# MAT\_CON1: CRS format

```

!C
!C***
!C*** MAT_CON1
!C***
!C
  subroutine MAT_CON1
  use pfem_util
  implicit REAL*8 (A-H, O-Z)

  allocate (index(0:N))
  index= 0

  do i= 1, N
    index(i)= index(i-1) + INLU(i)
  enddo

  NPLU= index(N)

  allocate (item(NPLU))

  do i= 1, N
    do k= 1, INLU(i)
      kk = k + index(i-1)
      item(kk)= IALU(i, k)
    enddo
  enddo

  deallocate (INLU, IALU)

  end subroutine MAT_CON1

```

**NPLU=index(N)**  
**Size of array: item**  
**Total number of non-zero off-diagonal blocks**

# MAT\_CON1: CRS format

```
!C
!C***
!C*** MAT_CON1
!C***
!C
  subroutine MAT_CON1
  use pfem_util
  implicit REAL*8 (A-H, O-Z)

  allocate (index(0:N))
  index= 0

  do i= 1, N
    index(i)= index(i-1) + INLU(i)
  enddo

  NPLU= index(N)

  allocate (item(NPLU))

  do i= 1, N
    do k= 1, INLU(i)
      kk = k + index(i-1)
      item(kk)= IALU(i, k)
    enddo
  enddo

  deallocate (INLU, IALU)

  end subroutine MAT_CON1
```

item  
store node ID starting from 1

# MAT\_CON1: CRS format

```
!C
!C***
!C*** MAT_CON1
!C***
!C
  subroutine MAT_CON1
  use pfem_util
  implicit REAL*8 (A-H, O-Z)

  allocate (index(0:N))
  index= 0

  do i= 1, N
    index(i)= index(i-1) + INLU(i)
  enddo

  NPLU= index(N)

  allocate (item(NPLU))

  do i= 1, N
    do k= 1, INLU(i)
      kk = k + index(i-1)
      item(kk)= IALU(i, k)
    enddo
  enddo

  deallocate (INLU, IALU)

end subroutine MAT_CON1
```

Not required any more

# Main Part

```
program heat3D
  use solver11
  use pfem_util

  implicit REAL*8(A-H, O-Z)

  call INPUT_CNTL
  call INPUT_GRID

  call MAT_CON0
  call MAT_CON1

  call MAT_ASS_MAIN
  call MAT_ASS_BC

  call SOLVE11

  call OUTPUT_UCD

end program heat3D
```



# MAT\_ASS\_MAIN (1/6)

```
!C
!C***
!C*** MAT_ASS_MAIN
!C***
!C
  subroutine MAT_ASS_MAIN
  use pfem_util
  implicit REAL*8 (A-H, O-Z)
  integer(kind=kint), dimension( 8) :: nodLOCAL

  allocate (AMAT(NPLU))
  allocate (B(N), D(N), X(N))

  AMAT= 0. d0
  B= 0. d0
  X= 0. d0
  D= 0. d0

  WEI (1)= +1. 0000000000D+00
  WEI (2)= +1. 0000000000D+00

  POS (1)= -0. 5773502692D+00
  POS (2)= +0. 5773502692D+00

  Non-Zero Off-Diagonal components (coef. matrix)
  RHS vector
  Unknowns
  Diagonal components (coef. matrix)
```



# MAT\_ASS\_MAIN (1/6)

```

!C
!C***
!C*** MAT_ASS_MAIN
!C***
!C
subroutine MAT_ASS_MAIN
use pfem_util
implicit REAL*8 (A-H,O-Z)
integer(kind=kint), dimension( 8) :: nodLOCAL

allocate (AMAT(NPLU))
allocate (B(N), D(N), X(N))

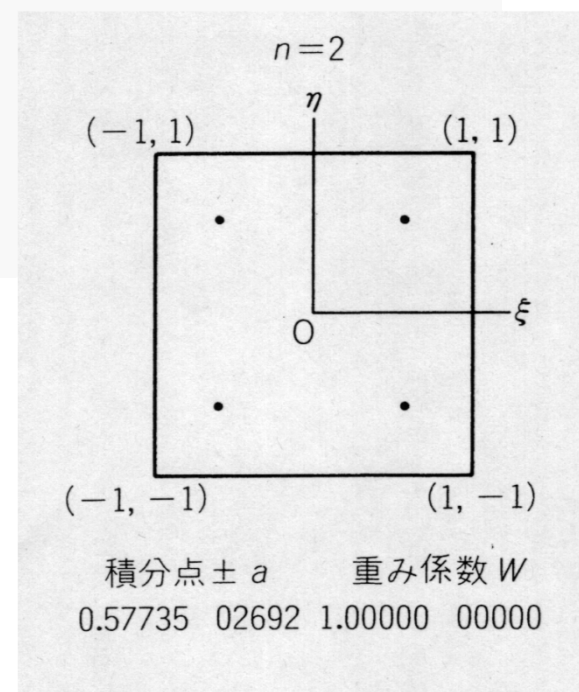
AMAT= 0. d0
B= 0. d0
X= 0. d0
D= 0. d0

WEI (1)= +1. 0000000000D+00
WEI (2)= +1. 0000000000D+00

POS (1)= -0. 5773502692D+00
POS (2)= +0. 5773502692D+00

```

*POS*: Quad. Point  
*WEI*: Weighting Factor



# MAT\_ASS\_MAIN (2/6)

```
!C
!C-- INIT.
!C   PNI  - 1st-order derivative of shape function by QSI
!C   PNE  - 1st-order derivative of shape function by ETA
!C   PNT  - 1st-order derivative of shape function by ZET
!C
```

```
do kp= 1, 2
do jp= 1, 2
do ip= 1, 2
```

```
QP1= 1. d0 + POS(ip)
QM1= 1. d0 - POS(ip)
EP1= 1. d0 + POS(jp)
EM1= 1. d0 - POS(jp)
TP1= 1. d0 + POS(kp)
TM1= 1. d0 - POS(kp)
```

```
SHAPE(ip, jp, kp, 1) = 08th * QM1 * EM1 * TM1
SHAPE(ip, jp, kp, 2) = 08th * QP1 * EM1 * TM1
SHAPE(ip, jp, kp, 3) = 08th * QP1 * EP1 * TM1
SHAPE(ip, jp, kp, 4) = 08th * QM1 * EP1 * TM1
SHAPE(ip, jp, kp, 5) = 08th * QM1 * EM1 * TP1
SHAPE(ip, jp, kp, 6) = 08th * QP1 * EM1 * TP1
SHAPE(ip, jp, kp, 7) = 08th * QP1 * EP1 * TP1
```

# MAT\_ASS\_MAIN (2/6)

```
!C
!C-- INIT.
!C   PNQ  - 1st-order derivative of shape function by QSI
!C   PNE  - 1st-order derivative of shape function by ETA
!C   PNT  - 1st-order derivative of shape function by ZET
!C
```

```
do kp= 1, 2
do jp= 1, 2
do ip= 1, 2
```

```
QP1= 1. d0 + POS(ip)
QM1= 1. d0 - POS(ip)
EP1= 1. d0 + POS(jp)
EM1= 1. d0 - POS(jp)
TP1= 1. d0 + POS(kp)
TM1= 1. d0 - POS(kp)
```

```
SHAPE(ip, jp, kp, 1) = 08th * QM1 * EM1 * TM1
SHAPE(ip, jp, kp, 2) = 08th * QP1 * EM1 * TM1
SHAPE(ip, jp, kp, 3) = 08th * QP1 * EP1 * TM1
SHAPE(ip, jp, kp, 4) = 08th * QM1 * EP1 * TM1
SHAPE(ip, jp, kp, 5) = 08th * QM1 * EM1 * TP1
SHAPE(ip, jp, kp, 6) = 08th * QP1 * EM1 * TP1
SHAPE(ip, jp, kp, 7) = 08th * QP1 * EP1 * TP1
SHAPE(ip, jp, kp, 8) = 08th * QP1 * EP1 * TP1
```

$$\begin{aligned}
 QP1(i) &= (1 + \xi_i), & QM1(i) &= (1 - \xi_i) \\
 EP1(j) &= (1 + \eta_j), & EM1(j) &= (1 - \eta_j) \\
 TP1(k) &= (1 + \zeta_k), & TM1(k) &= (1 - \zeta_k)
 \end{aligned}$$

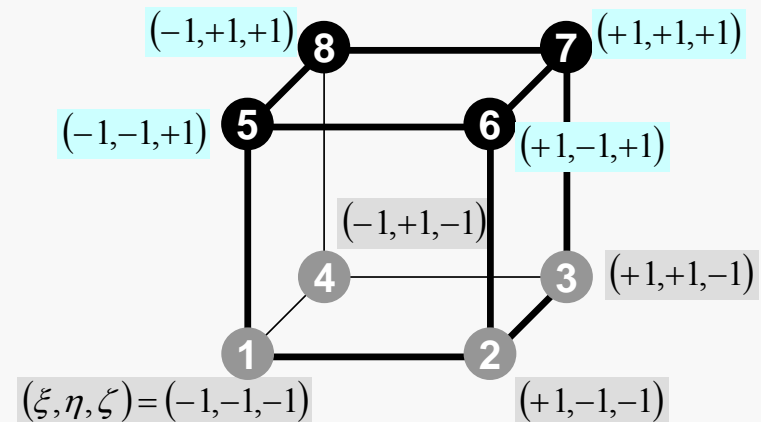
# MAT\_ASS\_MAIN (2/6)

```
!C
!C-- INIT.
!C   PNQ  - 1st-order derivative of shape function by QSI
!C   PNE  - 1st-order derivative of shape function by ETA
!C   PNT  - 1st-order derivative of shape function by ZET
!C
```

```
do kp= 1, 2
do jp= 1, 2
do ip= 1, 2
```

```
QP1= 1. d0 + POS(ip)
QM1= 1. d0 - POS(ip)
EP1= 1. d0 + POS(jp)
EM1= 1. d0 - POS(jp)
TP1= 1. d0 + POS(kp)
TM1= 1. d0 - POS(kp)
```

```
SHAPE(ip, jp, kp, 1) = 08th * QM1 * EM1 * TM1
SHAPE(ip, jp, kp, 2) = 08th * QP1 * EM1 * TM1
SHAPE(ip, jp, kp, 3) = 08th * QP1 * EP1 * TM1
SHAPE(ip, jp, kp, 4) = 08th * QM1 * EP1 * TM1
SHAPE(ip, jp, kp, 5) = 08th * QM1 * EM1 * TP1
SHAPE(ip, jp, kp, 6) = 08th * QP1 * EM1 * TP1
SHAPE(ip, jp, kp, 7) = 08th * QP1 * EP1 * TP1
SHAPE(ip, jp, kp, 8) = 08th * QP1 * EP1 * TP1
```



# MAT\_ASS\_MAIN (2/6)

```
!C
!C-- INIT.
!C   PNQ   - 1st-order derivative of shape function by QSI
!C   PNE   - 1st-order derivative of shape function by ETA
!C   PNT   - 1st-order derivative of shape function by ZET
!C
```

```
do kp= 1, 2
do jp= 1, 2
do ip= 1, 2
```

```
QP1= 1. d0 + POS (ip)
QM1= 1. d0 - POS (ip)
EP1= 1. d0 + POS (jp)
EM1= 1. d0 - POS (jp)
TP1= 1. d0 + POS (kp)
TM1= 1. d0 - POS (kp)
```

```
SHAPE (ip, jp, kp, 1) = 08th * QM1 * EM1 * TM1
SHAPE (ip, jp, kp, 2) = 08th * QP1 * EM1 * TM1
SHAPE (ip, jp, kp, 3) = 08th * QP1 * EP1 * TM1
SHAPE (ip, jp, kp, 4) = 08th * QM1 * EP1 * TM1
SHAPE (ip, jp, kp, 5) = 08th * QM1 * EM1 * TP1
SHAPE (ip, jp, kp, 6) = 08th * QP1 * EM1 * TP1
SHAPE (ip, jp, kp, 7) = 08th * QP1 * EP1 * TP1
SHAPE (ip, jp, kp, 8) = 08th * QP1 * EP1 * TP1
```

$$N_1(\xi, \eta, \zeta) = \frac{1}{8}(1-\xi)(1-\eta)(1-\zeta)$$

$$N_2(\xi, \eta, \zeta) = \frac{1}{8}(1+\xi)(1-\eta)(1-\zeta)$$

$$N_3(\xi, \eta, \zeta) = \frac{1}{8}(1+\xi)(1+\eta)(1-\zeta)$$

$$N_4(\xi, \eta, \zeta) = \frac{1}{8}(1-\xi)(1+\eta)(1-\zeta)$$

$$N_5(\xi, \eta, \zeta) = \frac{1}{8}(1-\xi)(1-\eta)(1+\zeta)$$

$$N_6(\xi, \eta, \zeta) = \frac{1}{8}(1+\xi)(1-\eta)(1+\zeta)$$

$$N_7(\xi, \eta, \zeta) = \frac{1}{8}(1+\xi)(1+\eta)(1+\zeta)$$

$$N_8(\xi, \eta, \zeta) = \frac{1}{8}(1-\xi)(1+\eta)(1+\zeta)$$

# MAT\_ASS\_MAIN (3/6)

```

PNQ (jp, kp, 1) = - 08th * EM1 * TM1
PNQ (jp, kp, 2) = + 08th * EM1 * TM1
PNQ (jp, kp, 3) = + 08th * EP1 * TM1
PNQ (jp, kp, 4) = - 08th * EP1 * TM1
PNQ (jp, kp, 5) = - 08th * EM1 * TP1
PNQ (jp, kp, 6) = + 08th * EM1 * TP1
PNQ (jp, kp, 7) = + 08th * EP1 * TP1
PNQ (jp, kp, 8) = - 08th * EP1 * TP1
PNE (ip, kp, 1) = - 08th * QM1 * TM1
PNE (ip, kp, 2) = - 08th * QP1 * TM1
PNE (ip, kp, 3) = + 08th * QP1 * TM1
PNE (ip, kp, 4) = + 08th * QM1 * TM1
PNE (ip, kp, 5) = - 08th * QM1 * TP1
PNE (ip, kp, 6) = - 08th * QP1 * TP1
PNE (ip, kp, 7) = + 08th * QP1 * TP1
PNE (ip, kp, 8) = + 08th * QM1 * TP1
PNT (ip, jp, 1) = - 08th * QM1 * EM1
PNT (ip, jp, 2) = - 08th * QP1 * EM1
PNT (ip, jp, 3) = - 08th * QP1 * EP1
PNT (ip, jp, 4) = - 08th * QM1 * EP1
PNT (ip, jp, 5) = + 08th * QM1 * EM1
PNT (ip, jp, 6) = + 08th * QP1 * EM1
PNT (ip, jp, 7) = + 08th * QP1 * EP1
PNT (ip, jp, 8) = + 08th * QM1 * EP1

```

```

enddo
enddo
enddo

```

```

do icel= 1, ICELTOT
  CONDO= COND

```

```

in1= ICELNOD (icel, 1)
in2= ICELNOD (icel, 2)
in3= ICELNOD (icel, 3)
in4= ICELNOD (icel, 4)
in5= ICELNOD (icel, 5)
in6= ICELNOD (icel, 6)
in7= ICELNOD (icel, 7)
in8= ICELNOD (icel, 8)

```

$$PNQ(j, k) = \frac{\partial N_l}{\partial \xi} (\xi = \xi_i, \eta = \eta_j, \zeta = \zeta_k)$$

$$PNE(i, k) = \frac{\partial N_l}{\partial \eta} (\xi = \xi_i, \eta = \eta_j, \zeta = \zeta_k)$$

$$PNT(i, j) = \frac{\partial N_l}{\partial \zeta} (\xi = \xi_i, \eta = \eta_j, \zeta = \zeta_k)$$

$$\frac{\partial N_1}{\partial \xi} (\xi_i, \eta_j, \zeta_k) = -\frac{1}{8} (1 - \eta_j) (1 - \zeta_k)$$

$$\frac{\partial N_2}{\partial \xi} (\xi_i, \eta_j, \zeta_k) = +\frac{1}{8} (1 - \eta_j) (1 - \zeta_k)$$

$$\frac{\partial N_3}{\partial \xi} (\xi_i, \eta_j, \zeta_k) = +\frac{1}{8} (1 + \eta_j) (1 - \zeta_k)$$

$$\frac{\partial N_3}{\partial \xi} (\xi_i, \eta_j, \zeta_k) = -\frac{1}{8} (1 + \eta_j) (1 - \zeta_k)$$

First Order Derivative  
of Shape Functions at  
( $\xi_i, \eta_j, \zeta_k$ )

# MAT\_ASS\_MAIN (3/6)

```

PNQ (jp, kp, 1) = - 08th * EM1 * TM1
PNQ (jp, kp, 2) = + 08th * EM1 * TM1
PNQ (jp, kp, 3) = + 08th * EP1 * TM1
PNQ (jp, kp, 4) = - 08th * EP1 * TM1
PNQ (jp, kp, 5) = - 08th * EM1 * TP1
PNQ (jp, kp, 6) = + 08th * EM1 * TP1
PNQ (jp, kp, 7) = + 08th * EP1 * TP1
PNQ (jp, kp, 8) = - 08th * EP1 * TP1
PNE (ip, kp, 1) = - 08th * QM1 * TM1
PNE (ip, kp, 2) = - 08th * QP1 * TM1
PNE (ip, kp, 3) = + 08th * QP1 * TM1
PNE (ip, kp, 4) = + 08th * QM1 * TM1
PNE (ip, kp, 5) = - 08th * QM1 * TP1
PNE (ip, kp, 6) = - 08th * QP1 * TP1
PNE (ip, kp, 7) = + 08th * QP1 * TP1
PNE (ip, kp, 8) = + 08th * QM1 * TP1
PNT (ip, jp, 1) = - 08th * QM1 * EM1
PNT (ip, jp, 2) = - 08th * QP1 * EM1
PNT (ip, jp, 3) = - 08th * QP1 * EP1
PNT (ip, jp, 4) = - 08th * QM1 * EP1
PNT (ip, jp, 5) = + 08th * QM1 * EM1
PNT (ip, jp, 6) = + 08th * QP1 * EM1
PNT (ip, jp, 7) = + 08th * QP1 * EP1
PNT (ip, jp, 8) = + 08th * QM1 * EP1

```

```

endo
endo
endo

```

```

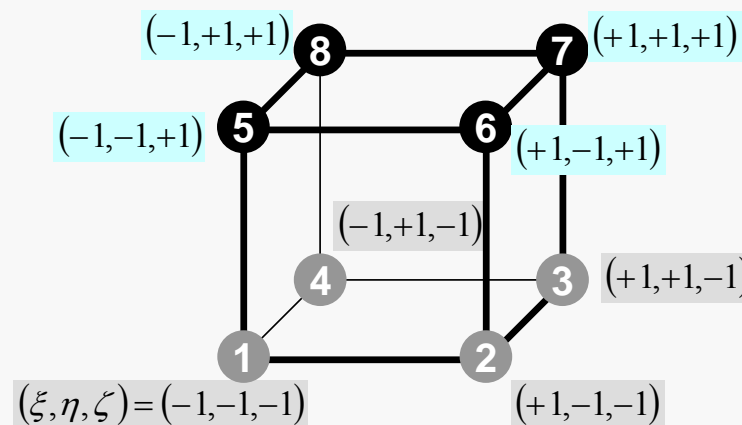
do icel= 1, ICELTOT
  CONDO= COND

```

```

in1= ICELNOD (icel, 1)
in2= ICELNOD (icel, 2)
in3= ICELNOD (icel, 3)
in4= ICELNOD (icel, 4)
in5= ICELNOD (icel, 5)
in6= ICELNOD (icel, 6)
in7= ICELNOD (icel, 7)
in8= ICELNOD (icel, 8)

```



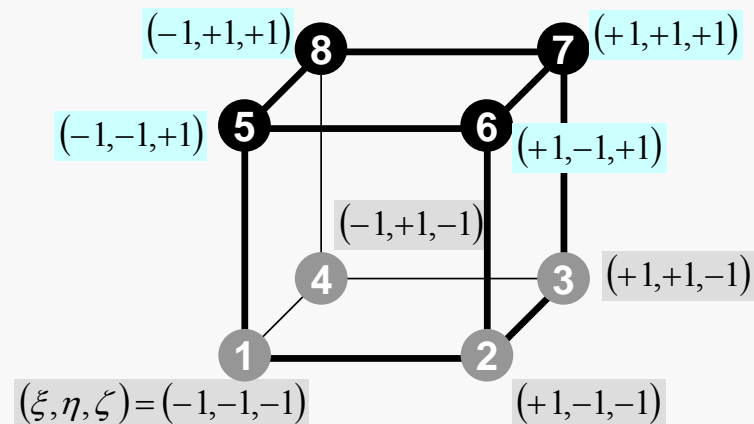
# MAT\_ASS\_MAIN (4/6)

```

nodLOCAL (1) = in1
nodLOCAL (2) = in2
nodLOCAL (3) = in3
nodLOCAL (4) = in4
nodLOCAL (5) = in5
nodLOCAL (6) = in6
nodLOCAL (7) = in7
nodLOCAL (8) = in8

```

Node ID (Global)



```

X1= XYZ (in1, 1)
X2= XYZ (in2, 1)
X3= XYZ (in3, 1)
X4= XYZ (in4, 1)
X5= XYZ (in5, 1)
X6= XYZ (in6, 1)
X7= XYZ (in7, 1)
X8= XYZ (in8, 1)
Y1= XYZ (in1, 2)
Y2= XYZ (in2, 2)
Y3= XYZ (in3, 2)
Y4= XYZ (in4, 2)
Y5= XYZ (in5, 2)
Y6= XYZ (in6, 2)
Y7= XYZ (in7, 2)
Y8= XYZ (in8, 2)
QVC= 08th * (X1+X2+X3+X4+X5+X6+X7+X8+
&           Y1+Y2+Y3+Y4+Y5+Y6+Y7+Y8)
Z1= XYZ (in1, 3)
Z2= XYZ (in2, 3)
Z3= XYZ (in3, 3)
Z4= XYZ (in4, 3)
Z5= XYZ (in5, 3)
Z6= XYZ (in6, 3)
Z7= XYZ (in7, 3)
Z8= XYZ (in8, 3)

call 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 )

```

&  
&  
&



# MAT\_ASS\_MAIN (4/6)

```

nodLOCAL (1) = in1
nodLOCAL (2) = in2
nodLOCAL (3) = in3
nodLOCAL (4) = in4
nodLOCAL (5) = in5
nodLOCAL (6) = in6
nodLOCAL (7) = in7
nodLOCAL (8) = in8

```

```

X1= XYZ (in1, 1)
X2= XYZ (in2, 1)
X3= XYZ (in3, 1)
X4= XYZ (in4, 1)
X5= XYZ (in5, 1)
X6= XYZ (in6, 1)
X7= XYZ (in7, 1)
X8= XYZ (in8, 1)

```

X-Coordinates  
of 8 nodes

```

Y1= XYZ (in1, 2)
Y2= XYZ (in2, 2)
Y3= XYZ (in3, 2)
Y4= XYZ (in4, 2)
Y5= XYZ (in5, 2)
Y6= XYZ (in6, 2)
Y7= XYZ (in7, 2)
Y8= XYZ (in8, 2)

```

Y-Coordinates  
of 8 nodes

```

& QVC= 08th * (X1+X2+X3+X4+X5+X6+X7+X8+
& Y1+Y2+Y3+Y4+Y5+Y6+Y7+Y8)

```

```

Z1= XYZ (in1, 3)
Z2= XYZ (in2, 3)
Z3= XYZ (in3, 3)
Z4= XYZ (in4, 3)
Z5= XYZ (in5, 3)
Z6= XYZ (in6, 3)
Z7= XYZ (in7, 3)
Z8= XYZ (in8, 3)

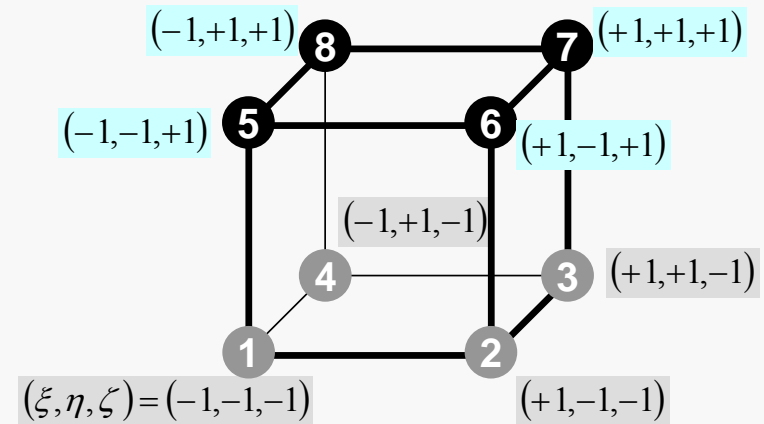
```

Z-Coordinates  
of 8 nodes

```

& call 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 )

```



# MAT\_ASS\_MAIN (4/6)

```

nodLOCAL (1)= in1
nodLOCAL (2)= in2
nodLOCAL (3)= in3
nodLOCAL (4)= in4
nodLOCAL (5)= in5
nodLOCAL (6)= in6
nodLOCAL (7)= in7
nodLOCAL (8)= in8

```

```

X1= XYZ(in1, 1)
X2= XYZ(in2, 1)
X3= XYZ(in3, 1)
X4= XYZ(in4, 1)
X5= XYZ(in5, 1)
X6= XYZ(in6, 1)
X7= XYZ(in7, 1)
X8= XYZ(in8, 1)

```

X-Coordinates  
of 8 nodes

```

Y1= XYZ(in1, 2)
Y2= XYZ(in2, 2)
Y3= XYZ(in3, 2)
Y4= XYZ(in4, 2)
Y5= XYZ(in5, 2)
Y6= XYZ(in6, 2)
Y7= XYZ(in7, 2)
Y8= XYZ(in8, 2)

```

Y-Coordinates  
of 8 nodes

```

& QVC= 08th * (X1+X2+X3+X4+X5+X6+X7+X8+
              Y1+Y2+Y3+Y4+Y5+Y6+Y7+Y8)

```

```

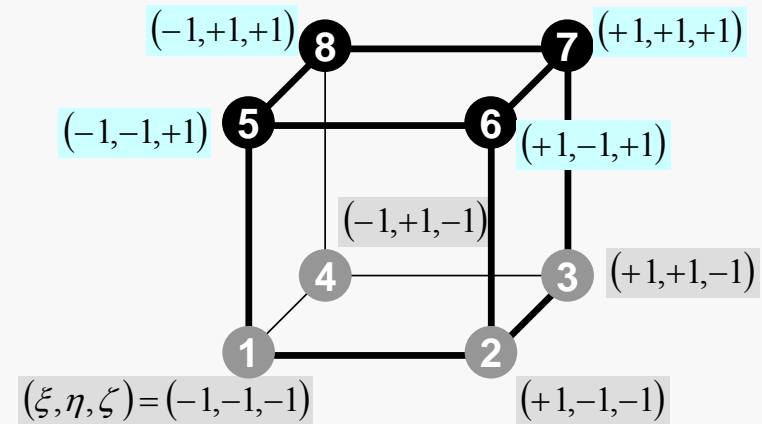
Z1= XYZ(in1, 3)
Z2= XYZ(in2, 3)
Z3= XYZ(in3, 3)
Z4= XYZ(in4, 3)
Z5= XYZ(in5, 3)
Z6= XYZ(in6, 3)
Z7= XYZ(in7, 3)
Z8= XYZ(in8, 3)

```

```

& call JACOBI (DETJ, PNQ, PNE, PNT, PNQ, 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 )

```



$$\frac{\partial}{\partial x} \left( \lambda \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left( \lambda \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left( \lambda \frac{\partial T}{\partial z} \right) + \dot{Q}(x, y, z) = 0$$

$$\dot{Q}(x, y, z) = QVOL |x_c + y_c|$$

Heat Gen. Rate is a function of location  
(cell center:  $x_c, y_c$ )

# MAT\_ASS\_MAIN (4/6)

```

nodLOCAL (1)= in1
nodLOCAL (2)= in2
nodLOCAL (3)= in3
nodLOCAL (4)= in4
nodLOCAL (5)= in5
nodLOCAL (6)= in6
nodLOCAL (7)= in7
nodLOCAL (8)= in8

```

```

X1= XYZ (in1, 1)
X2= XYZ (in2, 1)
X3= XYZ (in3, 1)
X4= XYZ (in4, 1)
X5= XYZ (in5, 1)
X6= XYZ (in6, 1)
X7= XYZ (in7, 1)
X8= XYZ (in8, 1)
Y1= XYZ (in1, 2)
Y2= XYZ (in2, 2)
Y3= XYZ (in3, 2)
Y4= XYZ (in4, 2)
Y5= XYZ (in5, 2)
Y6= XYZ (in6, 2)
Y7= XYZ (in7, 2)
Y8= XYZ (in8, 2)

```

```

& QVC= 08th * (X1+X2+X3+X4+X5+X6+X7+X8+
Y1+Y2+Y3+Y4+Y5+Y6+Y7+Y8)

```

```

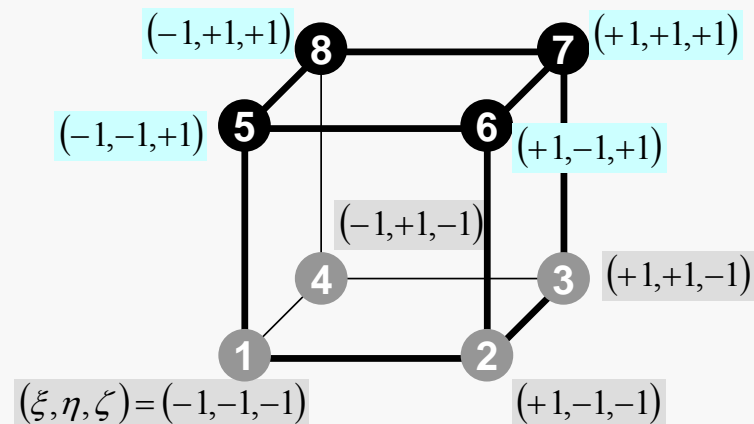
Z1= XYZ (in1, 3)
Z2= XYZ (in2, 3)
Z3= XYZ (in3, 3)
Z4= XYZ (in4, 3)
Z5= XYZ (in5, 3)
Z6= XYZ (in6, 3)
Z7= XYZ (in7, 3)
Z8= XYZ (in8, 3)

```

```

& call 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 )

```



$$\frac{\partial}{\partial x} \left( \lambda \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left( \lambda \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left( \lambda \frac{\partial T}{\partial z} \right) + \dot{Q}(x, y, z) = 0$$

$$\dot{Q}(x, y, z) = QVOL |x_c + y_c|$$

$$QVC = |x_c + y_c|$$

# MAT\_ASS\_MAIN (4/6)

```

nodLOCAL (1)= in1
nodLOCAL (2)= in2
nodLOCAL (3)= in3
nodLOCAL (4)= in4
nodLOCAL (5)= in5
nodLOCAL (6)= in6
nodLOCAL (7)= in7
nodLOCAL (8)= in8

X1= XYZ (in1, 1)
X2= XYZ (in2, 1)
X3= XYZ (in3, 1)
X4= XYZ (in4, 1)
X5= XYZ (in5, 1)
X6= XYZ (in6, 1)
X7= XYZ (in7, 1)
X8= XYZ (in8, 1)
Y1= XYZ (in1, 2)
Y2= XYZ (in2, 2)
Y3= XYZ (in3, 2)
Y4= XYZ (in4, 2)
Y5= XYZ (in5, 2)
Y6= XYZ (in6, 2)
Y7= XYZ (in7, 2)
Y8= XYZ (in8, 2)
QVC= 08th * (X1+X2+X3+X4+X5+X6+X7+X8+
&          Y1+Y2+Y3+Y4+Y5+Y6+Y7+Y8)
Z1= XYZ (in1, 3)
Z2= XYZ (in2, 3)
Z3= XYZ (in3, 3)
Z4= XYZ (in4, 3)
Z5= XYZ (in5, 3)
Z6= XYZ (in6, 3)
Z7= XYZ (in7, 3)
Z8= XYZ (in8, 3)

& call JACOBI (DETJ, PNO, PNE, PNT, PNQ, 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 )
&

```

# JACOBI (1/4)

```
subroutine JACOBI (DETJ, PNQ, PNE, PNT, PNQ, 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 )
```

```
!C
!C
!C
!C
```

```
calculates JACOBIAN & INVERSE JACOBIAN
dNi/dx, dNi/dy & dNi/dz
```

```
implicit REAL*8 (A-H, O-Z)
dimension DETJ(2, 2, 2)
dimension PNQ(2, 2, 8), PNE(2, 2, 8), PNT(2, 2, 8)
dimension PNQ(2, 2, 2, 8), PNY(2, 2, 2, 8), PNZ(2, 2, 2, 8)
```

```
do kp= 1, 2
do jp= 1, 2
do ip= 1, 2
PNX(ip, jp, kp, 1)=0. d0
PNX(ip, jp, kp, 2)=0. d0
PNX(ip, jp, kp, 3)=0. d0
PNX(ip, jp, kp, 4)=0. d0
PNX(ip, jp, kp, 5)=0. d0
PNX(ip, jp, kp, 6)=0. d0
PNX(ip, jp, kp, 7)=0. d0
PNX(ip, jp, kp, 8)=0. d0
PNY(ip, jp, kp, 1)=0. d0
PNY(ip, jp, kp, 2)=0. d0
PNY(ip, jp, kp, 3)=0. d0
PNY(ip, jp, kp, 4)=0. d0
PNY(ip, jp, kp, 5)=0. d0
PNY(ip, jp, kp, 6)=0. d0
PNY(ip, jp, kp, 7)=0. d0
PNY(ip, jp, kp, 8)=0. d0
PNZ(ip, jp, kp, 1)=0. d0
PNZ(ip, jp, kp, 2)=0. d0
PNZ(ip, jp, kp, 3)=0. d0
PNZ(ip, jp, kp, 4)=0. d0
PNZ(ip, jp, kp, 5)=0. d0
PNZ(ip, jp, kp, 6)=0. d0
PNZ(ip, jp, kp, 7)=0. d0
PNZ(ip, jp, kp, 8)=0. d0
```

Input

$$\left[ \frac{\partial N_l}{\partial \xi}, \frac{\partial N_l}{\partial \eta}, \frac{\partial N_l}{\partial \zeta} \right], (x_l, y_l, z_l) (l = 1 \sim 8)$$

Output

$$\left[ \frac{\partial N_l}{\partial x}, \frac{\partial N_l}{\partial y}, \frac{\partial N_l}{\partial z} \right], \det|J|$$

**Values at each Gaussian Quad.  
Points: (ip,jp,kp)**

# Partial Diff. on Natural Coord. (1/4)

- According to formulae:

$$\frac{\partial N_i(\xi, \eta, \zeta)}{\partial \xi} = \frac{\partial N_i}{\partial x} \frac{\partial x}{\partial \xi} + \frac{\partial N_i}{\partial y} \frac{\partial y}{\partial \xi} + \frac{\partial N_i}{\partial z} \frac{\partial z}{\partial \xi}$$

$$\frac{\partial N_i(\xi, \eta, \zeta)}{\partial \eta} = \frac{\partial N_i}{\partial x} \frac{\partial x}{\partial \eta} + \frac{\partial N_i}{\partial y} \frac{\partial y}{\partial \eta} + \frac{\partial N_i}{\partial z} \frac{\partial z}{\partial \eta}$$

$$\frac{\partial N_i(\xi, \eta, \zeta)}{\partial \zeta} = \frac{\partial N_i}{\partial x} \frac{\partial x}{\partial \zeta} + \frac{\partial N_i}{\partial y} \frac{\partial y}{\partial \zeta} + \frac{\partial N_i}{\partial z} \frac{\partial z}{\partial \zeta}$$

$\left[ \frac{\partial N_i}{\partial \xi}, \frac{\partial N_i}{\partial \eta}, \frac{\partial N_i}{\partial \zeta} \right]$  can be easily derived according to definitions.

$\left[ \frac{\partial N_i}{\partial x}, \frac{\partial N_i}{\partial y}, \frac{\partial N_i}{\partial z} \right]$  are required for computations.

# Partial Diff. on Natural Coord. (2/4)

- In matrix form:

$$\begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \zeta} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{Bmatrix} = [J] \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{Bmatrix}$$

$$[J] = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix}$$

$[J]$  : Jacobi matrix, Jacobian

# Partial Diff. on Natural Coord. (3/4)

- Components of Jacobian:

$$J_{11} = \frac{\partial x}{\partial \xi} = \frac{\partial}{\partial \xi} \left( \sum_{i=1}^8 N_i x_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \xi} x_i, \quad J_{12} = \frac{\partial y}{\partial \xi} = \frac{\partial}{\partial \xi} \left( \sum_{i=1}^8 N_i y_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \xi} y_i,$$

$$J_{13} = \frac{\partial z}{\partial \xi} = \frac{\partial}{\partial \xi} \left( \sum_{i=1}^8 N_i z_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \xi} z_i$$

$$J_{21} = \frac{\partial x}{\partial \eta} = \frac{\partial}{\partial \eta} \left( \sum_{i=1}^8 N_i x_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \eta} x_i, \quad J_{22} = \frac{\partial y}{\partial \eta} = \frac{\partial}{\partial \eta} \left( \sum_{i=1}^8 N_i y_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \eta} y_i,$$

$$J_{23} = \frac{\partial z}{\partial \eta} = \frac{\partial}{\partial \eta} \left( \sum_{i=1}^8 N_i z_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \eta} z_i$$

$$J_{31} = \frac{\partial x}{\partial \zeta} = \frac{\partial}{\partial \zeta} \left( \sum_{i=1}^8 N_i x_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \zeta} x_i, \quad J_{32} = \frac{\partial y}{\partial \zeta} = \frac{\partial}{\partial \zeta} \left( \sum_{i=1}^8 N_i y_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \zeta} y_i,$$

$$J_{33} = \frac{\partial z}{\partial \zeta} = \frac{\partial}{\partial \zeta} \left( \sum_{i=1}^8 N_i z_i \right) = \sum_{i=1}^8 \frac{\partial N_i}{\partial \zeta} z_i$$





# JACOBI (3/4)

```

dXdT =
&      + PNT (ip, jp, 1) * X1 + PNT (ip, jp, 2) * X2
&      + PNT (ip, jp, 3) * X3 + PNT (ip, jp, 4) * X4
&      + PNT (ip, jp, 5) * X5 + PNT (ip, jp, 6) * X6
&      + PNT (ip, jp, 7) * X7 + PNT (ip, jp, 8) * X8

dYdT =
&      + PNT (ip, jp, 1) * Y1 + PNT (ip, jp, 2) * Y2
&      + PNT (ip, jp, 3) * Y3 + PNT (ip, jp, 4) * Y4
&      + PNT (ip, jp, 5) * Y5 + PNT (ip, jp, 6) * Y6
&      + PNT (ip, jp, 7) * Y7 + PNT (ip, jp, 8) * Y8

dZdT =
&      + PNT (ip, jp, 1) * Z1 + PNT (ip, jp, 2) * Z2
&      + PNT (ip, jp, 3) * Z3 + PNT (ip, jp, 4) * Z4
&      + PNT (ip, jp, 5) * Z5 + PNT (ip, jp, 6) * Z6
&      + PNT (ip, jp, 7) * Z7 + PNT (ip, jp, 8) * Z8

&
&
&
&
DETJ (ip, jp, kp) = dXdQ*(dYdE*dZdT-dZdE*dYdT) +
&                  dYdQ*(dZdE*dXdT-dXdE*dZdT) +
&                  dZdQ*(dXdE*dYdT-dYdE*dXdT)

```

```

!C
!C==

```

INVERSE JACOBIAN

coef= 1. d0 / DETJ (ip, jp, kp)

a11= coef \* ( dYdE\*dZdT - dZdE\*dYdT )

a12= coef \* ( dZdQ\*dYdT - dYdQ\*dZdT )

a13= coef \* ( dYdQ\*dZdE - dZdQ\*dYdE )

a21= coef \* ( dZdE\*dXdT - dXdE\*dZdT )

a22= coef \* ( dXdQ\*dZdT - dZdQ\*dXdT )

a23= coef \* ( dZdQ\*dXdE - dXdQ\*dZdE )

a31= coef \* ( dXdE\*dYdT - dYdE\*dXdT )

a32= coef \* ( dYdQ\*dXdT - dXdQ\*dYdT )

a33= coef \* ( dXdQ\*dYdE - dYdQ\*dXdE )

DETJ (ip, jp, kp) = dabs (DETJ (ip, jp, kp))

$$[J] = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix}$$

# Partial Diff. on Natural Coord. (4/4)

- Partial differentiation on global coordinate system is introduced as follows (with inverse of Jacobian matrix ( $3 \times 3$ ))

$$\left\{ \begin{array}{c} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{array} \right\} = \left[ \begin{array}{ccc} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{array} \right]^{-1} \left\{ \begin{array}{c} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \zeta} \end{array} \right\} = [J]^{-1} \left\{ \begin{array}{c} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \zeta} \end{array} \right\}$$

# JACOBI (3/4)

```

dXdT =
&      + PNT (ip, jp, 1) * X1 + PNT (ip, jp, 2) * X2
&      + PNT (ip, jp, 3) * X3 + PNT (ip, jp, 4) * X4
&      + PNT (ip, jp, 5) * X5 + PNT (ip, jp, 6) * X6
&      + PNT (ip, jp, 7) * X7 + PNT (ip, jp, 8) * X8

dYdT =
&      + PNT (ip, jp, 1) * Y1 + PNT (ip, jp, 2) * Y2
&      + PNT (ip, jp, 3) * Y3 + PNT (ip, jp, 4) * Y4
&      + PNT (ip, jp, 5) * Y5 + PNT (ip, jp, 6) * Y6
&      + PNT (ip, jp, 7) * Y7 + PNT (ip, jp, 8) * Y8

dZdT =
&      + PNT (ip, jp, 1) * Z1 + PNT (ip, jp, 2) * Z2
&      + PNT (ip, jp, 3) * Z3 + PNT (ip, jp, 4) * Z4
&      + PNT (ip, jp, 5) * Z5 + PNT (ip, jp, 6) * Z6
&      + PNT (ip, jp, 7) * Z7 + PNT (ip, jp, 8) * Z8

DETJ (ip, jp, kp) = dXdQ*(dYdE*dZdT-dZdE*dYdT) +
&                  dYdQ*(dZdE*dXdT-dXdE*dZdT) +
&                  dZdQ*(dXdE*dYdT-dYdE*dXdT)

```

```
!C
!C==
```

INVERSE JACOBIAN

coef= 1. d0 / DETJ(ip, jp, kp)

a11= coef \* ( dYdE\*dZdT - dZdE\*dYdT )

a12= coef \* ( dZdQ\*dYdT - dYdQ\*dZdT )

a13= coef \* ( dYdQ\*dZdE - dZdQ\*dYdE )

a21= coef \* ( dZdE\*dXdT - dXdE\*dZdT )

a22= coef \* ( dXdQ\*dZdT - dZdQ\*dXdT )

a23= coef \* ( dZdQ\*dXdE - dXdQ\*dZdE )

a31= coef \* ( dXdE\*dYdT - dYdE\*dXdT )

a32= coef \* ( dYdQ\*dXdT - dXdQ\*dYdT )

a33= coef \* ( dXdQ\*dYdE - dYdQ\*dXdE )

DETJ(ip, jp, kp)= dabs(DETJ(ip, jp, kp))

$$[J]^{-1} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

# JACOBI (4/4)

```

!C
!C== set the dNi/dX, dNi/dY & dNi/dZ components
  PNQ(ip, jp, kp, 1) = a11*PNQ(ip, kp, 1) + a12*PNE(ip, kp, 1) + a13*PNT(ip, jp, 1)
  PNQ(ip, jp, kp, 2) = a11*PNQ(ip, kp, 2) + a12*PNE(ip, kp, 2) + a13*PNT(ip, jp, 2)
  PNQ(ip, jp, kp, 3) = a11*PNQ(ip, kp, 3) + a12*PNE(ip, kp, 3) + a13*PNT(ip, jp, 3)
  PNQ(ip, jp, kp, 4) = a11*PNQ(ip, kp, 4) + a12*PNE(ip, kp, 4) + a13*PNT(ip, jp, 4)
  PNQ(ip, jp, kp, 5) = a11*PNQ(ip, kp, 5) + a12*PNE(ip, kp, 5) + a13*PNT(ip, jp, 5)
  PNQ(ip, jp, kp, 6) = a11*PNQ(ip, kp, 6) + a12*PNE(ip, kp, 6) + a13*PNT(ip, jp, 6)
  PNQ(ip, jp, kp, 7) = a11*PNQ(ip, kp, 7) + a12*PNE(ip, kp, 7) + a13*PNT(ip, jp, 7)
  PNQ(ip, jp, kp, 8) = a11*PNQ(ip, kp, 8) + a12*PNE(ip, kp, 8) + a13*PNT(ip, jp, 8)

  PNY(ip, jp, kp, 1) = a21*PNQ(ip, kp, 1) + a22*PNE(ip, kp, 1) + a23*PNT(ip, jp, 1)
  PNY(ip, jp, kp, 2) = a21*PNQ(ip, kp, 2) + a22*PNE(ip, kp, 2) + a23*PNT(ip, jp, 2)
  PNY(ip, jp, kp, 3) = a21*PNQ(ip, kp, 3) + a22*PNE(ip, kp, 3) + a23*PNT(ip, jp, 3)
  PNY(ip, jp, kp, 4) = a21*PNQ(ip, kp, 4) + a22*PNE(ip, kp, 4) + a23*PNT(ip, jp, 4)
  PNY(ip, jp, kp, 5) = a21*PNQ(ip, kp, 5) + a22*PNE(ip, kp, 5) + a23*PNT(ip, jp, 5)
  PNY(ip, jp, kp, 6) = a21*PNQ(ip, kp, 6) + a22*PNE(ip, kp, 6) + a23*PNT(ip, jp, 6)
  PNY(ip, jp, kp, 7) = a21*PNQ(ip, kp, 7) + a22*PNE(ip, kp, 7) + a23*PNT(ip, jp, 7)
  PNY(ip, jp, kp, 8) = a21*PNQ(ip, kp, 8) + a22*PNE(ip, kp, 8) + a23*PNT(ip, jp, 8)

  PNZ(ip, jp, kp, 1) = a31*PNQ(ip, kp, 1) + a32*PNE(ip, kp, 1) + a33*PNT(ip, jp, 1)
  PNZ(ip, jp, kp, 2) = a31*PNQ(ip, kp, 2) + a32*PNE(ip, kp, 2) + a33*PNT(ip, jp, 2)
  PNZ(ip, jp, kp, 3) = a31*PNQ(ip, kp, 3) + a32*PNE(ip, kp, 3) + a33*PNT(ip, jp, 3)
  PNZ(ip, jp, kp, 4) = a31*PNQ(ip, kp, 4) + a32*PNE(ip, kp, 4) + a33*PNT(ip, jp, 4)
  PNZ(ip, jp, kp, 5) = a31*PNQ(ip, kp, 5) + a32*PNE(ip, kp, 5) + a33*PNT(ip, jp, 5)
  PNZ(ip, jp, kp, 6) = a31*PNQ(ip, kp, 6) + a32*PNE(ip, kp, 6) + a33*PNT(ip, jp, 6)
  PNZ(ip, jp, kp, 7) = a31*PNQ(ip, kp, 7) + a32*PNE(ip, kp, 7) + a33*PNT(ip, jp, 7)
  PNZ(ip, jp, kp, 8) = a31*PNQ(ip, kp, 8) + a32*PNE(ip, kp, 8) + a33*PNT(ip, jp, 8)
enddo
enddo
enddo

```

$$\begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix}^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \zeta} \end{Bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \zeta} \end{Bmatrix}$$

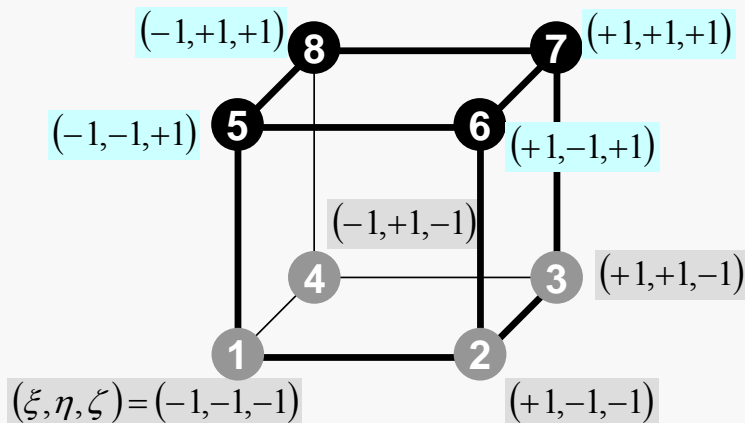
# MAT\_ASS\_MAIN (5/6)

```

!C
!C== CONSTRUCT the GLOBAL MATRIX
do ie= 1, 8
  ip = nodLOCAL (ie)
do je= 1, 8
  jp = nodLOCAL (je)

  kk= 0
  if (jp.ne.ip) then
    iiS= index(ip)-1 + 1
    iiE= index(ip )
    do k= iiS, iiE
      if ( item(k).eq.jp ) then
        kk= k
        exit
      endif
    enddo
  endif
endif

```



Non-Zero Off-Diagonal Block  
in Global Matix

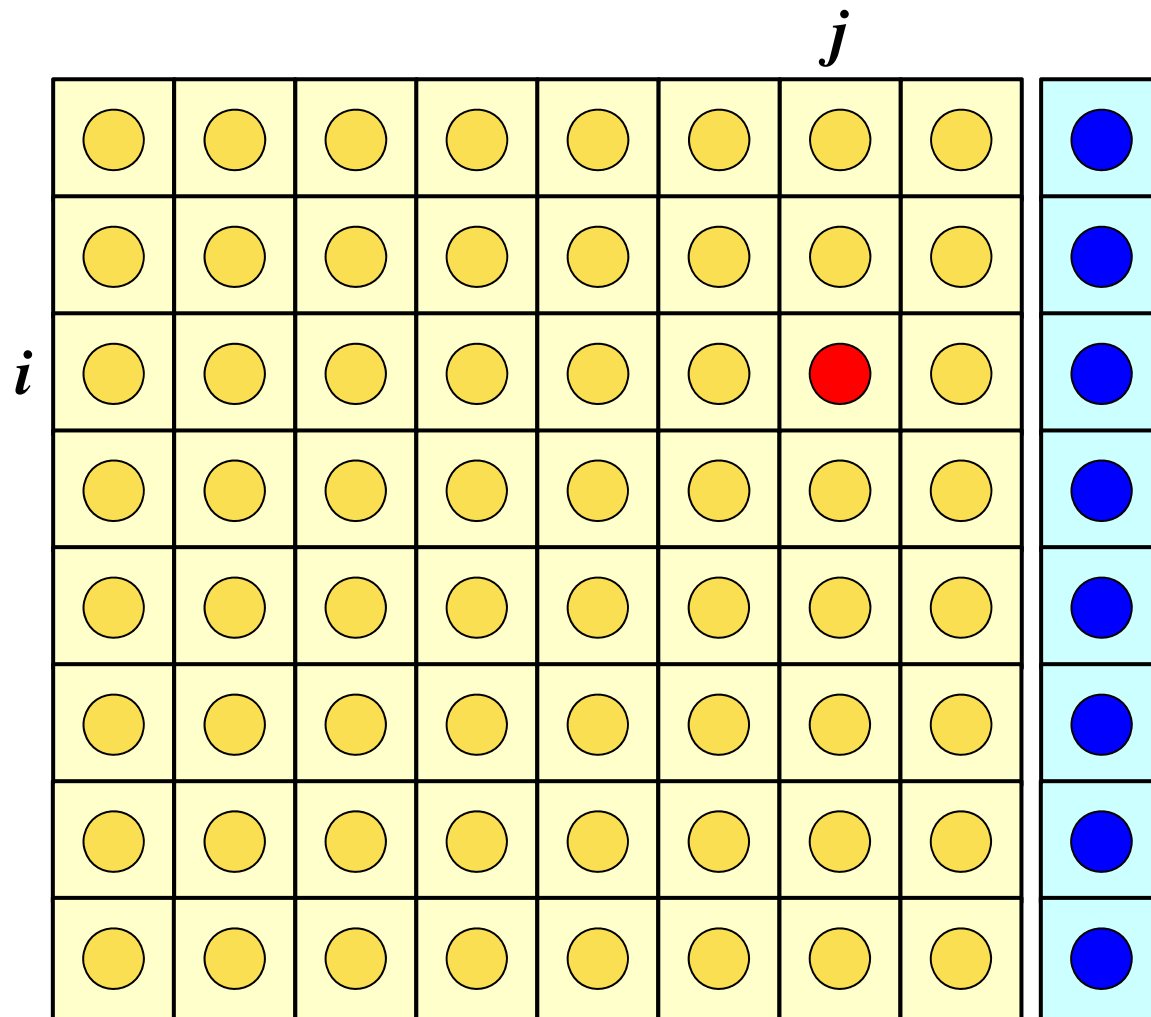
$$A_{ip, jp}$$

kk: address in "item"

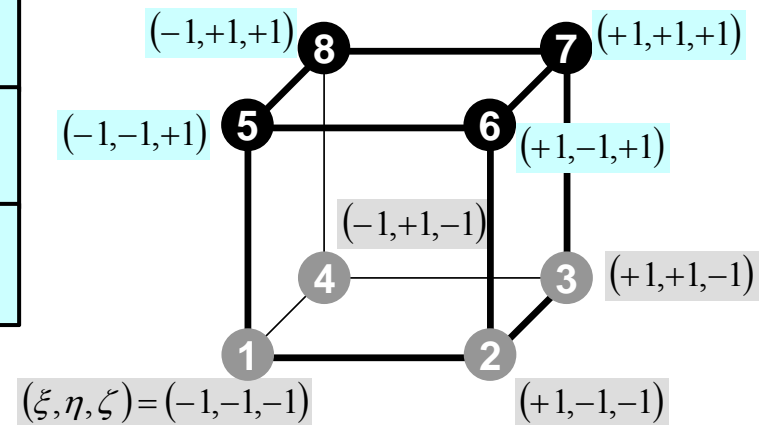
ip= nodLOCAL[ie]  
jp= nodLOCAL[je]

Node ID (ip,jp)  
starting from 1

# Element Matrix: 8x8



$$[k_{ij}] \quad (i, j = 1 \dots 8)$$



# MAT\_ASS\_MAIN (5/6)

```

!C
!C== CONSTRUCT the GLOBAL MATRIX
do ie= 1, 8
  ip = nodLOCAL (ie)
do je= 1, 8
  jp = nodLOCAL (je)

  kk= 0
  if (jp.ne.ip) then
    iiS= index(ip-1) + 1
    iiE= index(ip )
    do k= iiS, iiE
      if ( item(k).eq.jp ) then
        kk= k
        exit
      endif
    enddo
  endif
endif

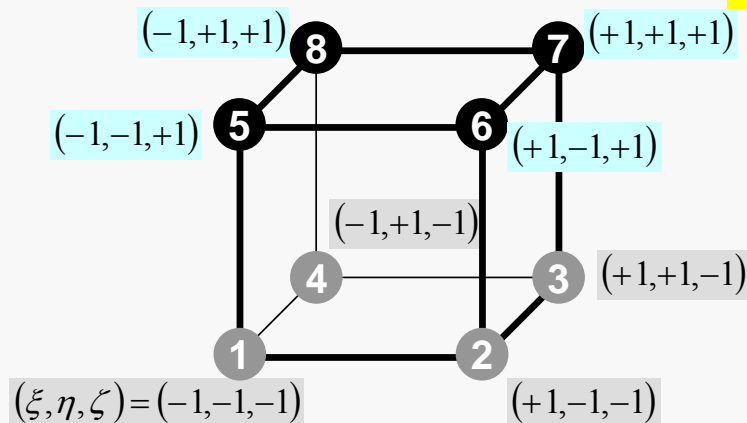
```

Element Matrix ( $i_e \sim j_e$ ): Local ID  
 Global Matrix ( $i_p \sim j_p$ ): Global ID

kk: address in "item" starting from "1"

k: starting from "1"

ip,jp: starting from "1"





# MAT\_ASS\_MAIN (6/6)

```

QVO = 0. d0
COEFij= 0. d0
do kpn= 1, 2
do jpn= 1, 2
do ipn= 1, 2
  coef= dabs (DETJ (ipn, jpn, kpn)) *WEI (ipn) *WEI (jpn) *WEI (kpn)

  PNXi= PNX (ipn, jpn, kpn, ie)
  PNYi= PNY (ipn, jpn, kpn, ie)
  PNZi= PNZ (ipn, jpn, kpn, ie)

  PNXj= PNX (ipn, jpn, kpn, je)
  PNYj= PNY (ipn, jpn, kpn, je)
  PNZj= PNZ (ipn, jpn, kpn, je)

  & COEFij= COEFij + coef * CONDO *
      (PNXi*PNXj+PNYi*PNYj+PNZi*PNZj)

  SHi= SHAPE (ipn, jpn, kpn, ie)
  QVO= QVO + SHi * QVOL * coef
enddo
enddo
enddo

if (jp. eq. ip) then
  D(ip)= D(ip) + COEFij
  B(ip)= B(ip) + QVO*QVC
else
  AMAT(kk)= AMAT(kk) + COEFij
endif
enddo
enddo
enddo

return
end

```

$$- \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} \left\{ \lambda \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \lambda \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + \lambda \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right\} \det|J| d\xi d\eta d\zeta$$

# MAT\_ASS\_MAIN (6/6)

```

QVO = 0. d0
COEFij= 0. d0
do kpn= 1, 2
do jpn= 1, 2
do ipn= 1, 2
  coef= dabs (DETJ (ipn, jpn, kpn)) *WEI (ipn) *WEI (jpn) *WEI (kpn)

  PNXi= PNX (ipn, jpn, kpn, ie)
  PNYi= PNY (ipn, jpn, kpn, ie)
  PNZi= PNZ (ipn, jpn, kpn, ie)

  PNXj= PNX (ipn, jpn, kpn, je)
  PNYj= PNY (ipn, jpn, kpn, je)
  PNZj= PNZ (ipn, jpn, kpn, je)

  & COEFij= COEFij + coef * CONDO *
      (PNXi*PNXj+PNYi*PNYj+PNZi*PNZj)

  SHi= SHAPE (ipn, jpn, kpn, ie)
  QVO= QVO + SHi * QVOL * coef
enddo
enddo
enddo

if (jp. eq. ip) then
  D(ip)= D(ip) + COEFij
  B(ip)= B(ip) + QVO*QVC
else
  AMAT(kk)= AMAT(kk) + COEFij
endif
enddo
enddo
enddo

return
end

```

$$\begin{aligned}
 I &= \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} f(\xi, \eta, \zeta) d\xi d\eta d\zeta \\
 &= \sum_{i=1}^L \sum_{j=1}^M \sum_{k=1}^N \boxed{W_i \cdot W_j \cdot W_k} \cdot \boxed{f(\xi_i, \eta_j, \zeta_k)}
 \end{aligned}$$

$$- \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} \left\{ \lambda \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \lambda \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + \lambda \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right\} \det|J| d\xi d\eta d\zeta$$

# MAT\_ASS\_MAIN (6/6)

```

QVO = 0. d0
COEFij= 0. d0
do kpn= 1, 2
do jpn= 1, 2
do ipn= 1, 2
  coef= dabs (DETJ (ipn, jpn, kpn)) *WEI (ipn) *WEI (jpn) *WEI (kpn)

  PNXi= PNx (ipn, jpn, kpn, ie)
  PNYi= PNY (ipn, jpn, kpn, ie)
  PNZi= PNz (ipn, jpn, kpn, ie)

  PNXj= PNx (ipn, jpn, kpn, je)
  PNYj= PNY (ipn, jpn, kpn, je)
  PNZj= PNz (ipn, jpn, kpn, je)

  & COEFij= COEFij + coef * CONDO *
      (PNXi*PNXj+PNYi*PNYj+PNZi*PNZj)

  SHi= SHAPE (ipn, jpn, kpn, ie)
  QVO= QVO + SHi * QVOL * coef
enddo
enddo
enddo

if (jp. eq. ip) then
  D(ip)= D(ip) + COEFij
  B(ip)= B(ip) + QVO*QVC
else
  AMAT(kk)= AMAT(kk) + COEFij
endif
enddo
enddo
enddo

return
end

```

$$\text{coef} = W_i \cdot W_j \cdot W_k \cdot \det |J(\xi_i, \eta_j, \zeta_k)|$$

$$\begin{aligned}
 I &= \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} f(\xi, \eta, \zeta) d\xi d\eta d\zeta \\
 &= \sum_{i=1}^L \sum_{j=1}^M \sum_{k=1}^N W_i \cdot W_j \cdot W_k \cdot f(\xi_i, \eta_j, \zeta_k)
 \end{aligned}$$

$$- \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} \left\{ \lambda \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \lambda \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + \lambda \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right\} \det |J| d\xi d\eta d\zeta$$



# 系数行列：MAT\_ASS\_MAIN (6/6)

```

QV0 = 0. d0
COEFij= 0. d0
do kpn= 1, 2
do jpn= 1, 2
do ipn= 1, 2
  coef= dabs (DETJ (ipn, jpn, kpn)) *WEI (ipn) *WEI (jpn) *WEI (kpn)

  PNXi= PNX (ipn, jpn, kpn, ie)
  PNYi= PNY (ipn, jpn, kpn, ie)
  PNZi= PNZ (ipn, jpn, kpn, ie)

  PNXj= PNX (ipn, jpn, kpn, je)
  PNYj= PNY (ipn, jpn, kpn, je)
  PNZj= PNZ (ipn, jpn, kpn, je)

  & COEFij= COEFij + coef * CONDO *
      (PNXi*PNXj+PNYi*PNYj+PNZi*PNZj)

  SHi= SHAPE (ipn, jpn, kpn, ie)
  QV0= QV0 + SHi * QVOL * coef
enddo
enddo
enddo

if (jp. eq. ip) then
  D(ip)= D(ip) + COEFij
  B(ip)= B(ip) + QV0*QVC
else
  AMAT(kk)= AMAT(kk) + COEFij
endif
enddo
enddo
enddo

return
end

```

$$[k]^{(e)} \{\phi\}^{(e)} = \{f\}^{(e)}$$

$$\{f\}^{(e)} = \int_V \dot{Q} [N]^T dV$$

$$\dot{Q}(x, y, z) = QVOL |x_C + y_C|$$

$$QVC = |x_C + y_C|$$

$$QV0 = \int_V QVOL [N]^T dV$$

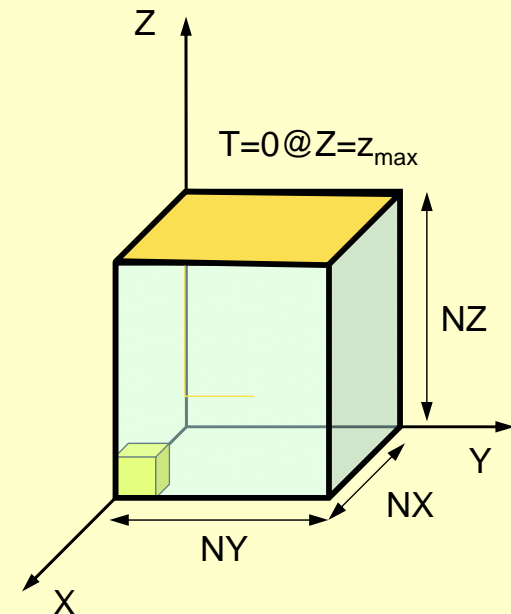
$$\{f\}^{(e)} = QV0 \cdot QVC$$

# MAT\_ASS\_BC: Overview

```
do i= 1, N    Loop for Nodes
  "Mark" nodes where Dirichlet B.C. are applied (IWKX)
enddo
```

```
do i= 1, N    Loop for Nodes
  if (IWKX(i,1).eq.1) then  if "marked" nodes
    corresponding components of RHS (B),
    Diagonal (D) are corrected
    do k= index(i-1)+1, index(i)  Non-Zero Off-Diagonal Nodes
      corresponding comp. of non-zero off-diagonal
      components (AMAT) are corrected
    enddo
  endif
enddo
```

```
do i= 1, N    Loop for Nodes
  do k= index(i-1)+1, index(i)  Non-Zero Off-Diagonal Nodes
    if (IWKX(item(k),1).eq.1) then
      corresponding components of RHS and AMAT are corrected (col.)
    endif
  enddo
enddo
```



# MAT\_ASS\_BC (1/2)

```
subroutine MAT_ASS_BC
use pfem_util
implicit REAL*8 (A-H, O-Z)

allocate (IWKX(N, 2))
IWKX= 0

!C
!C== Z=Zmax

do in= 1, N
  IWKX(in, 1)= 0
enddo

  ib0= -1
do ib0= 1, NODGRPtot
  if (NODGRP_NAME(ib0).eq.'Zmax') exit
enddo

do ib= NODGRP_INDEX(ib0-1)+1, NODGRP_INDEX(ib0)
  in= NODGRP_ITEM(ib)
  IWKX(in, 1)= 1
enddo
```

If the node “in” is included in the node group “Zmax”

$IWKX(in, 1) = 1$

# MAT\_ASS\_BC (2/2)

```
do in= 1, N
  if (IWKX(in,1).eq.1) then
    B(in)= 0.d0
    D(in)= 1.d0

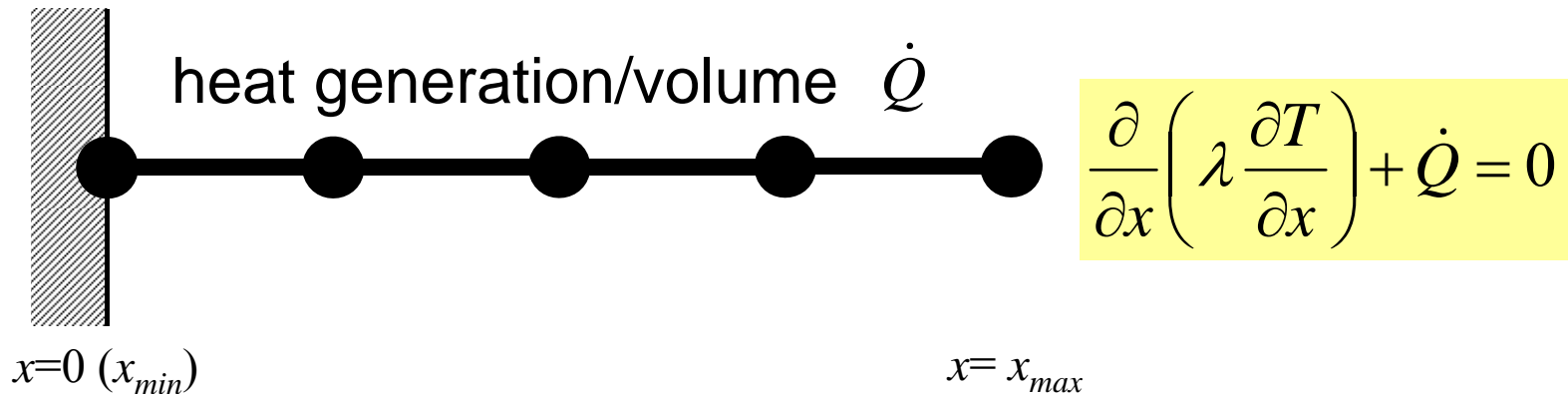
    iS= index(in-1) + 1
    iE= index(in )
    do k= iS, iE
      AMAT(k)= 0.d0
    enddo
  endif
enddo

do in= 1, N
  iS= index(in-1) + 1
  iE= index(in )
  do k= iS, iE
    if (IWKX(item(k),1).eq.1) then
      AMAT(k)= 0.d0
    endif
  enddo
enddo

!C==
return
end
```



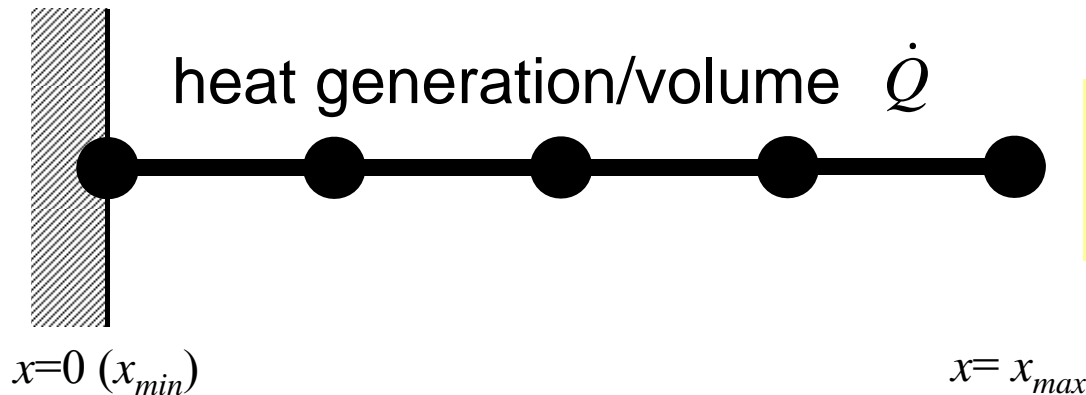
# 1D Steady State Heat Conduction



- **Uniform: Sectional Area:  $A$ , Thermal Conductivity:  $\lambda$**
- Heat Generation Rate/Volume/Time [ $QL^{-3}T^{-1}$ ]  $\dot{Q}$
- Boundary Conditions
  - $x=0$  :  $T=0$  (Fixed Temperature)
  - $x=x_{max}$  :  $\frac{\partial T}{\partial x} = 0$  (Insulated)

# (Linear) Equation at $x=0$

$$T_1 = 0 \text{ (or } T_0 = 0)$$



$$\frac{\partial}{\partial x} \left( \lambda \frac{\partial T}{\partial x} \right) + \dot{Q} = 0$$

- **Uniform: Sectional Area:  $A$ , Thermal Conductivity:  $\lambda$**
- Heat Generation Rate/Volume/Time [ $QL^{-3}T^{-1}$ ]  $\dot{Q}$
- Boundary Conditions
  - $x=0$  :  $T=0$  (Fixed Temperature)
  - $x=x_{max}$  :  $\frac{\partial T}{\partial x} = 0$  (Insulated)

# Program: 1d.f (6/6)

## Dirichlet B.C. @ X=0

```

!C
!C +-----+
!C | BOUNDARY CONDITIONS |
!C +-----+
!C===

!C
!C-- X=Xmin
      i= 1
      js= INDEX(i-1)

      AMAT (js+1)= 0. d0
      DIAG (i)= 1. d0
      RHS (i)= 0. d0

      do k= 1, NPLU
        if (ITEM(k).eq. 1) AMAT (k)= 0. d0
      enddo
!C===

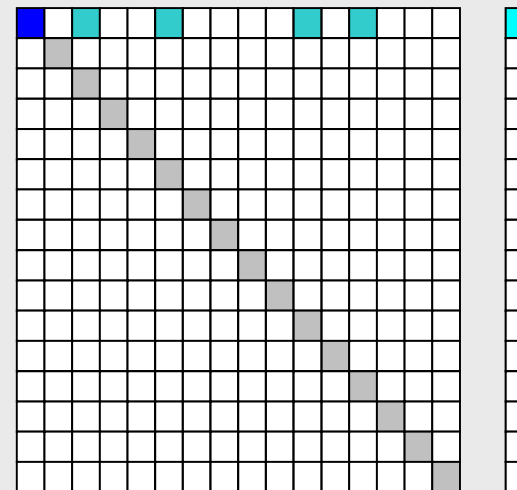
```

$$T_1=0$$

Diagonal Component=1

RHS=0

Off-Diagonal Components= 0.



# Program: 1d.f (6/6)

## Dirichlet B.C. @ X=0

```

!C
!C +-----+
!C | BOUNDARY CONDITIONS |
!C +-----+
!C===

!C
!C-- X=Xmin
      i= 1
      jS= INDEX(i-1)

      AMAT(jS+1)= 0. d0
      DIAG(i)= 1. d0
      RHS (i)= 0. d0

      do k= 1, NPLU
        if (ITEM(k).eq. 1) AMAT(k)= 0. d0
      enddo
!C===

```

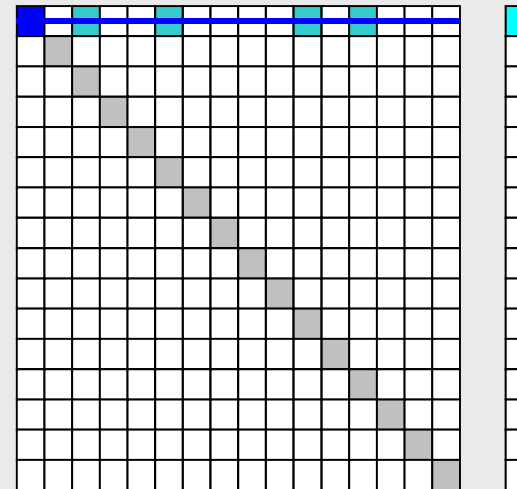
$$T_1=0$$

Diagonal Component=1

RHS=0

Off-Diagonal Components= 0.

Erase !



# Program: 1d.f (6/6)

## Dirichlet B.C. @ X=0

```

!C
!C +-----+
!C | BOUNDARY CONDITIONS |
!C +-----+
!C===

!C
!C-- X=Xmin
  i= 1
  js= INDEX(i-1)

  AMAT (js+1)= 0. d0
  DIAG (i)= 1. d0
  RHS (i)= 0. d0

  do k= 1, NPLU
    if (ITEM(k).eq.1) AMAT (k)= 0. d0
  enddo
!C===

```

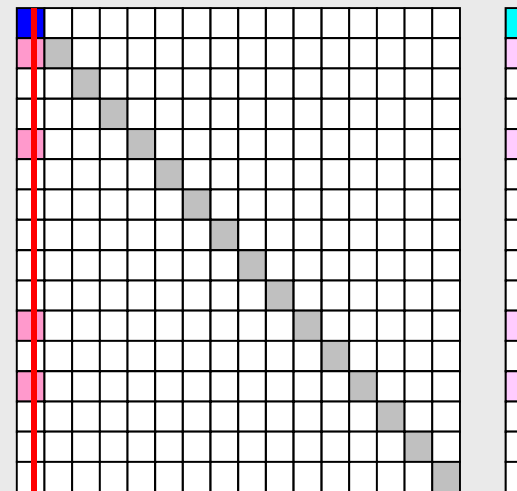
$$T_1=0$$

Diagonal Component=1

RHS=0

Off-Diagonal Components= 0.

### Elimination and Erase



Column components of boundary nodes (Dirichlet B.C.) are moved to RHS and eliminated for keeping symmetrical feature of the matrix (in this case just erase off-diagonal components)

# if $T_1 \neq 0$

```
!C
!C +-----+
!C | BOUNDARY CONDITIONS |
!C +-----+
!C===
```

Column components of boundary nodes (Dirichlet B.C.) are moved to RHS and eliminated for keeping symmetrical feature of the matrix.

```
!C
!C-- X=Xmin
      i= 1
      jS= INDEX(i-1)

      AMAT(jS+1)= 0. d0
      DIAG(i)= 1. d0
      RHS (i)= PHImin
```

$$Diag_j \phi_j + \sum_{k=Index[j]}^{Index[j+1]-1} Amat_k \phi_{Item[k]} = Rhs_j$$

```
do i= 1, N
  do k= INDEX(i-1)+1, INDEX(i)
    if (ITEM(k).eq.1) then
      RHS (i)= RHS(i) - AMAT(k)*PHImin
      AMAT(k)= 0. d0
    endif
  enddo
enddo
```

```
!C===
```

# if $T_1 \neq 0$

```
!C
!C +-----+
!C | BOUNDARY CONDITIONS |
!C +-----+
!C===
```

```
!C
!C-- X=Xmin
      i= 1
      jS= INDEX(i-1)
```

```
      AMAT(jS+1)= 0. d0
      DIAG(i)= 1. d0
      RHS (i)= PHImin
```

```
      do i= 1, N
        do k= INDEX(i-1)+1, INDEX(i)
          if (ITEM(k).eq.1) then
            RHS (i)= RHS(i) - AMAT(k)*PHImin
            AMAT(k)= 0. d0
          endif
        enddo
      enddo
```

```
!C===
```

$$\begin{aligned}
 & \text{Diag}_j \phi_j + \sum_{k=\text{Index}[j], k \neq k_s}^{\text{Index}[j+1]-1} \text{Amat}_k \phi_{\text{Item}[k]} \\
 &= \text{Rhs}_j - \text{Amat}_{k_s} \phi_{\text{Item}[k_s]} \\
 &= \text{Rhs}_j - \text{Amat}_{k_s} \phi_{\min} \quad \text{where } \text{Item}[k_s] = 0
 \end{aligned}$$

Column components of boundary nodes (Dirichlet B.C.) are moved to RHS and eliminated for keeping symmetrical feature of the matrix.

# MAT\_ASS\_BC (2/2)

```
do in= 1, N
  if (IWKX(in,1).eq.1) then
    B(in)= 0.d0
    D(in)= 1.d0
```

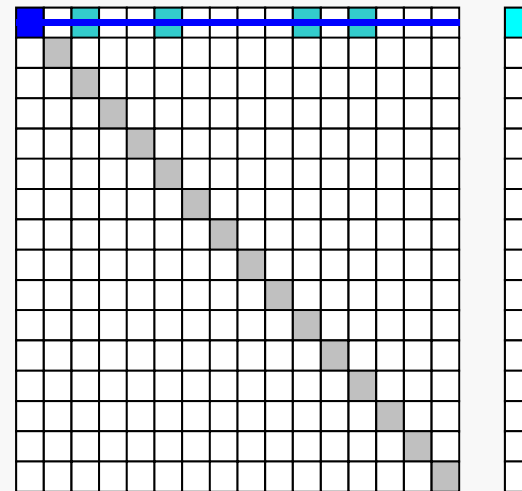
```
    iS= index(in-1) + 1
    iE= index(in )
    do k= iS, iE
      AMAT(k)= 0.d0
    enddo
  endif
enddo
```

```
do in= 1, N
  iS= index(in-1) + 1
  iE= index(in )
  do k= iS, iE
    if (IWKX(item(k),1).eq.1) then
      AMAT(k)= 0.d0
    endif
  enddo
enddo
```

```
!C==
return
end
```

Boundary Nodes: IWKX(in,1)=1

Erase !!



Same as 1D case



# MAT\_ASS\_BC (2/2)

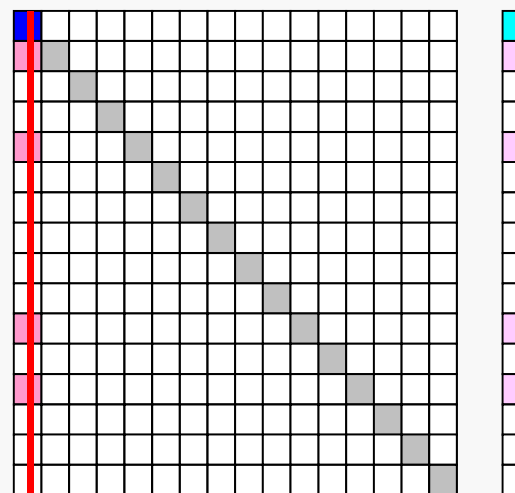
```
do in= 1, N
  if (IWKX(in,1).eq.1) then
    B(in)= 0. d0
    D(in)= 1. d0
```

```
    iS= index(in-1) + 1
    iE= index(in )
    do k= iS, iE
      AMAT(k)= 0. d0
    enddo
  endif
enddo
```

```
do in= 1, N
  iS= index(in-1) + 1
  iE= index(in )
  do k= iS, iE
    if (IWKX(item(k),1).eq.1) then
      AMAT(k)= 0. d0
    endif
  enddo
enddo
```

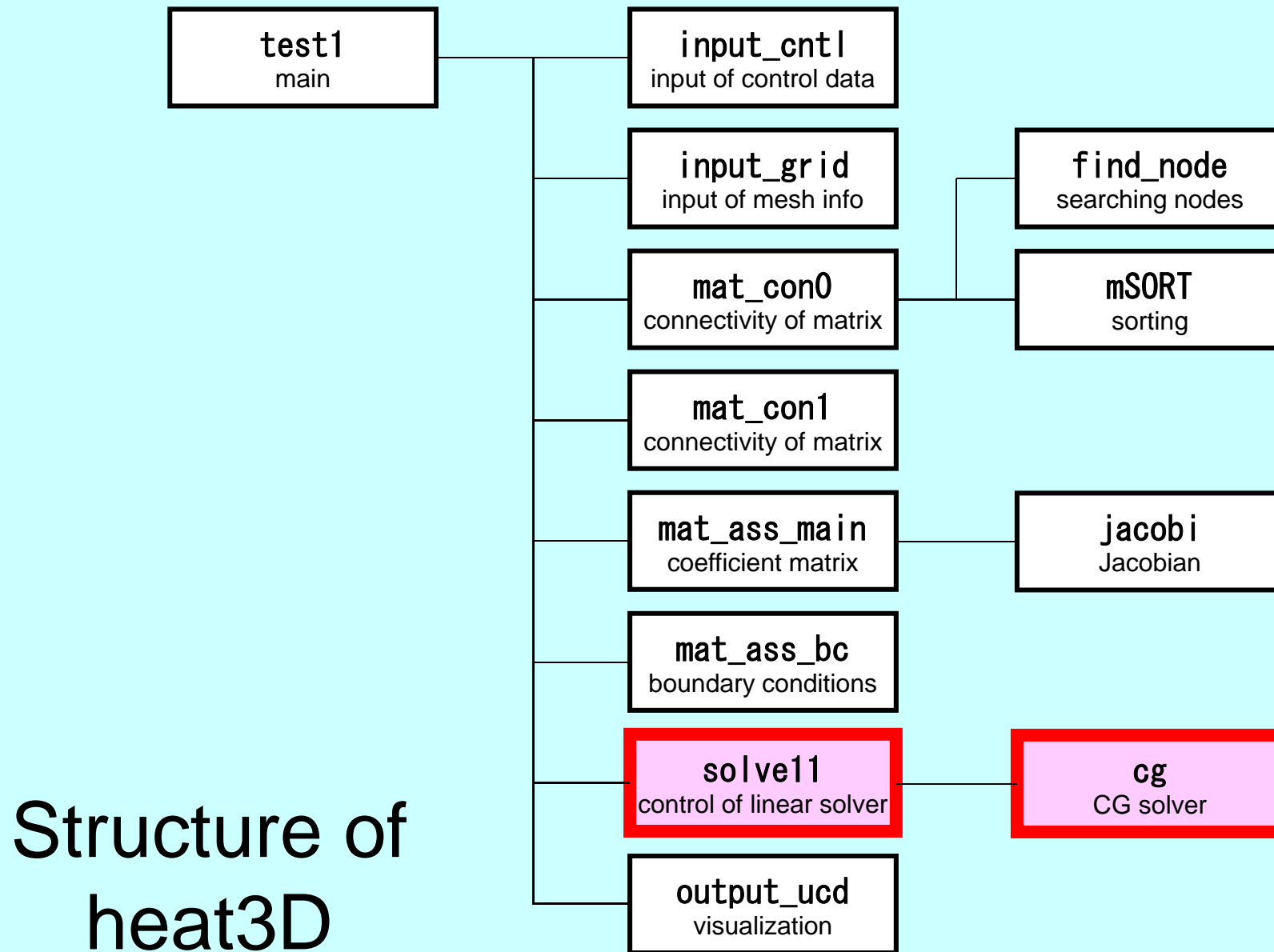
```
!C==
return
end
```

Boundary Nodes: IWKX(in,1)=1



**Elimination and Erase**

Same as 1D case



# Main Part

```
program heat3D
  use solver11
  use pfem_util

  implicit REAL*8(A-H, O-Z)

  call INPUT_CNTL
  call INPUT_GRID

  call MAT_CONO
  call MAT_CON1

  call MAT_ASS_MAIN
  call MAT_ASS_BC

  call SOLVE11

  call OUTPUT_UCD

end program heat3D
```

# SOLVE11

```

module SOLVER11
contains
  subroutine SOLVE11

    use pfem_util
    use solver_CG

    implicit REAL*8 (A-H, O-Z)

    integer :: ERROR, ICFLAG
    character(len=char_length) :: BUF

    data ICFLAG/0/

!C
!C +-----+
!C | PARAMETERS |
!C +-----+
!C==
    ITER      = pfemIarray(1)
    RESID     = pfemRarray(1)
!C==
!C
!C +-----+
!C | ITERATIVE solver |
!C +-----+
!C==
    call CG
    & ( N, NPLU, D, AMAT, index, item, B, X, RESID, ITER, ERROR ) &

    ITERactual= ITER
!C==

  end subroutine SOLVE11
end module SOLVER11

```

**ITER** = pfemIarray(1)                      Max. Iterations for CG  
**RESID** = pfemRarray(1)                      Convergence Criteria for CG

# Preconditioned CG Solver

## Diagonal Scaling/Point Jacobi Preconditioning

```

Compute  $\mathbf{r}^{(0)} = \mathbf{b} - [\mathbf{A}]\mathbf{x}^{(0)}$ 
for i= 1, 2, ...
  solve  $[\mathbf{M}]\mathbf{z}^{(i-1)} = \mathbf{r}^{(i-1)}$ 
   $\rho_{i-1} = \mathbf{r}^{(i-1)} \cdot \mathbf{z}^{(i-1)}$ 
  if i=1
     $\mathbf{p}^{(1)} = \mathbf{z}^{(0)}$ 
  else
     $\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$ 
     $\mathbf{p}^{(i)} = \mathbf{z}^{(i-1)} + \beta_{i-1} \mathbf{p}^{(i-1)}$ 
  endif
   $\mathbf{q}^{(i)} = [\mathbf{A}]\mathbf{p}^{(i)}$ 
   $\alpha_i = \rho_{i-1} / \mathbf{p}^{(i)} \cdot \mathbf{q}^{(i)}$ 
   $\mathbf{x}^{(i)} = \mathbf{x}^{(i-1)} + \alpha_i \mathbf{p}^{(i)}$ 
   $\mathbf{r}^{(i)} = \mathbf{r}^{(i-1)} - \alpha_i \mathbf{q}^{(i)}$ 
  check convergence  $|\mathbf{r}|$ 
end

```

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

# 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.

# CG Solver (1/6)

```

module solver_CG
contains

  subroutine CG                                     &
&    (N, NPLU, D, AMAT, index, item, B, X, RESID, ITER, ERROR)

  implicit REAL*8(A-H, O-Z)
  include 'precision.inc'

  integer(kind=kint ), intent(in):: N, NPLU
  integer(kind=kint ), intent(inout):: ITER, ERROR
  real    (kind=kreal), intent(inout):: RESID
  real(kind=kreal), dimension(N)    , intent(inout):: B, X, D
  real(kind=kreal), dimension(NPLU), intent(inout):: AMAT
  integer(kind=kint ), dimension(0:N ), intent(in)  :: index
  integer(kind=kint ), dimension(NPLU), intent(in)  :: item

  real(kind=kreal), dimension(:, :), allocatable   :: WW

  integer(kind=kint), parameter :: R= 1
  integer(kind=kint), parameter :: Z= 2
  integer(kind=kint), parameter :: Q= 2
  integer(kind=kint), parameter :: P= 3
  integer(kind=kint), parameter :: DD= 4

  integer(kind=kint ) :: MAXIT
  real    (kind=kreal) :: TOL, W, SS

```

# CG Solver (1/6)

```
module solver_CG
```

```

WW (i, 1) = WW (i, R)   ⇒ {r}
WW (i, 2) = WW (i, Z)   ⇒ {z}
WW (i, 2) = WW (i, Q)   ⇒ {q}
WW (i, 3) = WW (i, P)   ⇒ {p}
WW (i, 4) = WW (i, DD) ⇒ 1/{D}

```

```

integer(kind=kint), intent(inout):: ITER, LRROR
real (kind=kreal), intent(inout):: RESID
real(kind=kreal), dimension(N) , intent(inout)
real(kind=kreal), dimension(NPLU), intent(inout)
integer(kind=kint ), dimension(0:N ), intent(in)
integer(kind=kint ), dimension(NPLU), intent(in)

```

```
real(kind=kreal), dimension(:, :), allocatable
```

```

integer(kind=kint), parameter :: R= 1
integer(kind=kint), parameter :: Z= 2
integer(kind=kint), parameter :: Q= 2
integer(kind=kint), parameter :: P= 3
integer(kind=kint), parameter :: DD= 4

```

```

integer(kind=kint ) :: MAXIT
real (kind=kreal) :: TOL, W, SS

```

```

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

```



# CG Solver (2/6)

```

!C
!C +-----+
!C | INIT. |
!C +-----+
!C===
      ERROR= 0
      allocate (WW(N,4))
      MAXIT = ITER
      TOL   = RESID
      X = 0. d0
!C===

!C +-----+
!C | {r0}= {b} - [A] {xini} |
!C +-----+
!C===

      do j= 1, N
      WW(j,DD)= 1. d0/D(j)
      WVAL= B(j) - D(j)*X(j)
      do k= index(j-1)+1, index(j)
      i= item(k)
      WVAL= WVAL - AMAT(k)*X(i)
      enddo
      WW(j,R)= WVAL
      enddo

```

$$\begin{aligned}
 WW(i, 1) &= WW(i, R) &\Rightarrow \{r\} \\
 WW(i, 2) &= WW(i, Z) &\Rightarrow \{z\} \\
 WW(i, 2) &= WW(i, Q) &\Rightarrow \{q\} \\
 WW(i, 3) &= WW(i, P) &\Rightarrow \{p\} \\
 WW(i, 4) &= WW(i, DD) &\Rightarrow 1/\{D\}
 \end{aligned}$$

Reciprocal numbers (倒数) of diagonal components are stored in  $WW(i, DD)$ . Computational cost for division is usually expensive.

# CG Solver (2/6)

```

!C
!C +-----+
!C | INIT. |
!C +-----+
!C===
      ERROR= 0
      allocate (WW(N, 4))
      MAXIT = ITER
      TOL   = RESID
      X = 0. d0
!C===

!C +-----+
!C | {r0} = {b} - [A] {xini} |
!C +-----+
!C===
      do j= 1, N
        WW(j, DD)= 1. d0/D(j)
        WVAL= B(j) - D(j)*X(j)
        do k= index(j-1)+1, index(j)
          i= item(k)
          WVAL= WVAL - AMAT(k)*X(i)
        enddo
        WW(j, R)= WVAL
      enddo

```

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

```

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

```

# CG Solver (3/6)

```

BNRM20= 0. d0
do i= 1, N
  BNRM20= BNRM20 + B(i)**2
enddo

BNRM2= BNRM20

if (BNRM2. eq. 0. d0) BNRM2= 1. d0
ITER = 0
!C===

do iter= 1, MAXIT
!C
!C***** Conjugate Gradient Iteration

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

```

**BNRM2=|b|<sup>2</sup>  
for convergence criteria  
of CG solvers**

# CG Solver (3/6)

```

BNRM20= 0. d0
do i= 1, N
  BNRM20= BNRM20 + B(i)**2
enddo

BNRM2= BNRM20

if (BNRM2. eq. 0. d0) BNRM2= 1. d0
ITER = 0
!C===

do iter= 1, MAXIT
!C
!C*****
!C
!C +-----+
!C | {z}= [Minv] {r} |
!C +-----+
!C===
do i= 1, N
  WW(i, Z)= WW(i, R) * WW(i, DD)
enddo
!C===

```

```

Compute  $r^{(0)} = b - [A]x^{(0)}$ 
for  $i = 1, 2, \dots$ 
  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

```

# CG Solver (4/6)

```

!C
!C +-----+
!C | {RHO}= {r} {z} |
!C +-----+
!C===
      RH00= 0. d0
      do i= 1, N
        RH00= RH00 + WW(i, R)*WW(i, Z)
      enddo
      RHO= RH00
!C===
!C +-----+
!C | {p} = {z} if      ITER=1      |
!C | BETA= RHO / RH01  otherwise  |
!C +-----+
!C===
      if ( ITER. eq. 1 ) then
        do i= 1, N
          WW(i, P)= WW(i, Z)
        enddo
      else
        BETA= RHO / RH01
        do i= 1, N
          WW(i, P)= WW(i, Z) + BETA*WW(i, P)
        enddo
      endif
!C===

```

```

Compute  $r^{(0)} = b - [A]x^{(0)}$ 
for  $i = 1, 2, \dots$ 
  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

```

# CG Solver (5/6)

```

!C +-----+
!C | {q} = [A] {p} |
!C +-----+
!C===
      do j= 1, N
        WVAL= D(j)*WW(j, P)
        do k= index(j-1)+1, index(j)
          i= item(k)
          WVAL= WVAL + AMAT(k)*WW(i, P)
        enddo
        WW(j, Q)= WVAL
      enddo
!C===
!C +-----+
!C | ALPHA= RHO / {p} {q} |
!C +-----+
!C===
      C10= 0. d0
      do i= 1, N
        C10= C10 + WW(i, P)*WW(i, Q)
      enddo
      C1= C1
      ALPHA= RHO / C1
!C===

```

```

Compute  $r^{(0)} = b - [A]x^{(0)}$ 
for  $i = 1, 2, \dots$ 
  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

```

# CG Solver (6/6)

```

!C
!C +-----+
!C | {x} = {x} + ALPHA*{p} |
!C | {r} = {r} - ALPHA*{q} |
!C +-----+
!C===
      do i= 1, N
          X(i) = X (i)  + ALPHA * WW(i, P)
          WW(i, R) = WW(i, R) - ALPHA * WW(i, Q)
      enddo
!C===
      DNRM2= 0. d0
      do i= 1, N
          DNRM2= DNRM2 + WW(i, R)**2
      enddo
      DNRM2= DNRM2
      RESID= dsqrt(DNRM2/BNRM2)

      if ( RESID. le. TOL  ) exit
      if ( ITER .eq. MAXIT ) ERROR= -300

      RH01 = RHO
  enddo
!C===

```

```

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

```

# CG Solver (6/6)

```

!C
!C +-----+
!C | {x} = {x} + ALPHA*{p} |
!C | {r} = {r} - ALPHA*{q} |
!C +-----+
!C===
      do i= 1, N
          X(i) = X (i)  + ALPHA * WW(i,P)
          WW(i,R)= WW(i,R) - ALPHA * WW(i,Q)
      enddo
!C===
      DNRM2= 0. d0
      do i= 1, N
          DNRM2= DNRM2 + WW(i,R)**2
      enddo
      DNRM2= DNRM2
      RESID= dsqrt(DNRM2/BNRM2)

      if ( RESID.le.TOL  ) exit
      if ( ITER .eq. MAXIT ) ERROR= -300

      RH01 = RH0
    enddo
!C===

```

```

Compute  $r^{(0)} = b - [A]x^{(0)}$ 
for  $i = 1, 2, \dots$ 
  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

```

$$\text{Resid} = \sqrt{\frac{\text{DNorm2}}{\text{BNorm2}}} = \frac{|r|}{|b|} = \frac{|Ax - b|}{|b|} \leq \text{Tol}$$