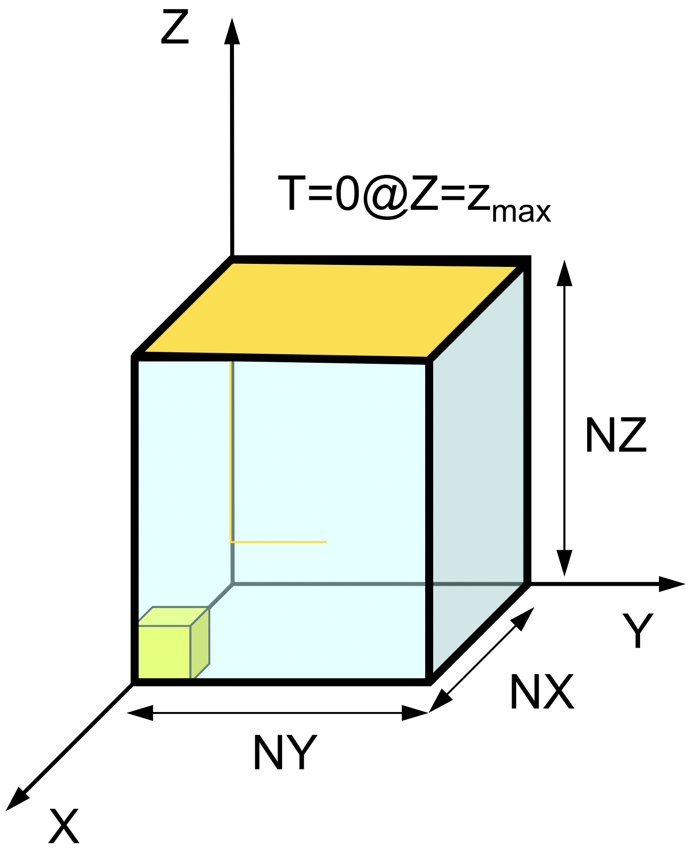
3D-FEM in Fortran
Steady State Heat Conduction

Kengo Nakajima
Information Technology Center

Technical & Scientific Computing II (4820-1028)
Seminar on Computer Science II (4810-1205)

3D Steady-State Heat Conduction

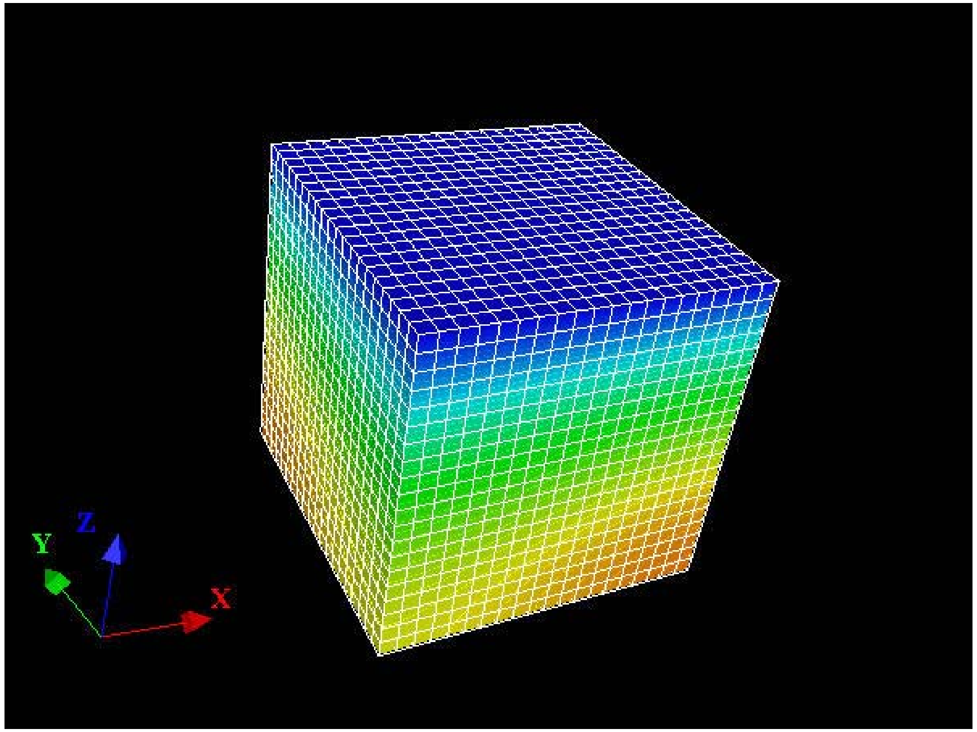
$$\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(\lambda\frac{\partial T}{\partial z}\right)+\dot{Q}(x,y,z)=0$$



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

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)

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

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

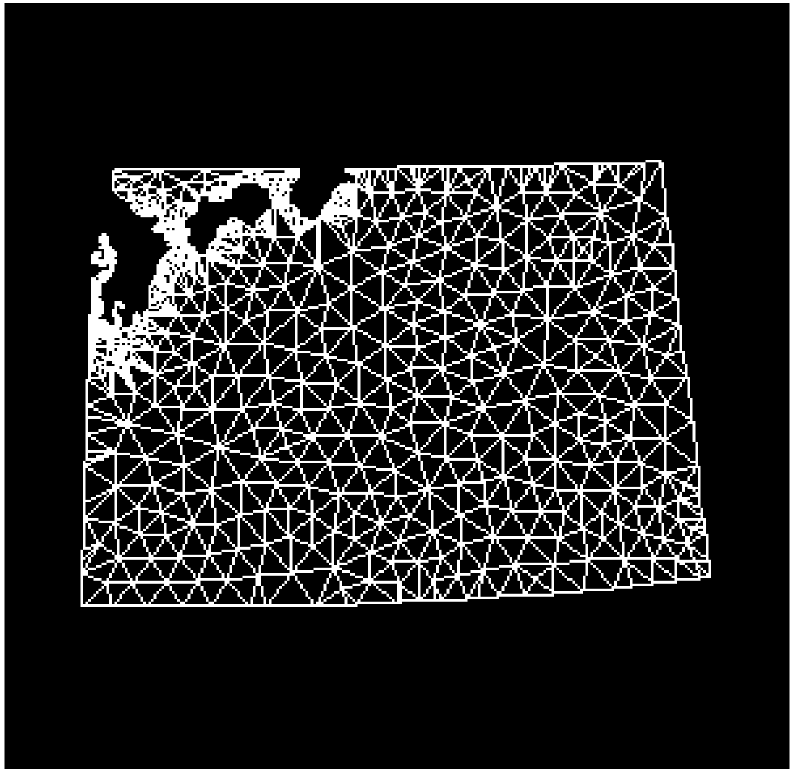
- Initialization
  - Control Data
  - Node, Connectivity of Elements (N: Node#, NE: Elem#)
  - Initialization of Arrays (Global/Element Matrices)
  - Element-Global Matrix Mapping (Index, Item)
- Generation of Matrix
  - Element-by-Element Operations (do icel= 1, NE)
    - Element matrices
    - Accumulation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

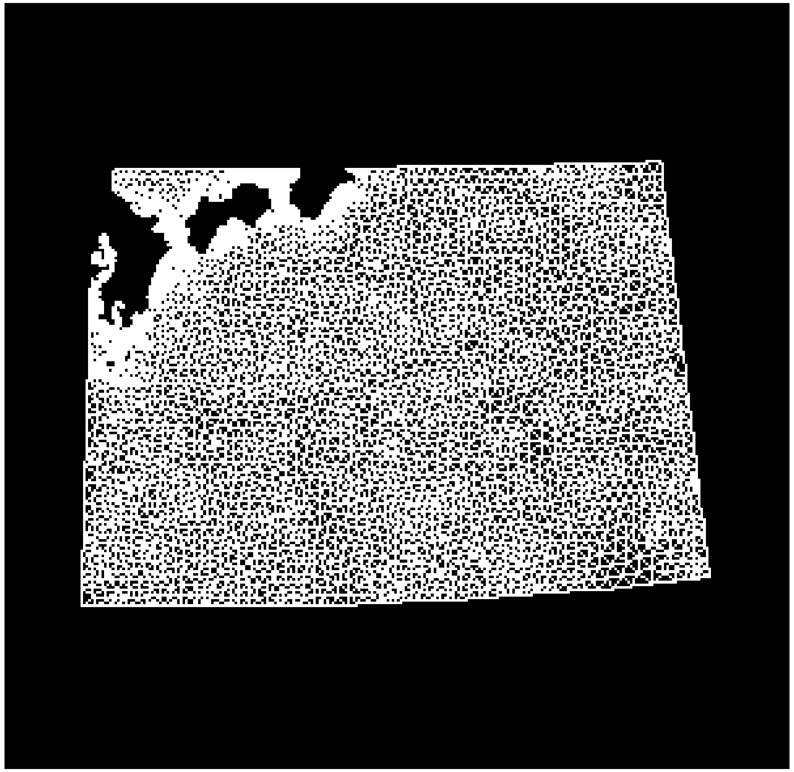
- Formulation of 3D Element
- 3D Heat Equations
  - Galerkin Method
  - Element Matrices

- Running the Code
- Data Structure
- Overview of the Program

Extension to 2D Prob.: Triangles
三角形要素

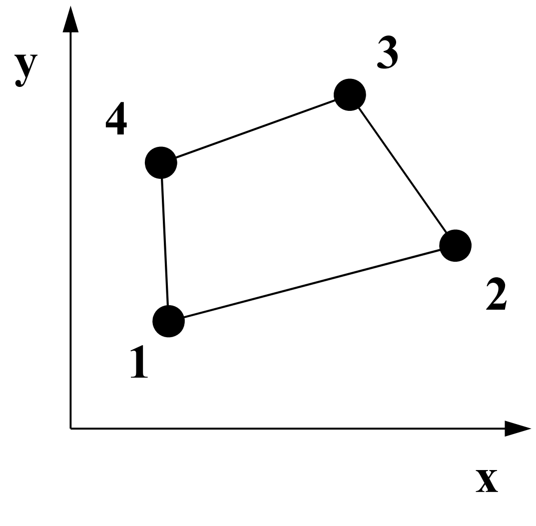
- Triangles can handle arbitrarily shaped object
- “Linear” triangular elements provide low accuracy, therefore they are not used in practical applications.





Extension to 2D Prob.: Quadrilaterals 四角形要素

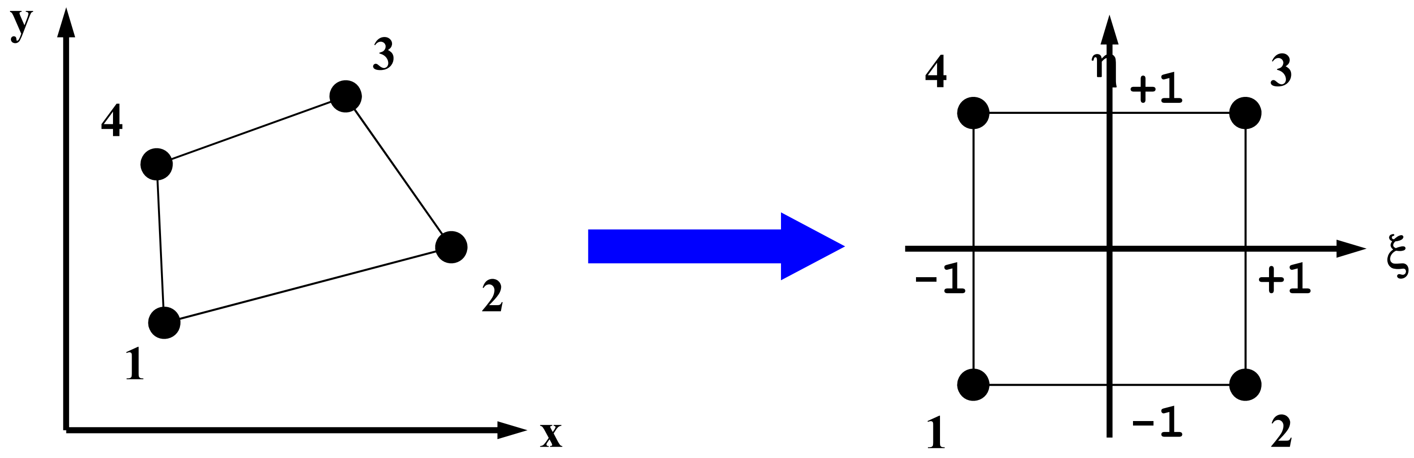
- Formulation of quad. elements is possible if same shape functions in 1D elements are applied along X- and Y- axis.
  - More accurate than triangles
- Each edge must be “parallel” with X- and Y- axis.
  - Similar to FDM



- This type of elements cannot be considered.

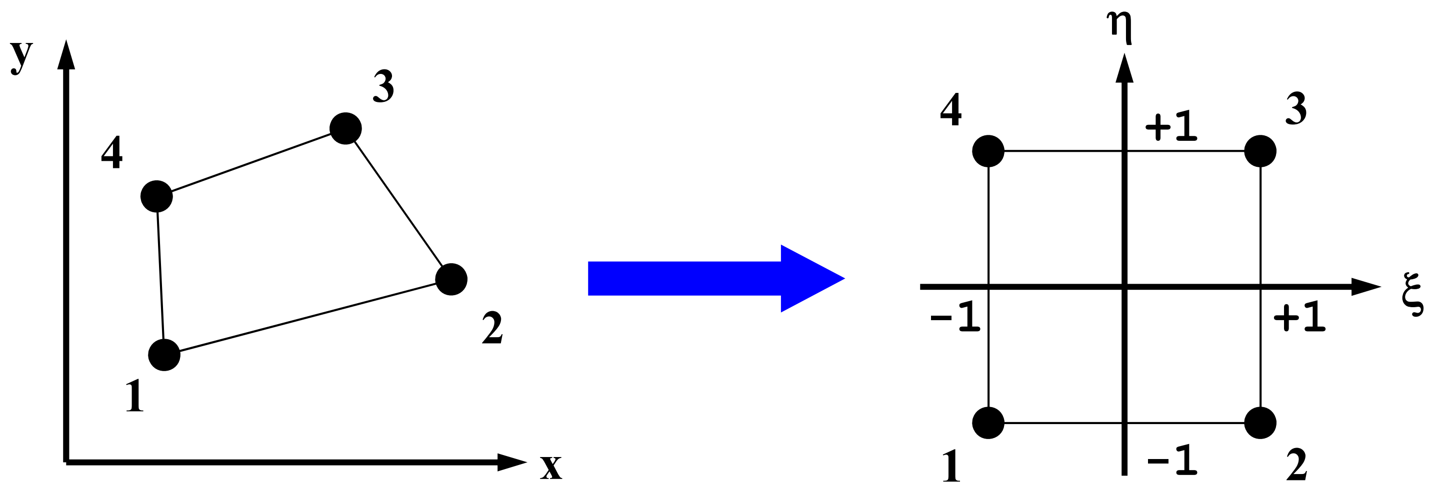
Isoparametric Element (1/3)

- Each element is mapped to square element [±1,±1] on natural/local coordinate (ξ,η)



- 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)

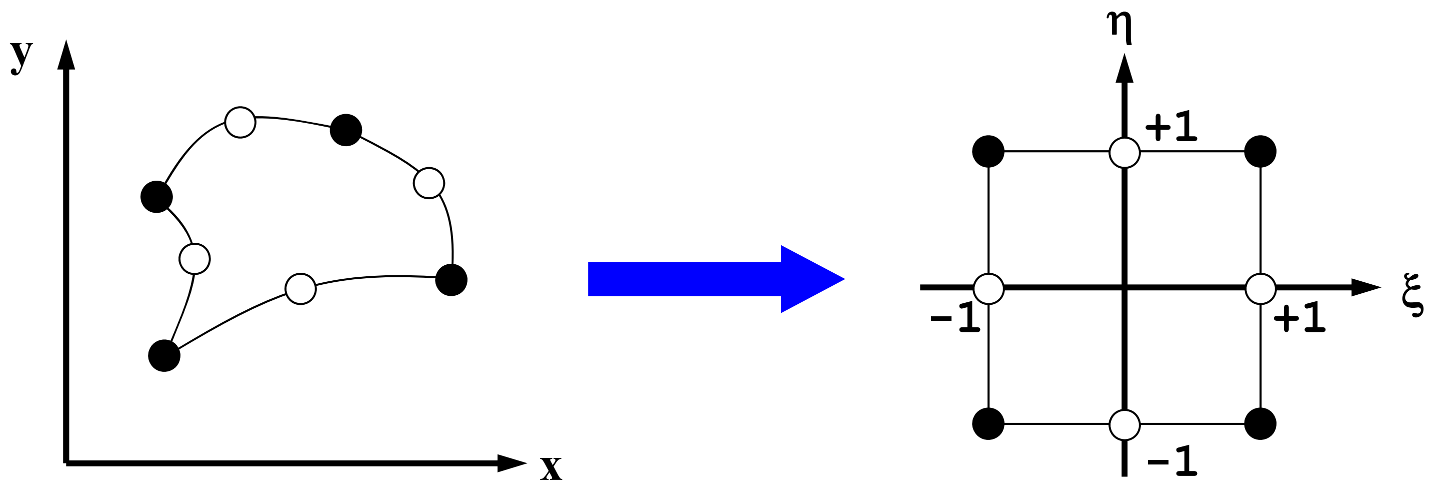


- Coordinate of each node: (x_1,y_1), (x_2,y_2), (x_3,y_3), (x_4,y_4)
- Temperature at each node: T_1, T_2, T_3, T_4

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

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

Isoparametric Element (3/3)



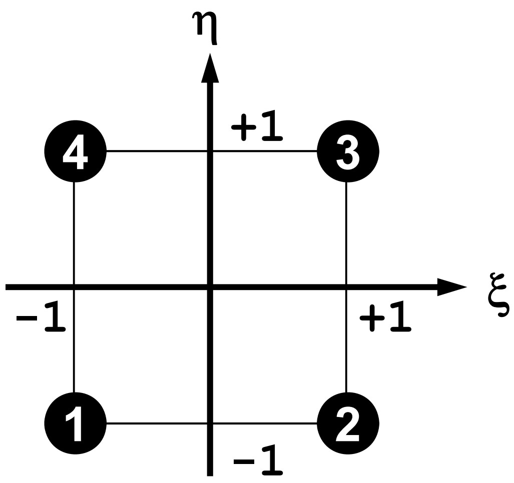
- Higher-order shape function can handle curved lines/surfaces.
- “Natural” coordinate system

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)

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

$$= \frac{T_1 + T_2 + T_3 + T_4}{4} + \frac{-T_1 + T_2 + T_3 - T_4}{4}\xi +$$

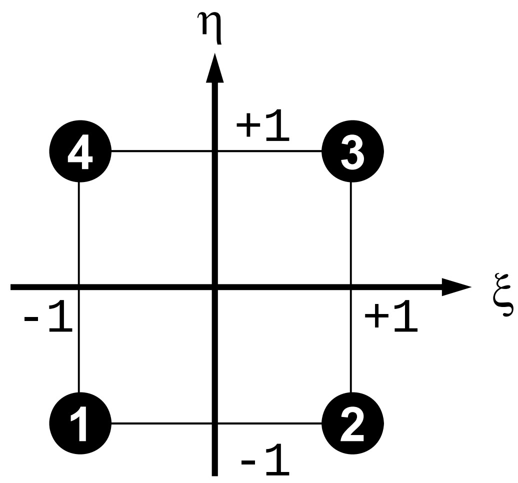
$$\frac{-T_1 - T_2 + T_3 + T_4}{4}\eta + \frac{T_1 - T_2 + T_3 - T_4}{4}\xi\eta$$

$$= \frac{1}{4}(1 - \xi - \eta + \xi\eta)T_1 + \frac{1}{4}(1 + \xi - \eta - \xi\eta)T_2 +$$

$$\frac{1}{4}(1 + \xi + \eta + \xi\eta)T_3 + \frac{1}{4}(1 - \xi + \eta - \xi\eta)T_4$$

$$= \frac{1}{4}(1 - \xi)(1 - \eta)T_1 + \frac{1}{4}(1 + \xi)(1 - \eta)T_2 +$$

$$\frac{1}{4}(1 + \xi)(1 + \eta)T_3 + \frac{1}{4}(1 - \xi)(1 + \eta)T_4$$



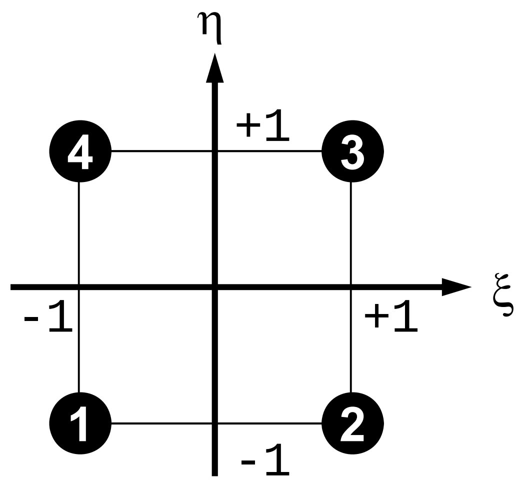
Shape Fn's on 2D Natural Coord. (2/3)

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

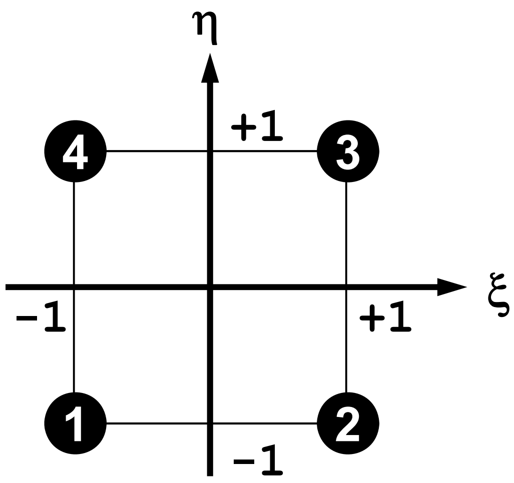
$$= \frac{T_1 + T_2 + T_3 + T_4}{4} + \frac{-T_1 + T_2 + T_3 - T_4}{4}\xi + \frac{-T_1 - T_2 + T_3 + T_4}{4}\eta + \frac{T_1 - T_2 + T_3 - T_4}{4}\xi\eta$$

$$= \frac{1}{4}(1 - \xi - \eta + \xi\eta)T_1 + \frac{1}{4}(1 + \xi - \eta - \xi\eta)T_2 + \frac{1}{4}(1 + \xi + \eta + \xi\eta)T_3 + \frac{1}{4}(1 - \xi + \eta - \xi\eta)T_4$$

$$= \underbrace{\frac{1}{4}(1-\xi)(1-\eta)}_{N_1}T_1 + \underbrace{\frac{1}{4}(1+\xi)(1-\eta)}_{N_2}T_2 + \underbrace{\frac{1}{4}(1+\xi)(1+\eta)}_{N_3}T_3 + \underbrace{\frac{1}{4}(1-\xi)(1+\eta)}_{N_4}T_4$$



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.

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

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

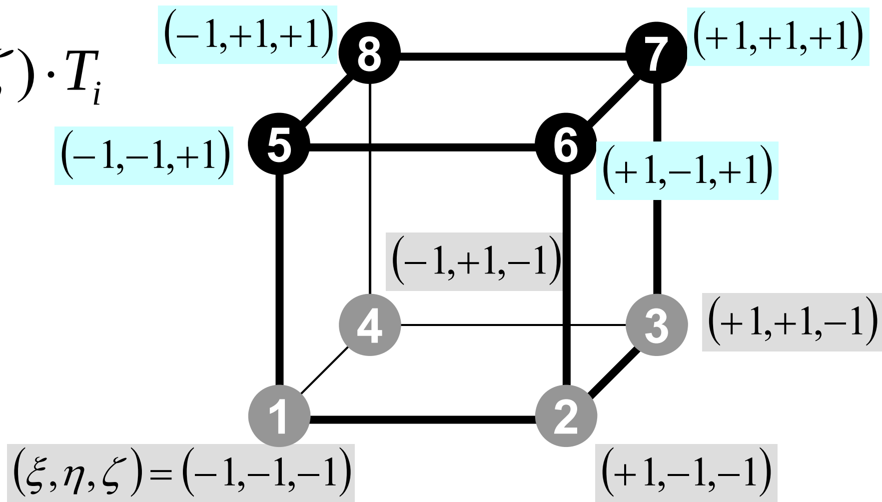
$$N_3(\xi,\eta,\zeta)=\frac{1}{8}(1+\xi)(1+\eta)(1-\zeta) \qquad N_7(\xi,\eta,\zeta)=\frac{1}{8}(1+\xi)(1+\eta)(1+\zeta)$$

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

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

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

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



- 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 λ):

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

$$T=[N]\{\phi\}$$

Distribution of temperature in each element (matrix form), ϕ: 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^{nd}-order diff. (surface integration terms are ignored):

$$\int_V [N]^T\left\{\lambda\left(T_{,xx}\right)+\lambda\left(T_{,yy}\right)+\lambda\left(T_{,zz}\right)\right\}dV$$

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

- 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 $\dot{Q}$:

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

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

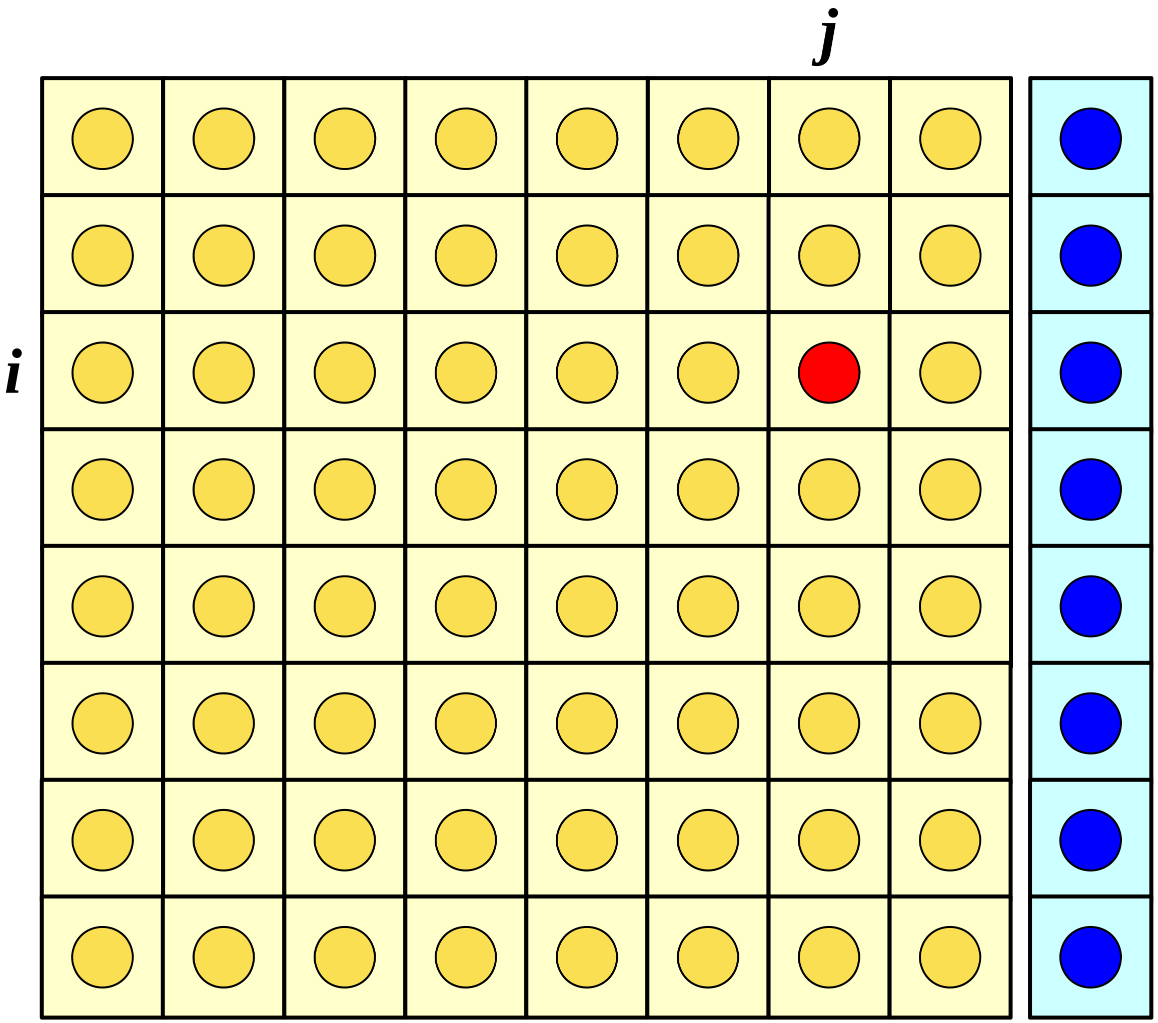
Weak Form with B.C.: on each elem.

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

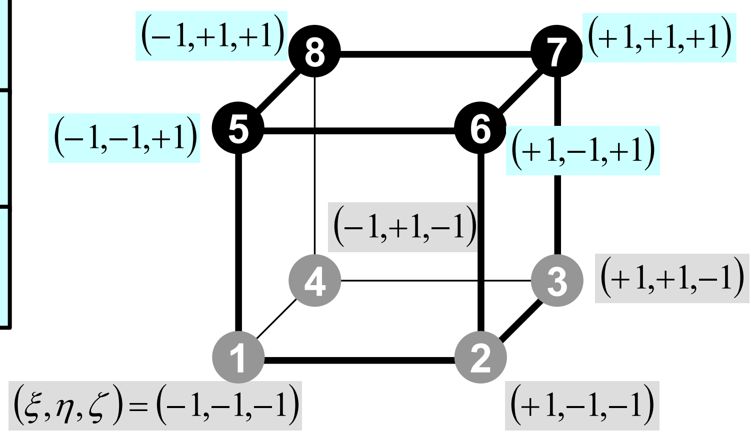
$$\left[k\right]^{(e)}=\int_V \lambda\left(\left[N_{,x}\right]^T\left[N_{,x}\right]\right)dV+\int_V \lambda\left(\left[N_{,y}\right]^T\left[N_{,y}\right]\right)dV$$
$$+\int_V \lambda\left(\left[N_{,z}\right]^T\left[N_{,z}\right]\right)dV$$

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

Element Matrix: 8x8

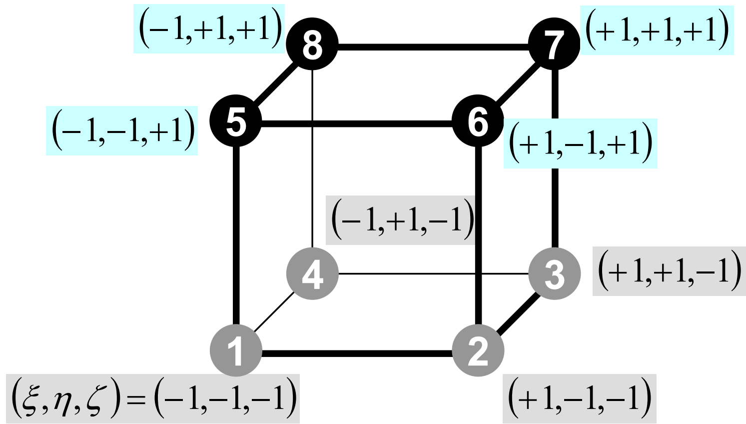


$$\left[k_{ij}\right] \quad \left(i, j = 1 \ldots 8\right)$$



Element Matrix: k_{ij}

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



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

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

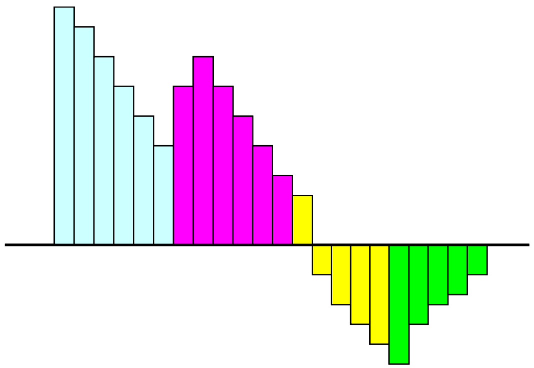
Next Stage: Integration

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:

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



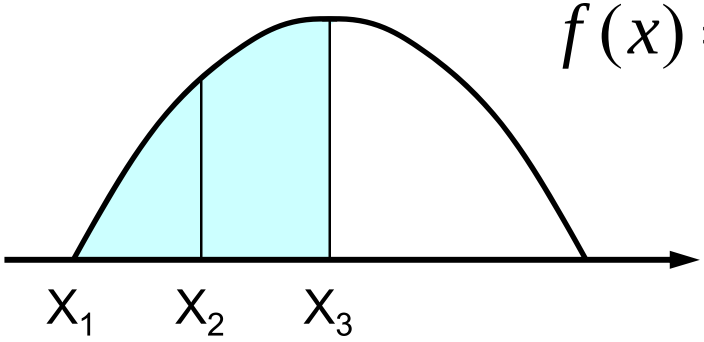
Gaussian Quadrature in 1D

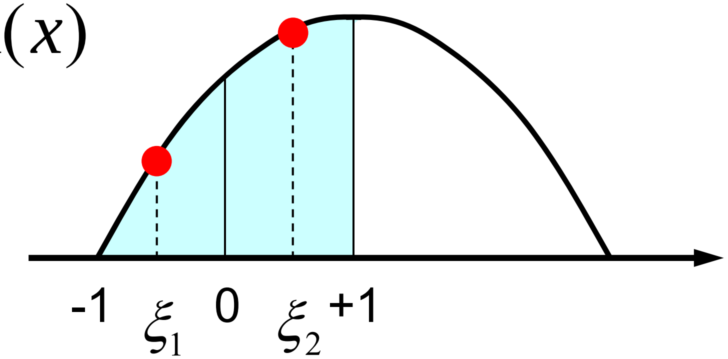
more accurate than Simpson's rule

Simpson's

Gauss

$$f(x) = \sin(x)$$





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

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

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

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

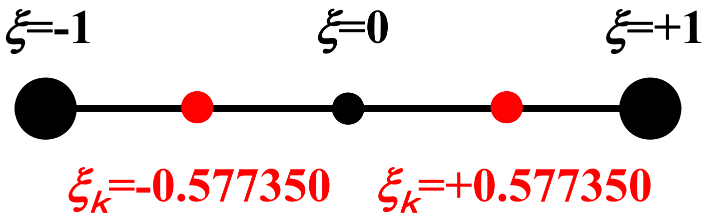
$$S = \int_0^{\pi/2} f(x)dx = \int_{-1}^{+1} f(\xi)\, h\, d\xi$$

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

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).

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



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

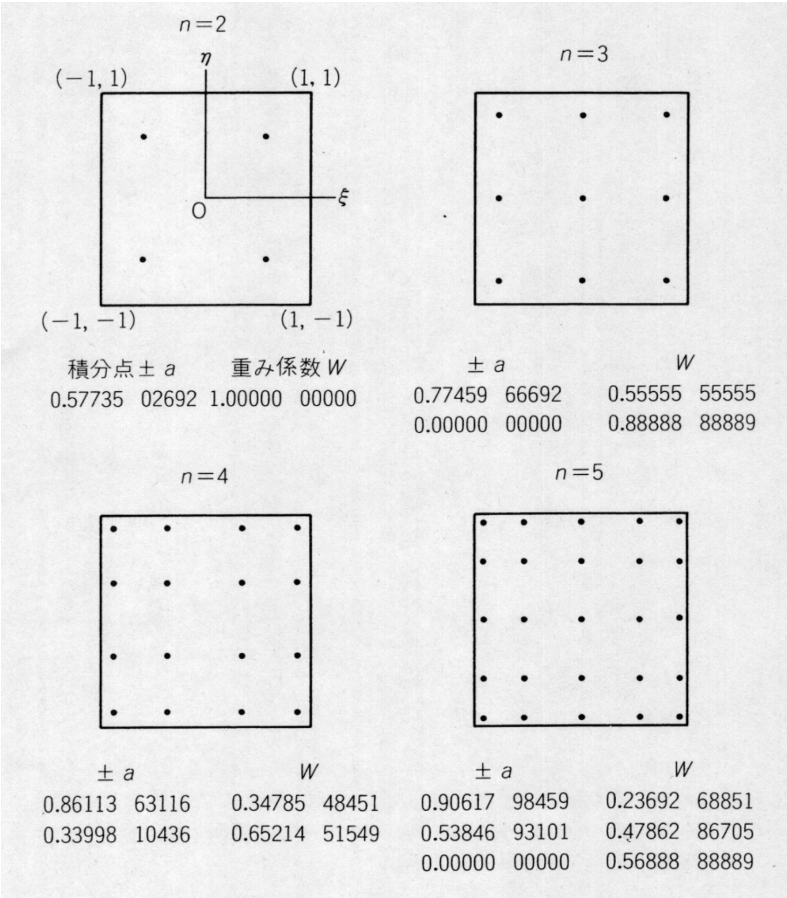
$$m=2 \quad \xi_k = \pm 0.577350,\ w_k = 1.00$$

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

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

Gaussian Quadrature

can be easily extended to 2D & 3D



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

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

m, n: number of quadrature points in ξ, η-direction

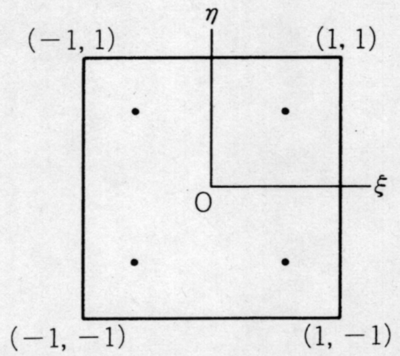
(ξ_i, η_j): Coordinates of Quad's

W_i, W_j: Weighting Factor

Gaussian Quadrature
ガウスの積分公式

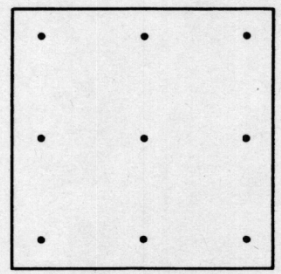
This configuration is widely used. In 2D problem, integration is done using values of “f” at 4 quad. points.

$n=2$



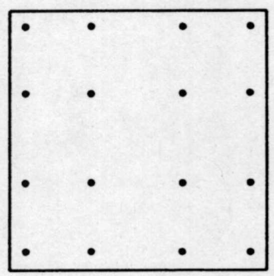
積分点 $\pm a$	重み係数 W
0.57735 02692	1.00000 00000

$n=3$



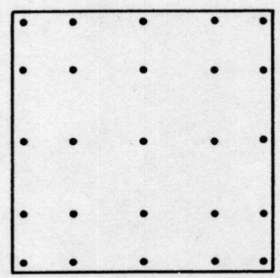
$\pm a$	W
0.77459 66692	0.55555 55555
0.00000 00000	0.88888 88889

$n=4$



$\pm a$	W
0.86113 63116	0.34785 48451
0.33998 10436	0.65214 51549

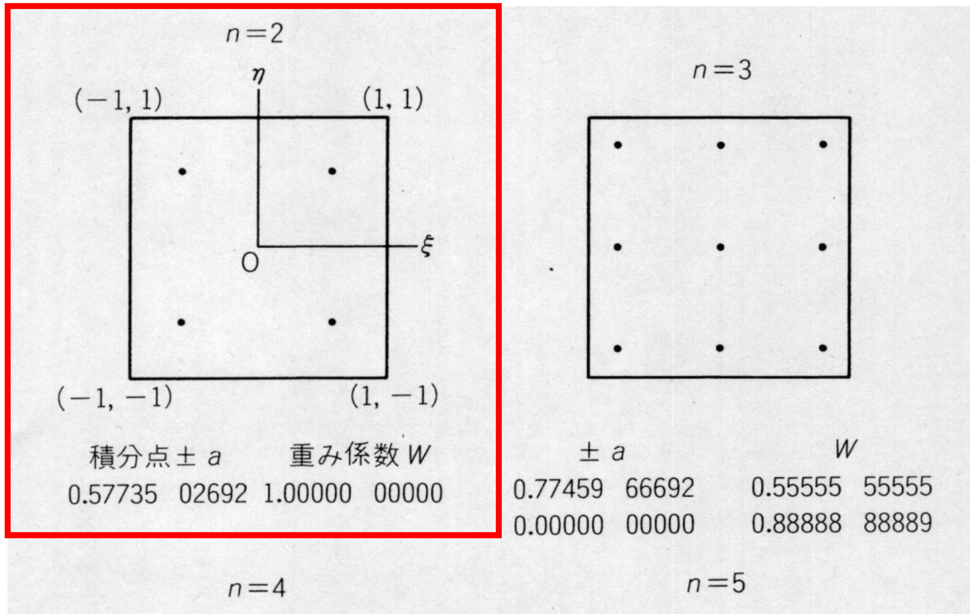
$n=5$



$\pm a$	W
0.90617 98459	0.23692 68851
0.53846 93101	0.47862 86705
0.00000 00000	0.56888 88889

Gaussian Quadrature
ガウスの積分公式

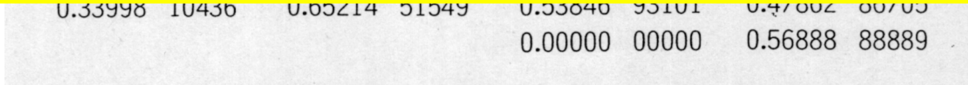
This configuration is widely used. In 2D problem, integration is done using values of “f” at 4 quad. points.



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

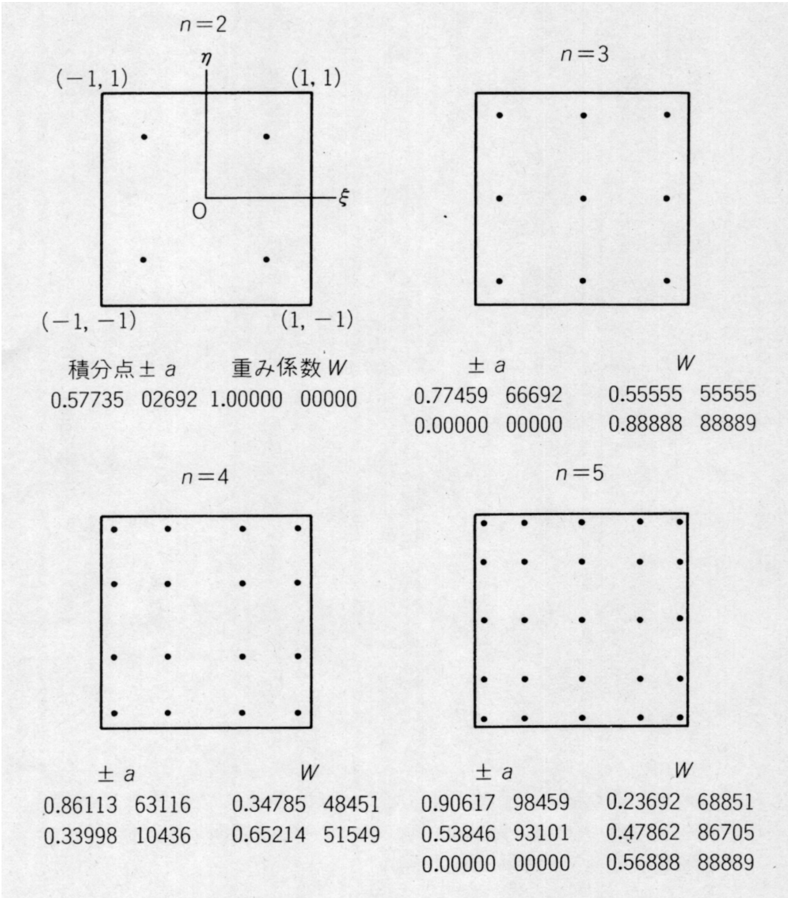
$$=1.0\times1.0\times f(-0.57735,-0.57735)+1.0\times1.0\times f(-0.57735,+0.57735)$$

$$+1.0\times1.0\times f(+0.57735,+0.57735)+1.0\times1.0\times f(+0.57735,-0.57735)$$



Next Stage: Integration

- 3D Natural/Local Coordinate (ξ,η,ζ):
  - Gaussian Quadrature



n=2

積分点 ± a	重み係数 W
0.57735 02692	1.00000 00000

n=3

± a	W
0.77459 66692	0.55555 55555
0.00000 00000	0.88888 88889

n=4

± a	W
0.86113 63116	0.34785 48451
0.33998 10436	0.65214 51549

n=5

± a	W
0.90617 98459	0.23692 68851
0.53846 93101	0.47862 86705
0.00000 00000	0.56888 88889

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

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

L,M,N: number of quadrature points in ξ,η,ζ-direction

(ξ_i,η_j,ζ_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:

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

$[J]$: Jacobi matrix, Jacobian

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

- Components of Jacobian:

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

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

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

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

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

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

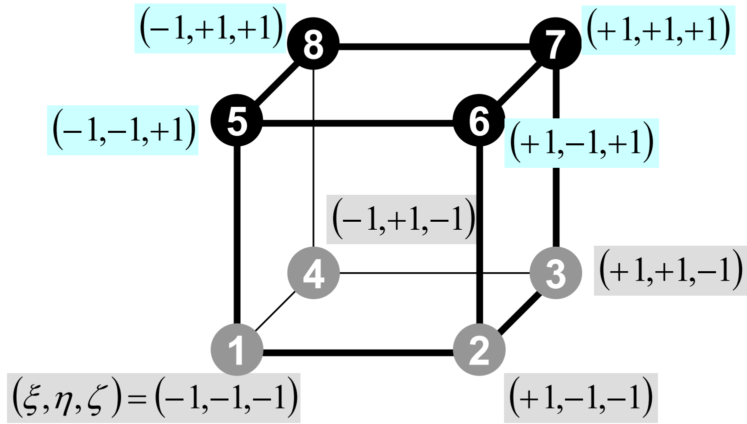
Partial Diff. on Natural Coord. (4/4)

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

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

Integration on Element

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

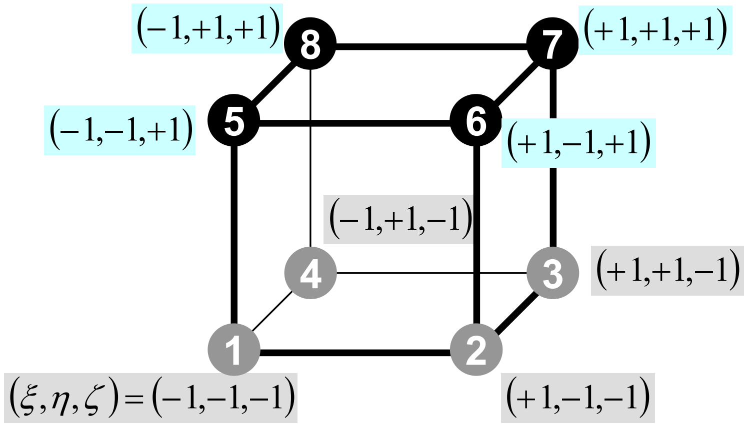


$$k_{ij} = -\int_V \left\{ \lambda \cdot N_{i,x} \cdot N_{j,x} + \lambda \cdot N_{i,y} \cdot N_{j,y} + \lambda \cdot N_{i,z} \cdot N_{j,z} \right\} dV$$

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

Integration on Natural Coord.

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



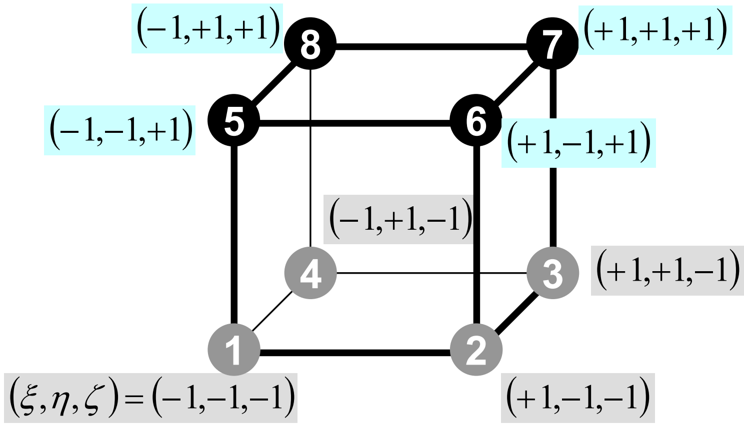
$$-\int_V \left\{ \lambda \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \lambda \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + \lambda \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right\} dV =$$

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

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

Gaussian Quadrature

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



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

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

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

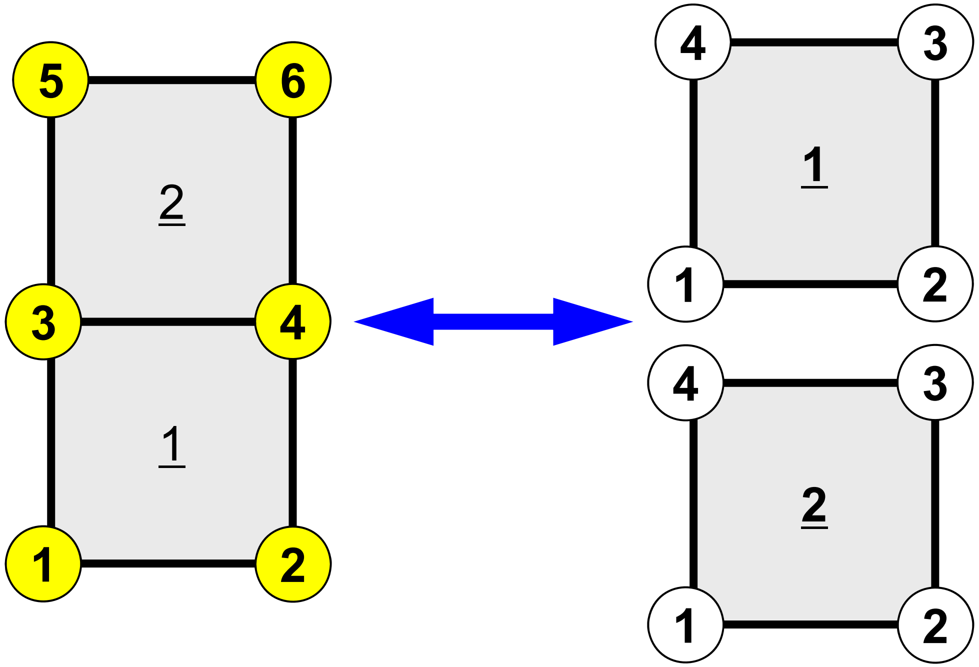
Remaining Procedures

- 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



$$[k^{(2)}]\{\phi^{(2)}\}=\{f^{(2)}\}$$

$$\begin{bmatrix} k_{11}^{(2)} & k_{12}^{(2)} & k_{13}^{(2)} & k_{14}^{(2)} \\ k_{21}^{(2)} & k_{22}^{(2)} & k_{23}^{(2)} & k_{24}^{(2)} \\ k_{31}^{(2)} & k_{32}^{(2)} & k_{33}^{(2)} & k_{34}^{(2)} \\ k_{41}^{(2)} & k_{42}^{(2)} & k_{43}^{(2)} & k_{44}^{(2)} \end{bmatrix} \begin{Bmatrix} \phi_1^{(2)} \\ \phi_2^{(2)} \\ \phi_3^{(2)} \\ \phi_4^{(2)} \end{Bmatrix} = \begin{Bmatrix} f_1^{(2)} \\ f_2^{(2)} \\ f_3^{(2)} \\ f_4^{(2)} \end{Bmatrix}$$

$$[k^{(1)}]\{\phi^{(1)}\}=\{f^{(1)}\}$$

$$\begin{bmatrix} k_{11}^{(1)} & k_{12}^{(1)} & k_{13}^{(1)} & k_{14}^{(1)} \\ k_{21}^{(1)} & k_{22}^{(1)} & k_{23}^{(1)} & k_{24}^{(1)} \\ k_{31}^{(1)} & k_{32}^{(1)} & k_{33}^{(1)} & k_{34}^{(1)} \\ k_{41}^{(1)} & k_{42}^{(1)} & k_{43}^{(1)} & k_{44}^{(1)} \end{bmatrix} \begin{Bmatrix} \phi_1^{(1)} \\ \phi_2^{(1)} \\ \phi_3^{(1)} \\ \phi_4^{(1)} \end{Bmatrix} = \begin{Bmatrix} f_1^{(1)} \\ f_2^{(1)} \\ f_3^{(1)} \\ f_4^{(1)} \end{Bmatrix}$$

$$[K]\{\Phi\}=\{F\}$$

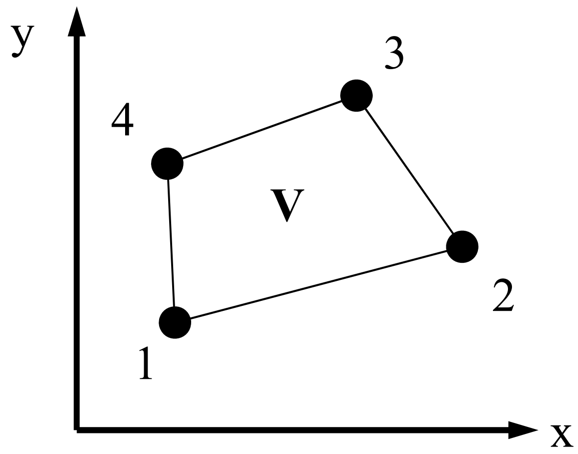
$$\begin{bmatrix} D_1 & X & X & X & & \\ X & D_2 & X & X & & \\ X & X & D_3 & X & X & X \\ X & X & X & D_4 & X & X \\ & & X & X & D_5 & X \\ & & X & X & X & D_6 \end{bmatrix} \begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \end{Bmatrix} = \begin{Bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \end{Bmatrix}$$

- 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)

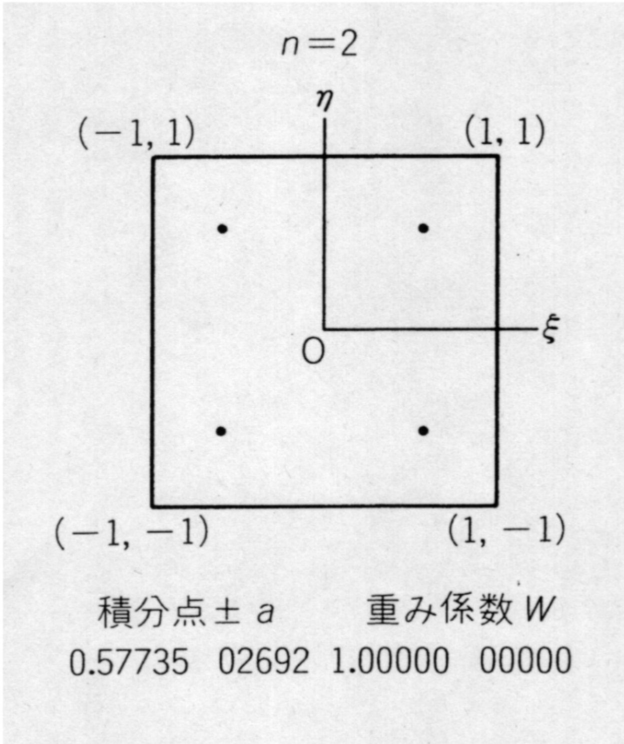
$$I = \int_V dV = \int_{-1}^{+1} \int_{-1}^{+1} \det\left|J\right| d\xi d\zeta$$

Tips (1/2)

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

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

```
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)

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

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

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

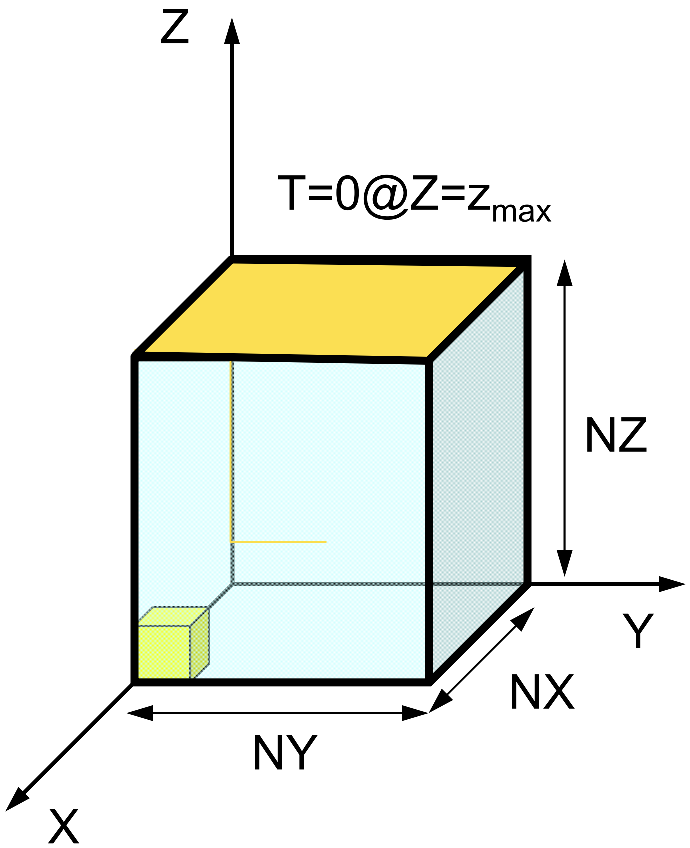
$$N_1(\xi,\eta)=\frac{1}{4}(1-\xi)(1-\eta),\quad N_2(\xi,\eta)=\frac{1}{4}(1+\xi)(1-\eta),$$

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

- 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 λ
- HEX meshes
  - 1x1x1 cubes
  - NX, NY, NZ cubes in each direction
- Boundary Conditions
  - $T=0@Z=z_{max}$
- 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

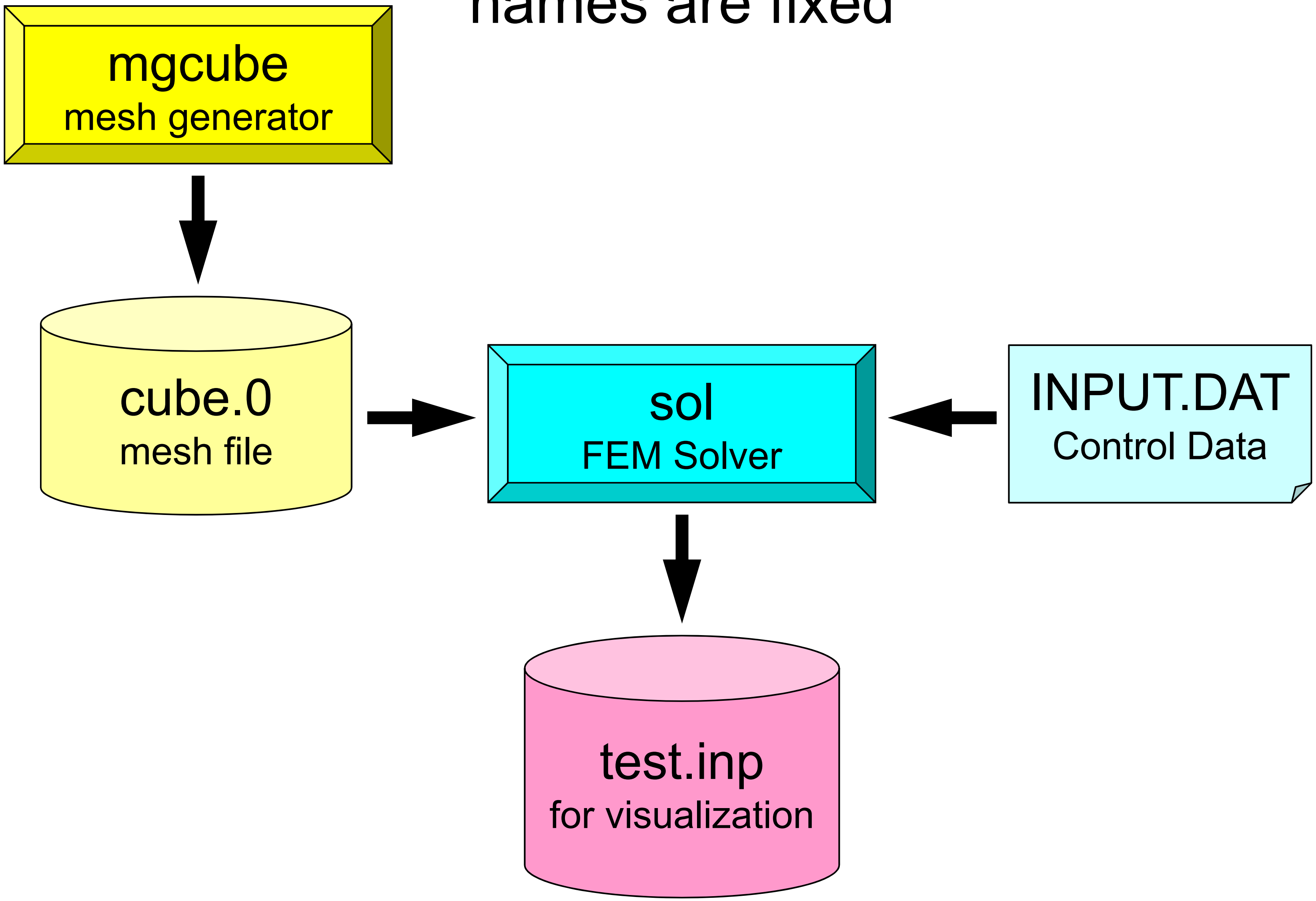
```
>$ cd <$E-TOP>/fem3d/src
>$ make
>$ ls ../run/sol
   sol
```

Install of Mesh Generator

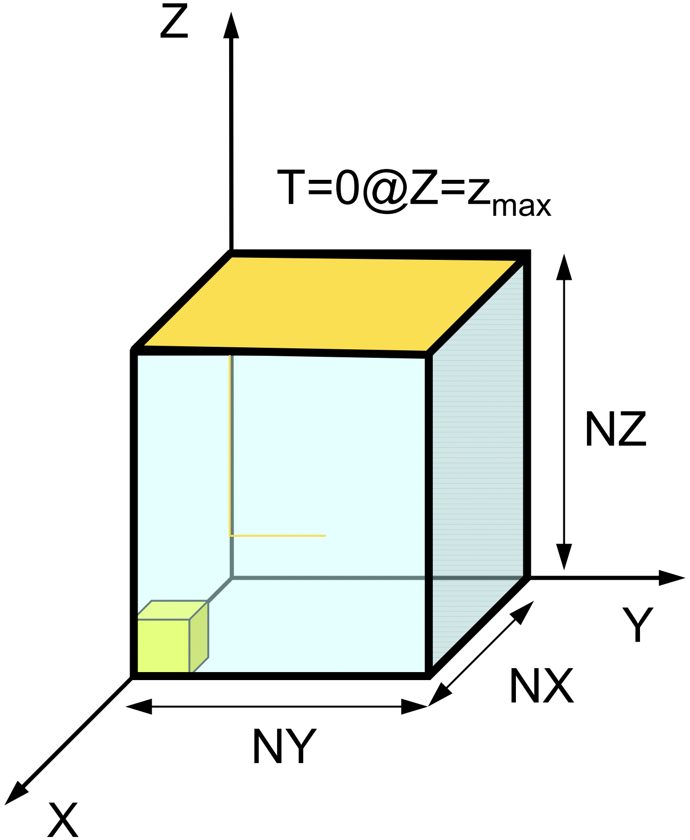
```
>$ cd <$E-TOP>/fem3d/run
>$ g95 -O3 mgcube.f -o mgcube
```

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

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

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

Running

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

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

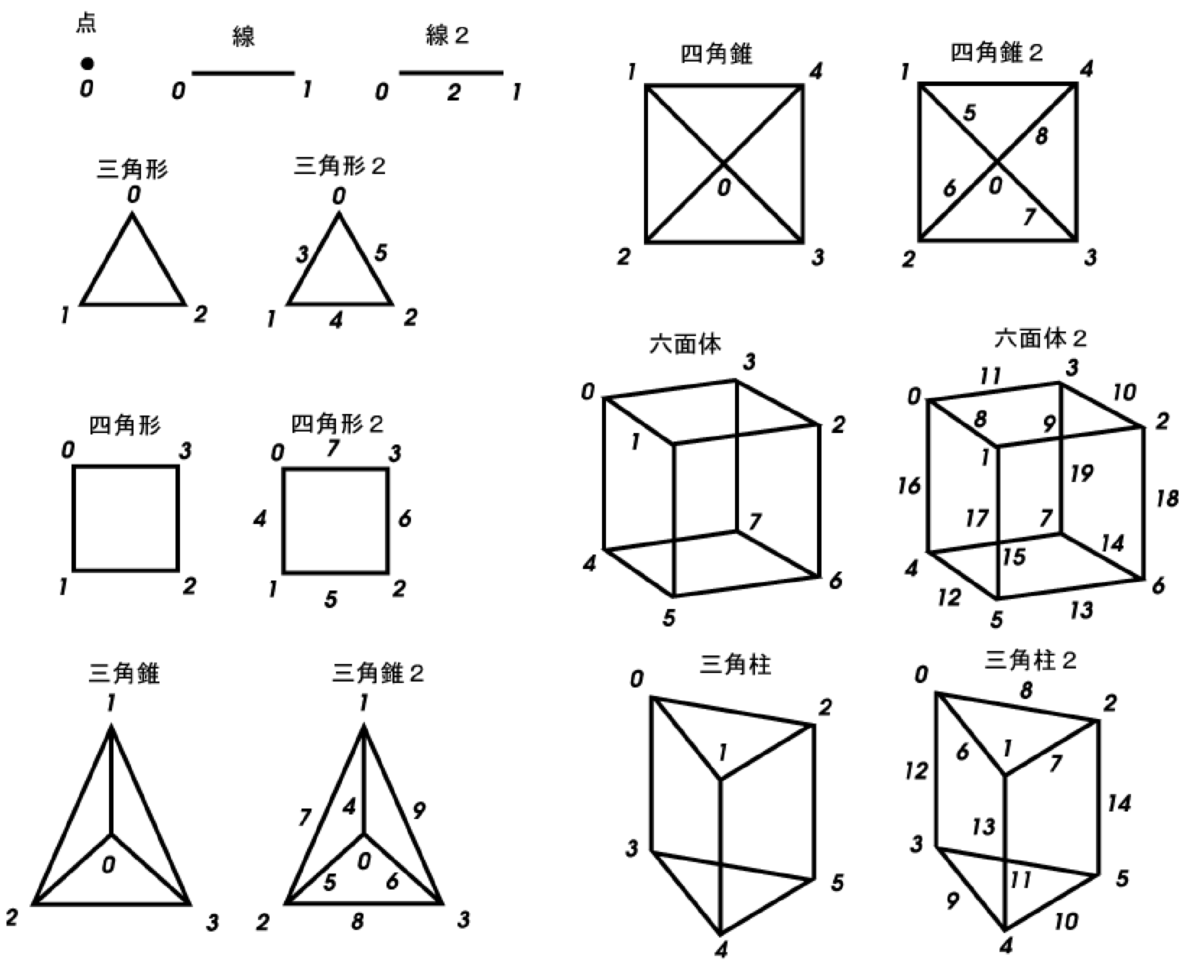
ParaView

- 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	キーワード
点	pt
線	line
三角形	tri
四角形	quad
四面体	tet
角錐	pyr
三角柱	prism
六面体	hex
二次要素	
線2	line2
三角形2	tri2
四角形2	quad2
四面体2	tet2
角錐2	pyr2
三角柱2	prism2
六面体2	hex2



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

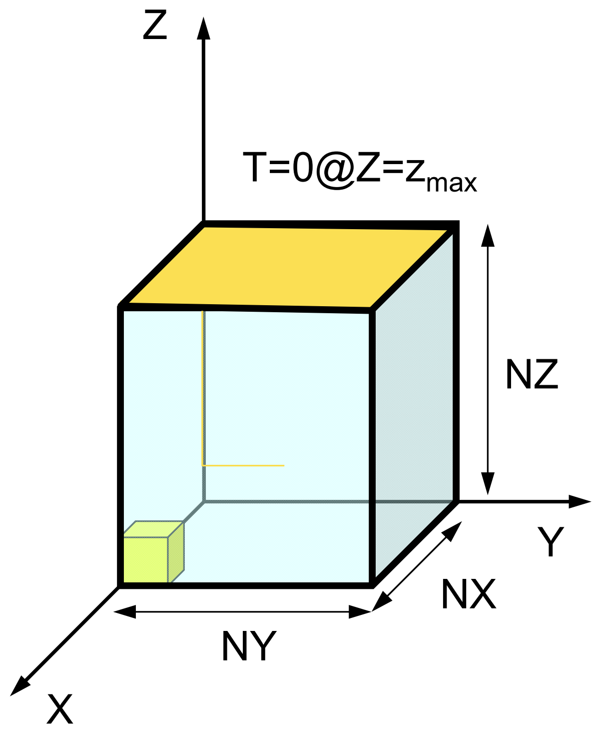
Overview of Mesh File: cube.0

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

Example of “cube.0” (NX=NY=NZ=4) Node

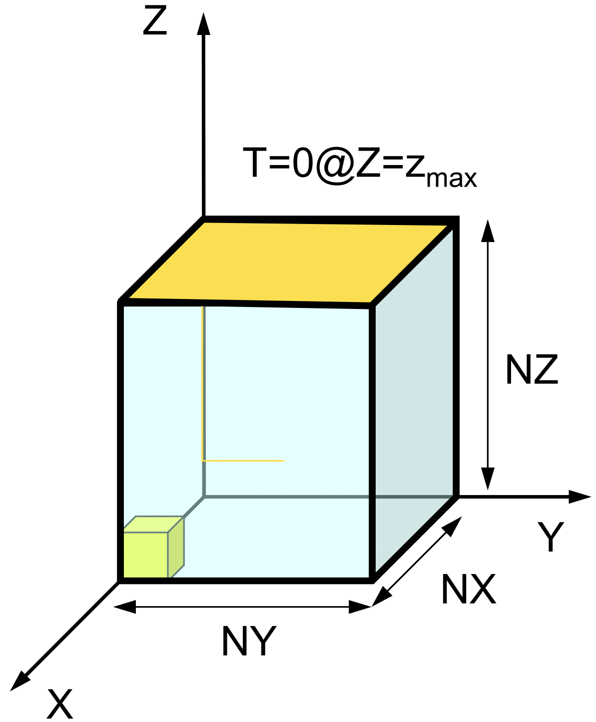
```
  125                               =5*5*5 (Node #)
    1      0.00     0.00     0.00
    2      1.00     0.00     0.00
    3      2.00     0.00     0.00
    4      3.00     0.00     0.00
    5      4.00     0.00     0.00
    6      0.00     1.00     0.00
    7      1.00     1.00     0.00
    8      2.00     1.00     0.00
    9      3.00     1.00     0.00

...

  121      0.00     4.00     4.00
  122      1.00     4.00     4.00
  123      2.00     4.00     4.00
  124      3.00     4.00     4.00
  125      4.00     4.00     4.00

Node ID X-coord.   Y        Z
```



movie

Example of "cube.0" (NX=NY=NZ=4) Element (1/2)

```
  64                               =4*4*4 (Element #)
 361  361  361  361  361  361  361  361  361  361
 361  361  361  361  361  361  361  361  361  361
 361  361  361  361  361  361  361  361  361  361
 361  361  361  361  361  361  361  361  361  361
 361  361  361  361  361  361  361  361  361  361
 361  361  361  361  361  361  361  361  361  361
 361  361  361  361
```

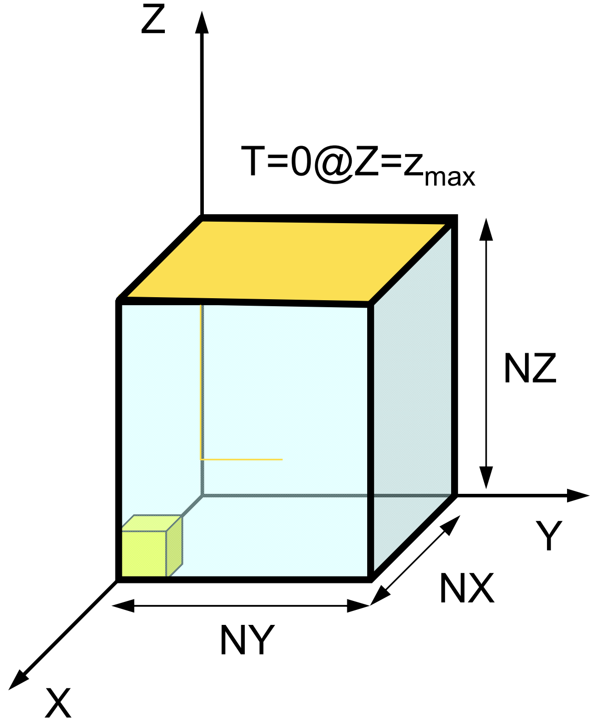


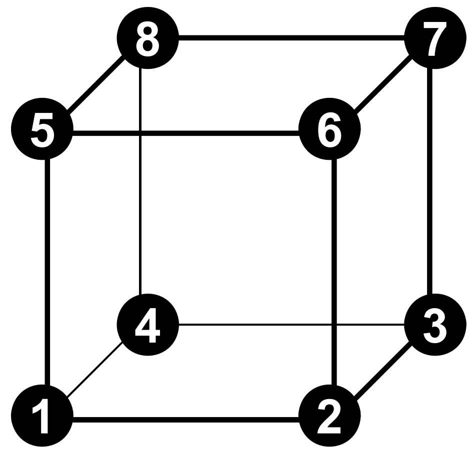
Element Type: 361

3D, Hexahedron, Linear (1st order)

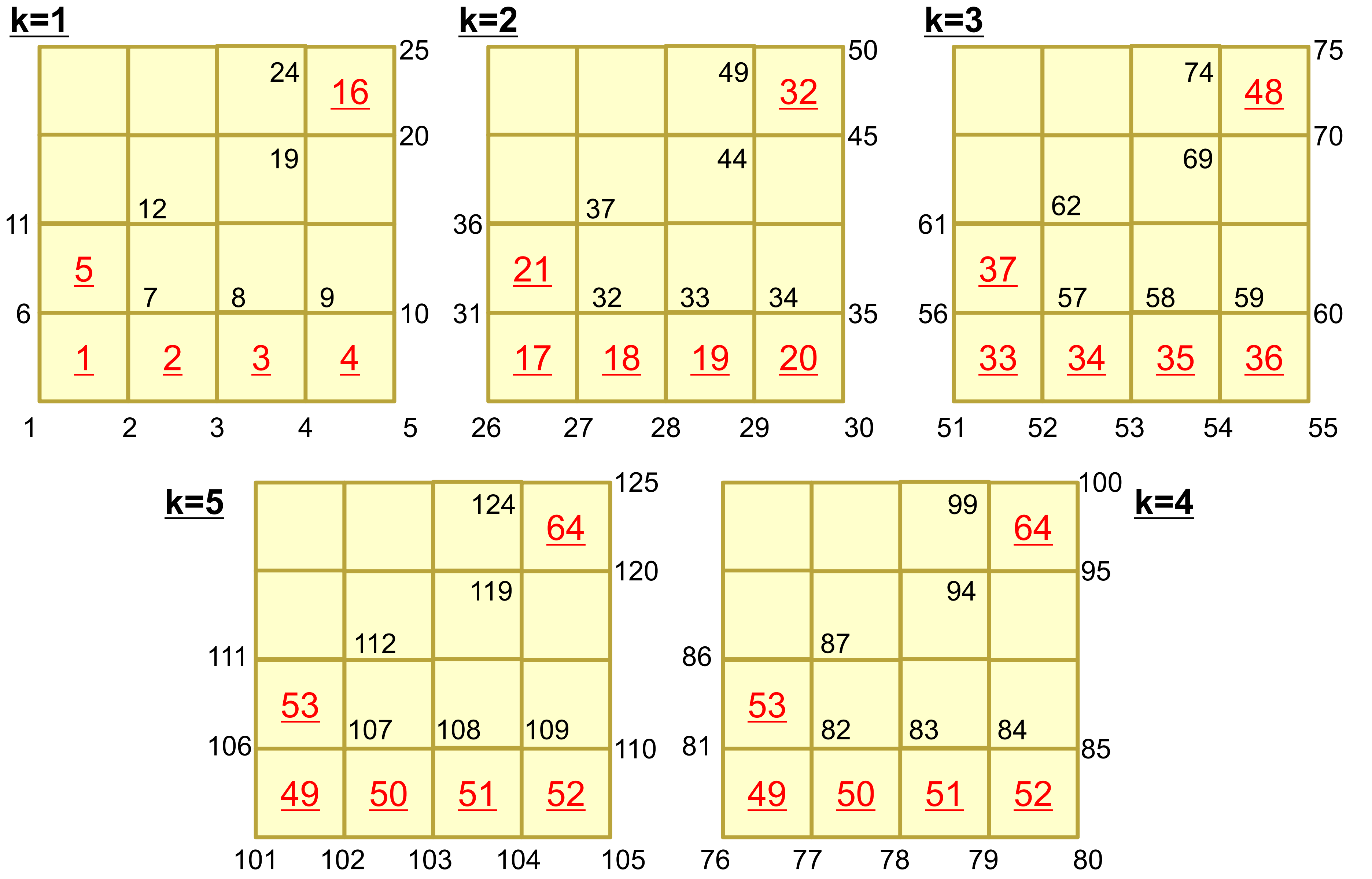
Example of “cube.0” Element (2/2)

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
...									
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
Elem ID	MAT-ID	ID of 8 nodes							



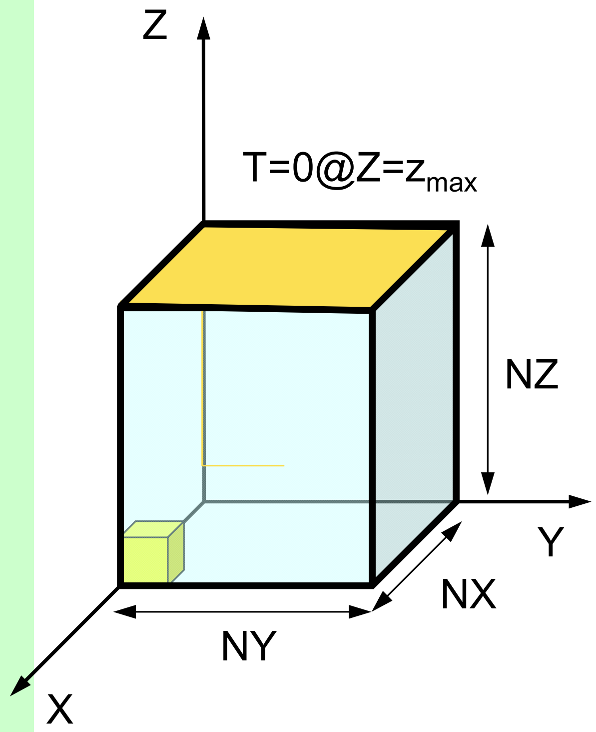


NX=NY=NZ=4, NXP1=NYP1=NZP1=5
ICELTOT= 64, INODTOT= 125, IBNODTOT= 25

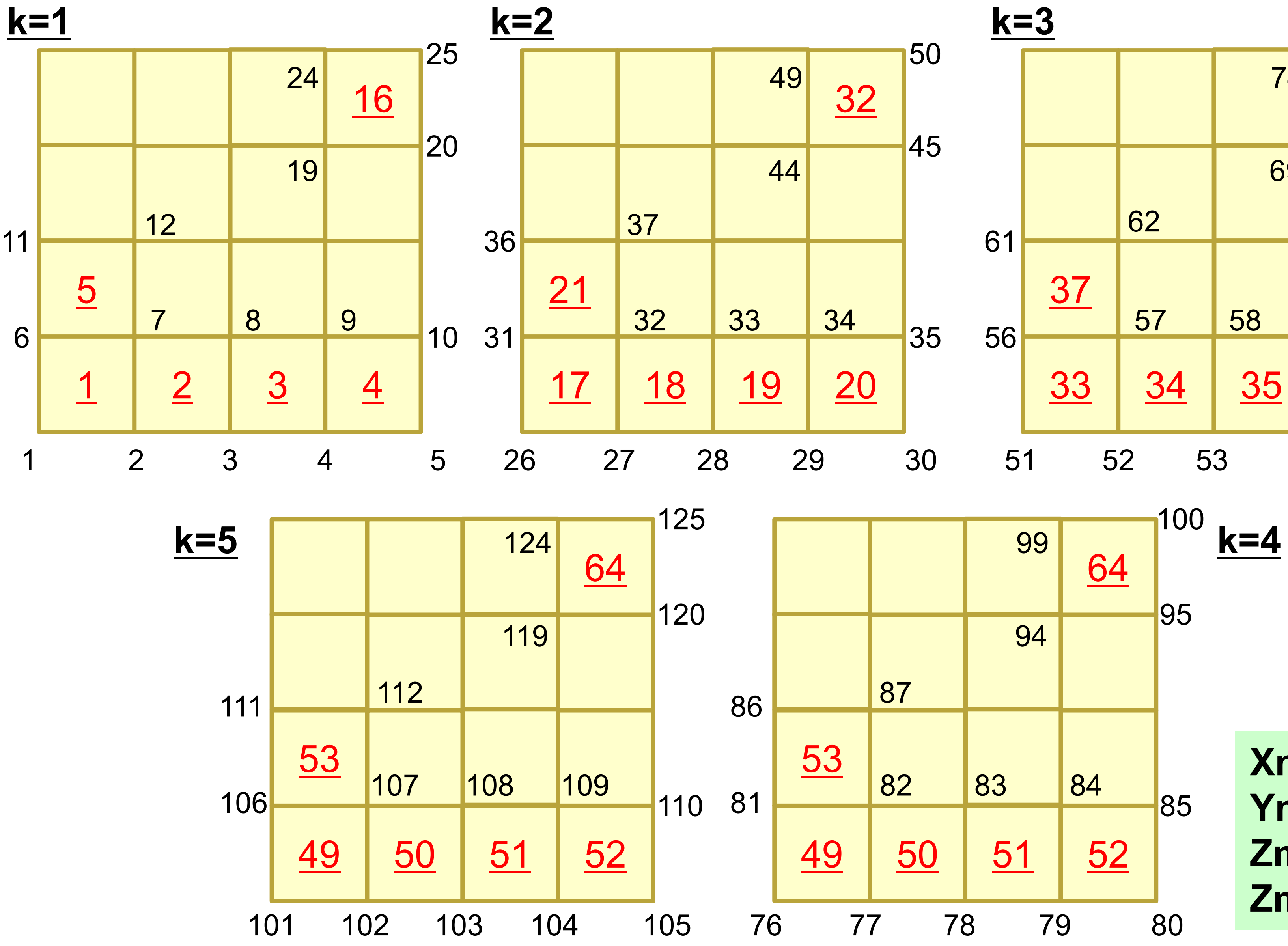


Example of “cube.0” Node Grp. Info.

```
   4                              Number of Groups
  25   50   75 100                Number of Nodes (ea. grp.)

Xmin
   1    6   11   16   21   26   31   36   41   46
  51   56   61   66   71   76   81   86   91   96
 101  106  111  116  121

Ymin
   1    2    3    4    5   26   27   28   29   30
  51   52   53   54   55   76   77   78   79   80
 101  102  103  104  105

Zmin
   1    2    3    4    5    6    7    8    9   10
  11   12   13   14   15   16   17   18   19   20
  21   22   23   24   25

Zmax
 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

no use after this line
```



NX=NY=NZ=4, NXP1=NYP1=NZP1=5
ICELTOT= 64, INODTOT= 125, IBNODTOT= 25



Xmin: i=1
Ymin: j=1
Zmin: k=1
Zmax:k=5

Mesh Generation

- Big Technical & Research Issue
  - Complicated Geometry
  - Large Scale
- Parallelization is difficult

- Commercial Mesh Generator
  - FEMAP
    - Interface to CAD Data Format

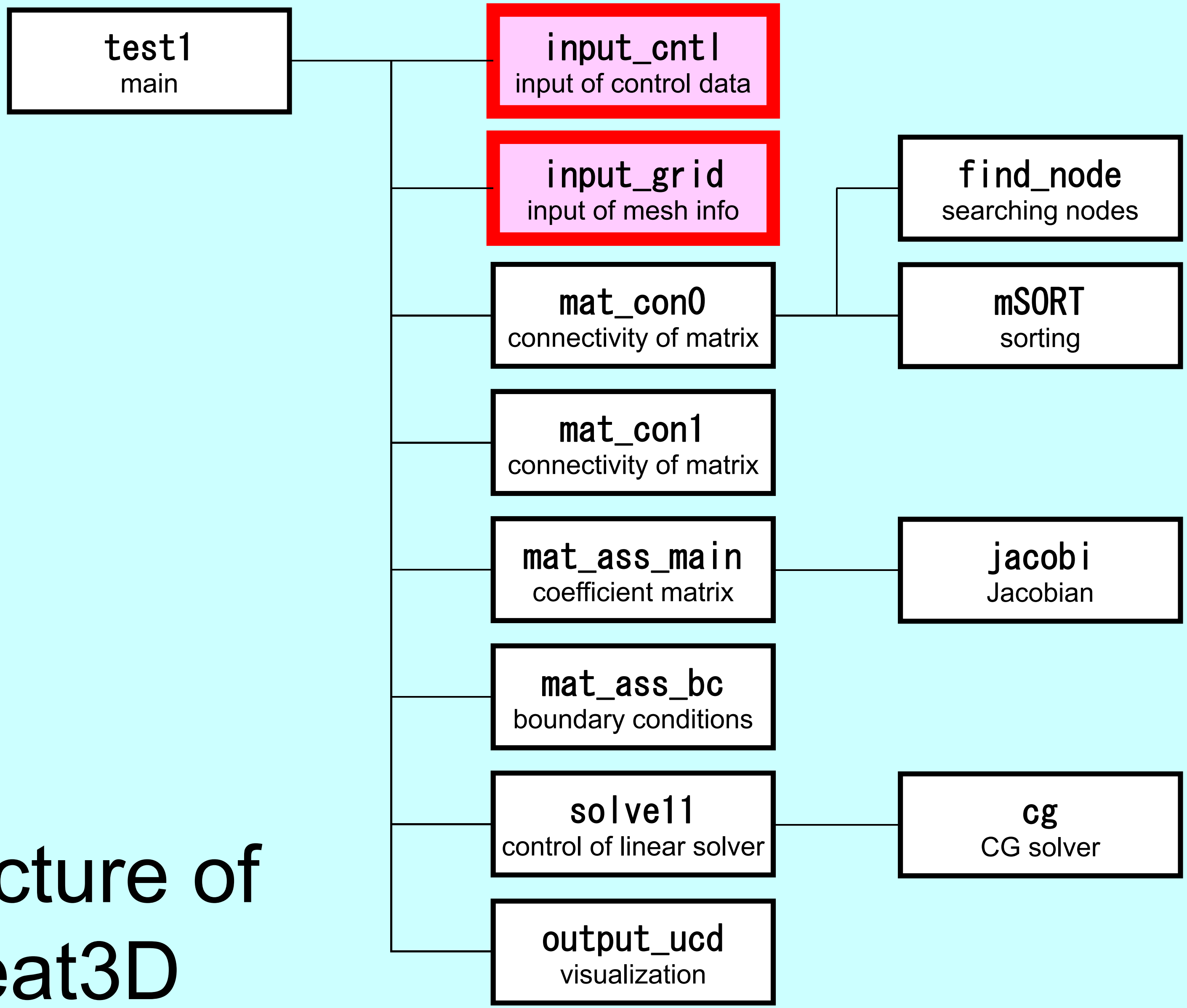
movie

- 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

Structure of heat3D



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
```

Global Variables: pfem_util.f (1/3)

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

Global Variables: pfem_util.h (2/3)

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

Global Variables: pfem_util.h (3/3)

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

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

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

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

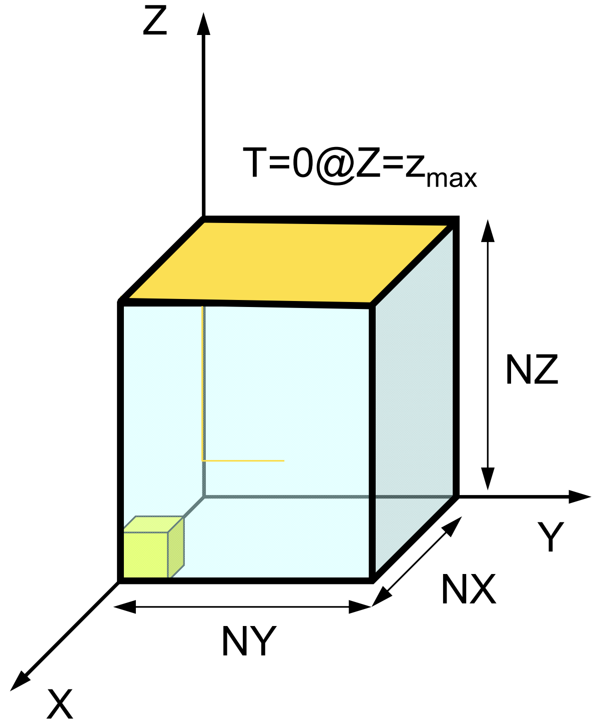
```
cube.0        fname
2000          ITER
1.0 1.0       COND, QVOL
1.0e-08       RESID
```

INPUT_GRID (1/3)

```
!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
```

Example of "cube.0" Node

```
 125                                     = N
   1       0.00      0.00      0.00
   2       1.00      0.00      0.00
   3       2.00      0.00      0.00
   4       3.00      0.00      0.00
   5       4.00      0.00      0.00
   6       0.00      1.00      0.00
   7       1.00      1.00      0.00
   8       2.00      1.00      0.00
   9       3.00      1.00      0.00

...

 121       0.00      4.00      4.00
 122       1.00      4.00      4.00
 123       2.00      4.00      4.00
 124       3.00      4.00      4.00
 125       4.00      4.00      4.00

                 XYZ(i,3)
```



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 );
}
```

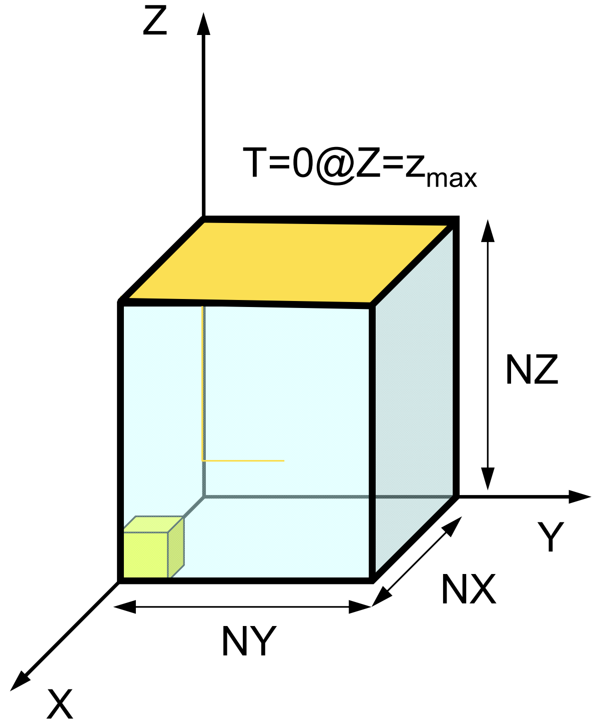
Same interface with FORTRAN

INPUT_GRID (2/3)

```
!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
```

Example of “cube.0” Element (1/2)

```
   64                                        = ICELTOT

  361  361  361  361  361  361  361  361  361  361
  361  361  361  361  361  361  361  361  361  361
  361  361  361  361  361  361  361  361  361  361
  361  361  361  361  361  361  361  361  361  361
  361  361  361  361  361  361  361  361  361  361
  361  361  361  361  361  361  361  361  361  361
  361  361  361  361
```

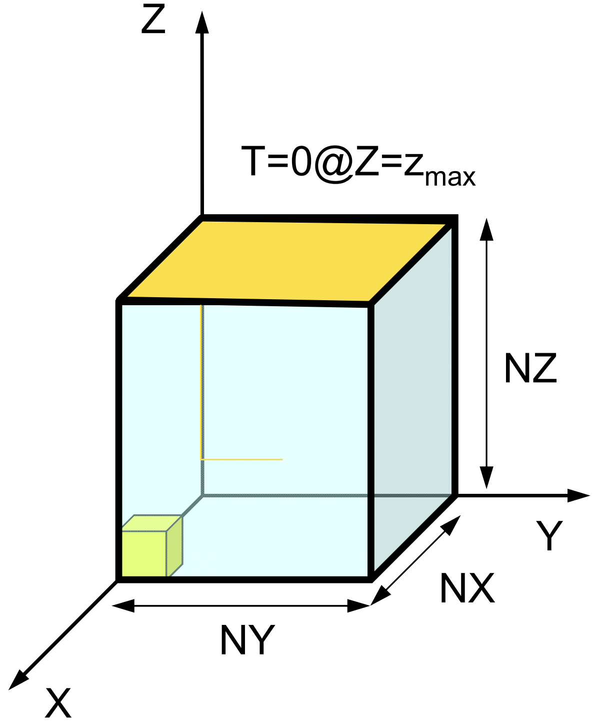


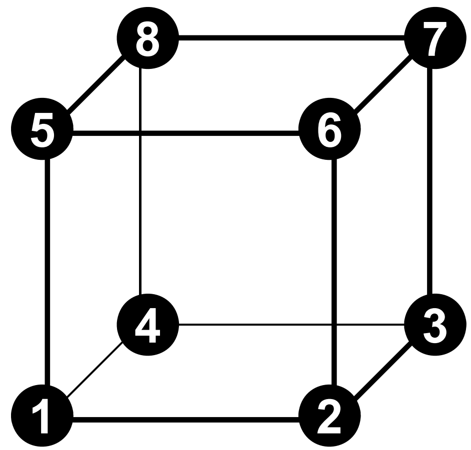
Element Type: 361

3D, Hexahedron, Linear (1st order)

Example of “cube.0” Element (2/2)

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
. . .									
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
	iMAT	ICELNOD(icel,8)							





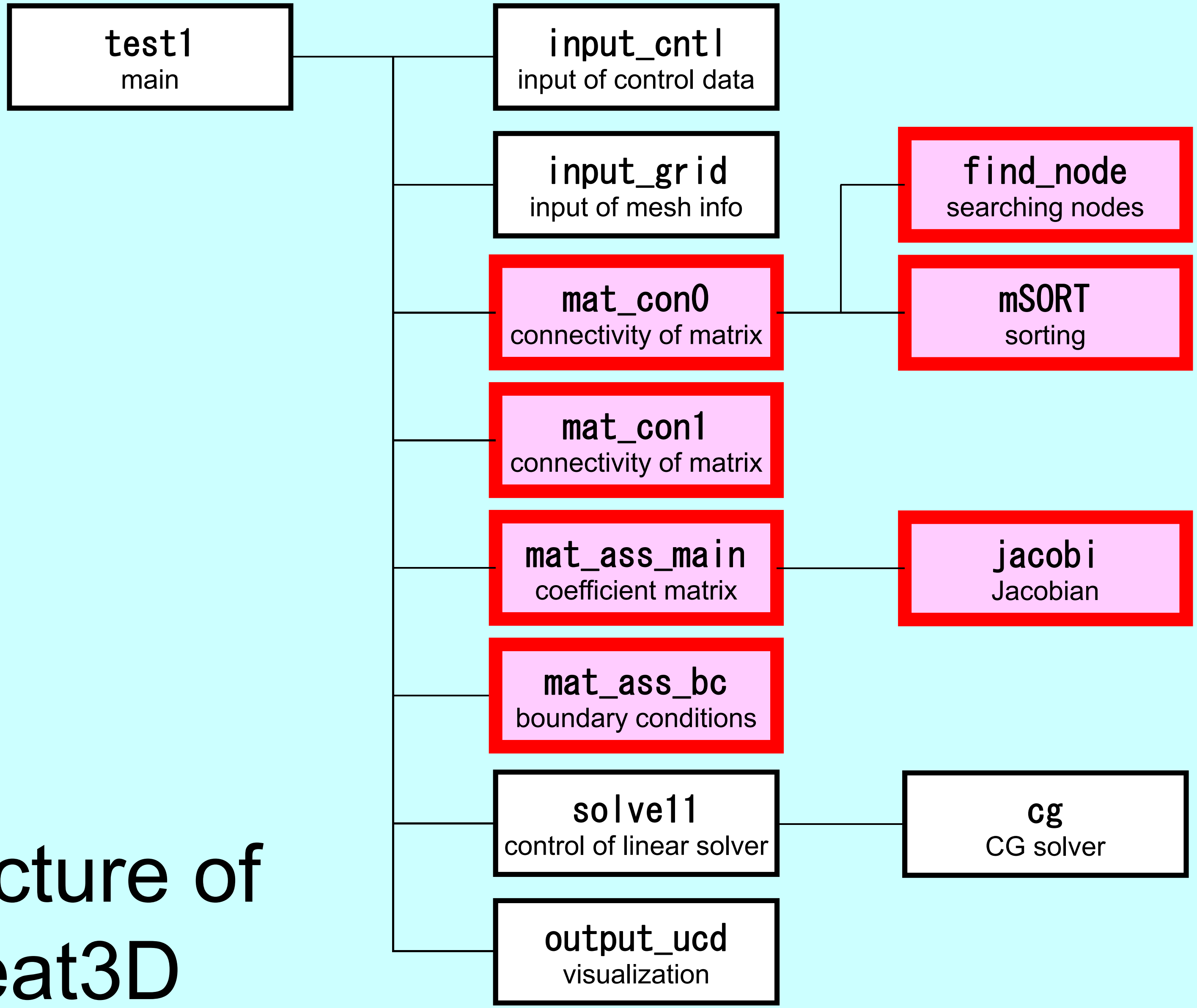
INPUT_GRID (3/3)

```
!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
```

Example of "cube.0" Node Grp. Info.

```
    4                                                       NODGRPtot
   25   50   75  100                                        NODGTP_INDEX(1-4)

 Xmin                                                       NODGRP_NAME(1)
    1    6   11   16   21   26   31   36   41   46          NODGRP_ITEM(1-25)
   51   56   61   66   71   76   81   86   91   96
  101  106  111  116  121

 Ymin                                                       NODGRP_NAME(2)
    1    2    3    4    5   26   27   28   29   30          NODGRP_ITEM(26-50)
   51   52   53   54   55   76   77   78   79   80
  101  102  103  104  105

 Zmin                                                       NODGRP_NAME(3)
    1    2    3    4    5    6    7    8    9   10          NODGRP_ITEM(51-75)
   11   12   13   14   15   16   17   18   19   20
   21   22   23   24   25

 Zmax                                                       NODGRP_NAME(4)
  101  102  103  104  105  106  107  108  109  110          NODGRP_ITEM(76-100)
  111  112  113  114  115  116  117  118  119  120
  121  122  123  124  125
```

Structure of heat3D



Global Variables: pfem_util.f (1/3)

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

Global Variables: pfem_util.f (2/3)

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

Global Variables: pfem_util.f (3/3)

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

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

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

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.

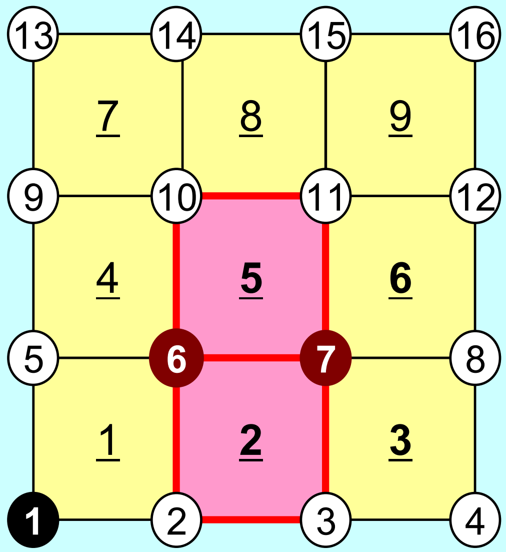
movie

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
```



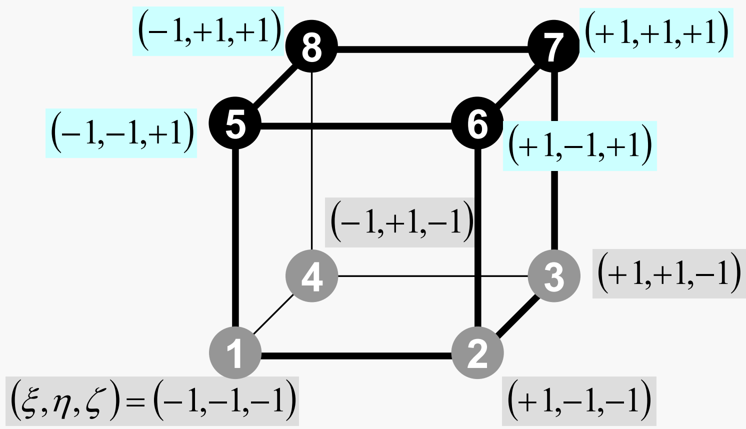
MAT_CON0: generates INU, IALU

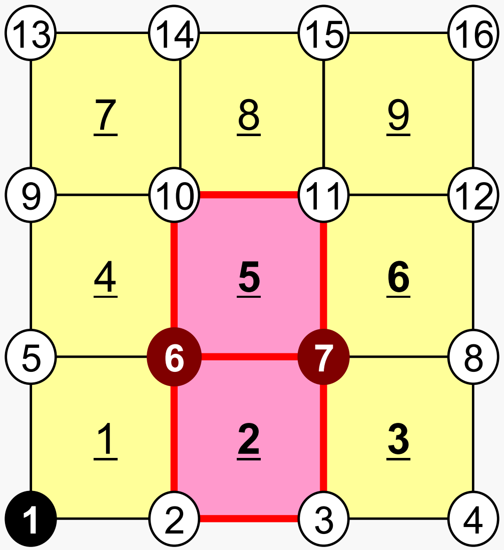
MAT_CON1: generates index, item

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_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
```

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

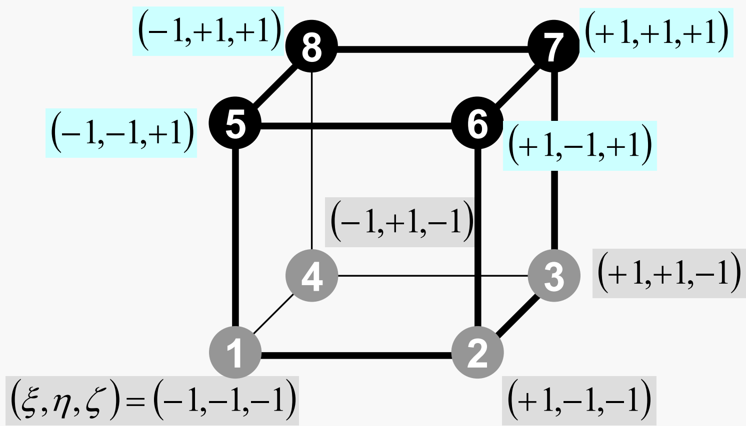
Generating Connectivity of Matrix MAT_CON0 (1/4)

```
!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
```

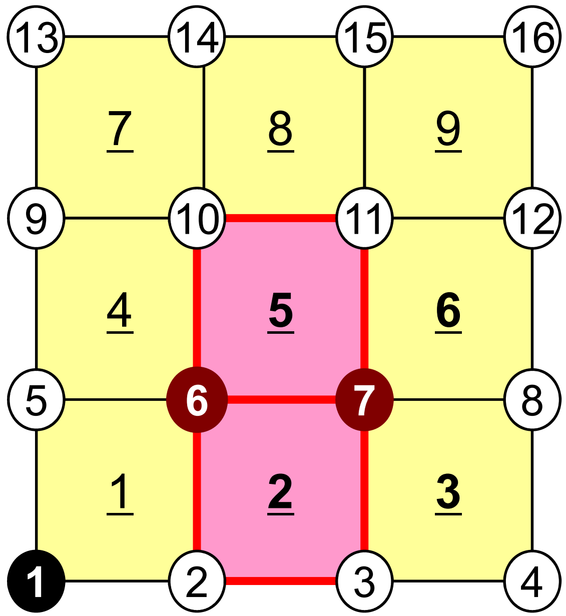
Array	Size	Description
INLU	(N)	Number of connected nodes to each node (lower/upper)
IALU	(N, NLU)	Corresponding connected node ID (column ID)

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

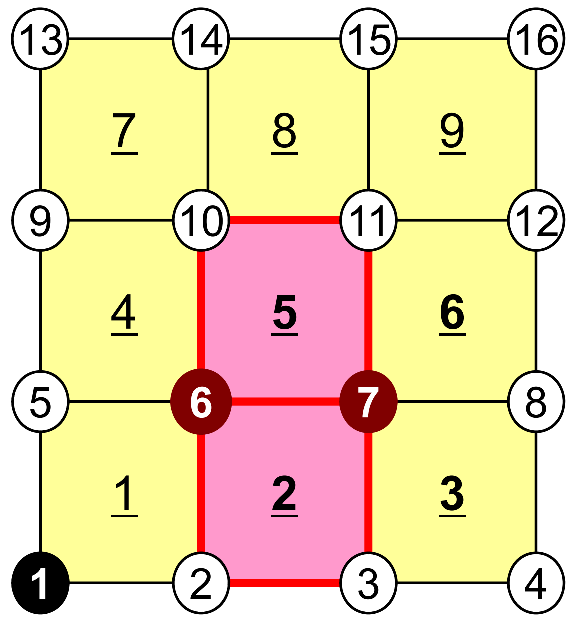


FIND_TS_NODE: Search Connectivity

INLU,IALU: Automatic Search

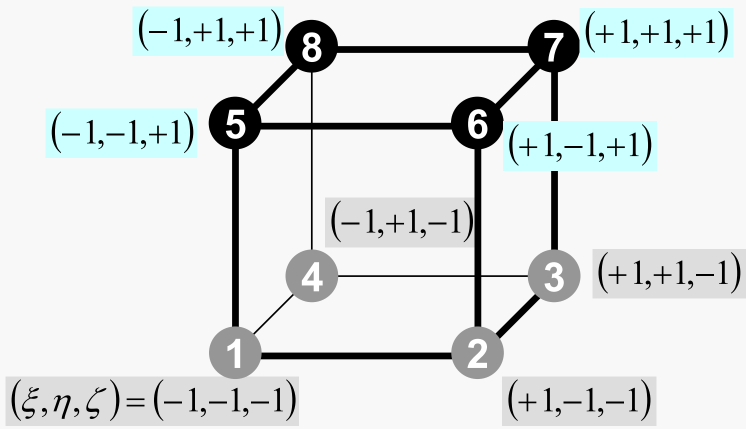
```
!C
!C***
!C*** FIND_TS_NODE
!C***
!C
        subroutine FIND_TS_NODE (ip1, ip2)

          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

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

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

FORTRAN

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

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

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

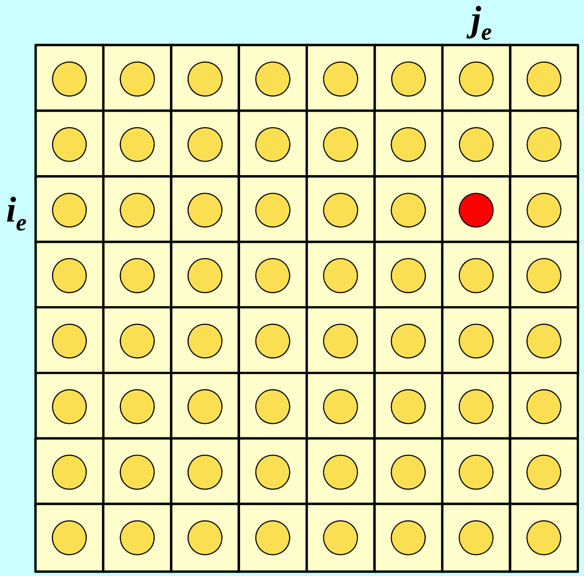
Main Part

```
      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: Overview

```
do kpn= 1, 2          Gaussian Quad. points in ζ-direction
  do jpn= 1, 2        Gaussian Quad. points in η-direction
    do ipn= 1, 2      Gaussian Quad. Pointe in ξ-direction
      Define Shape Function at Gaussian Quad. Points (8-points)
      Its derivative on natural/local coordinate is also defined.
    enddo
  enddo
enddo
```

```
do icel= 1, ICELTOT    Loop for Element
  Jacobian and derivative on global coordinate of shape functions at
  Gaussian Quad. Points are defined according to coordinates of 8 nodes. (JACOBI)

  do ie= 1, 8          Local Node ID
    do je= 1, 8        Local Node ID
      Global Node ID: ip, jp
      Address of A_ip,jp in "item" : kk

      do kpn= 1, 2        Gaussian Quad. points in ζ-direction
        do jpn= 1, 2      Gaussian Quad. points in η-direction
          do ipn= 1, 2    Gaussian Quad. points in ξ-direction
            integration on each element
            coefficients of element matrices
            accumulation to global matrix
          enddo
        enddo
      enddo
    enddo
  enddo
enddo
```



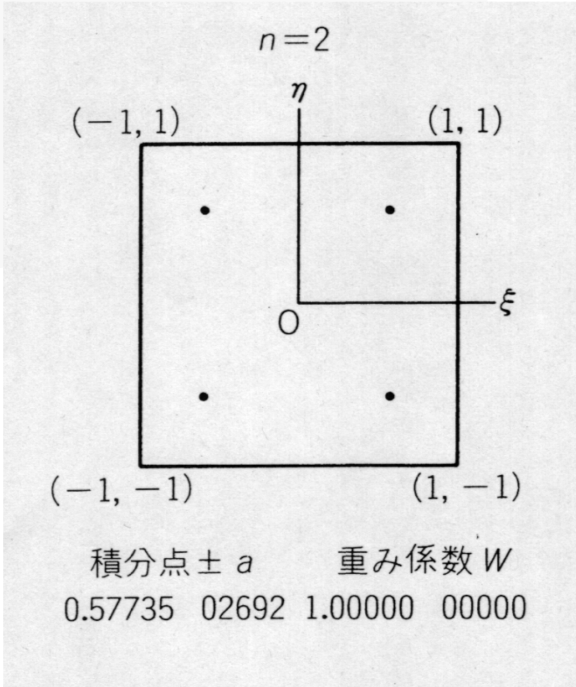
MAT_ASS_MAIN (1/6)

```
!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                    Non-Zero Off-Diagonal components (coef. matrix)
         B= 0.d0                    RHS vector
         X= 0.d0                    Unknowns
         D= 0.d0                    Diagonal components (coef. matrix)

      WEI(1)= +1.0000000000D+00
      WEI(2)= +1.0000000000D+00

      POS(1)= -0.5773502692D+00
      POS(2)= +0.5773502692D+00
```

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     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)= O8th * QM1 * EM1 * TM1
        SHAPE(ip,jp,kp,2)= O8th * QP1 * EM1 * TM1
        SHAPE(ip,jp,kp,3)= O8th * QP1 * EP1 * TM1
        SHAPE(ip,jp,kp,4)= O8th * QM1 * EP1 * TM1
        SHAPE(ip,jp,kp,5)= O8th * QM1 * EM1 * TP1
        SHAPE(ip,jp,kp,6)= O8th * QP1 * EM1 * TP1
        SHAPE(ip,jp,kp,7)= O8th * QP1 * EP1 * TP1
        SHAPE(ip,jp,kp,8)= O8th * QM1 * 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)= O8th * QM1 * EM1 * TM1
        SHAPE(ip,jp,kp,2)= O8th * QP1 * EM1 * TM1
        SHAPE(ip,jp,kp,3)= O8th * QP1 * EP1 * TM1
        SHAPE(ip,jp,kp,4)= O8th * QM1 * EP1 * TM1
        SHAPE(ip,jp,kp,5)= O8th * QM1 * EM1 * TP1
        SHAPE(ip,jp,kp,6)= O8th * QP1 * EM1 * TP1
        SHAPE(ip,jp,kp,7)= O8th * QP1 * EP1 * TP1
        SHAPE(ip,jp,kp,8)= O8th * QM1 * EP1 * TP1
```

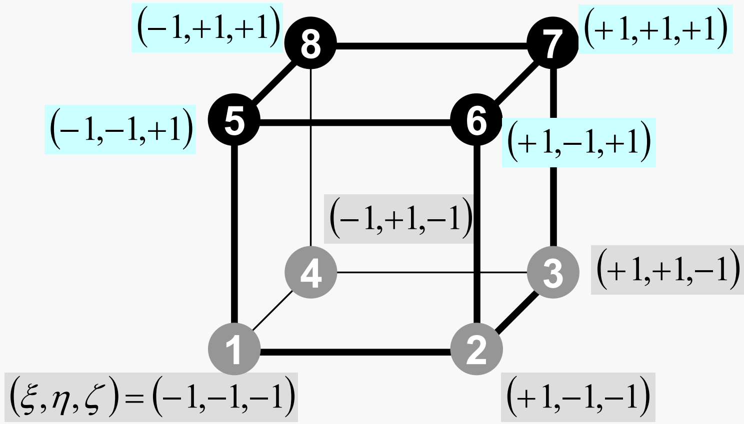
$$QP1(\mathrm{i})=(1+\xi_i), \quad QM1(\mathrm{i})=(1-\xi_i)$$

$$EP1(\mathrm{j})=(1+\eta_j), \quad EM1(\mathrm{j})=(1-\eta_i)$$

$$TP1(\mathrm{k})=(1+\zeta_k), \quad TM1(\mathrm{k})=(1-\zeta_k)$$

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)= O8th * QM1 * EM1 * TM1
        SHAPE(ip,jp,kp,2)= O8th * QP1 * EM1 * TM1
        SHAPE(ip,jp,kp,3)= O8th * QP1 * EP1 * TM1
        SHAPE(ip,jp,kp,4)= O8th * QM1 * EP1 * TM1
        SHAPE(ip,jp,kp,5)= O8th * QM1 * EM1 * TP1
        SHAPE(ip,jp,kp,6)= O8th * QP1 * EM1 * TP1
        SHAPE(ip,jp,kp,7)= O8th * QP1 * EP1 * TP1
        SHAPE(ip,jp,kp,8)= O8th * QM1 * 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)= O8th * QM1 * EM1 * TM1
        SHAPE(ip,jp,kp,2)= O8th * QP1 * EM1 * TM1
        SHAPE(ip,jp,kp,3)= O8th * QP1 * EP1 * TM1
        SHAPE(ip,jp,kp,4)= O8th * QM1 * EP1 * TM1
        SHAPE(ip,jp,kp,5)= O8th * QM1 * EM1 * TP1
        SHAPE(ip,jp,kp,6)= O8th * QP1 * EM1 * TP1
        SHAPE(ip,jp,kp,7)= O8th * QP1 * EP1 * TP1
        SHAPE(ip,jp,kp,8)= O8th * QM1 * 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)= - O8th * EM1 * TM1
      PNQ(jp,kp,2)= + O8th * EM1 * TM1
      PNQ(jp,kp,3)= + O8th * EP1 * TM1
      PNQ(jp,kp,4)= - O8th * EP1 * TM1
      PNQ(jp,kp,5)= - O8th * EM1 * TP1
      PNQ(jp,kp,6)= + O8th * EM1 * TP1
      PNQ(jp,kp,7)= + O8th * EP1 * TP1
      PNQ(jp,kp,8)= - O8th * EP1 * TP1
      PNE(ip,kp,1)= - O8th * QM1 * TM1
      PNE(ip,kp,2)= - O8th * QP1 * TM1
      PNE(ip,kp,3)= + O8th * QP1 * TM1
      PNE(ip,kp,4)= + O8th * QM1 * TM1
      PNE(ip,kp,5)= - O8th * QM1 * TP1
      PNE(ip,kp,6)= - O8th * QP1 * TP1
      PNE(ip,kp,7)= + O8th * QP1 * TP1
      PNE(ip,kp,8)= + O8th * QM1 * TP1
      PNT(ip,jp,1)= - O8th * QM1 * EM1
      PNT(ip,jp,2)= - O8th * QP1 * EM1
      PNT(ip,jp,3)= - O8th * QP1 * EP1
      PNT(ip,jp,4)= - O8th * QM1 * EP1
      PNT(ip,jp,5)= + O8th * QM1 * EM1
      PNT(ip,jp,6)= + O8th * QP1 * EM1
      PNT(ip,jp,7)= + O8th * QP1 * EP1
      PNT(ip,jp,8)= + O8th * QM1 * EP1
    enddo
    enddo
    enddo

    do icel= 1, ICELTOT
      COND0= 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)
```

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

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

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

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

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

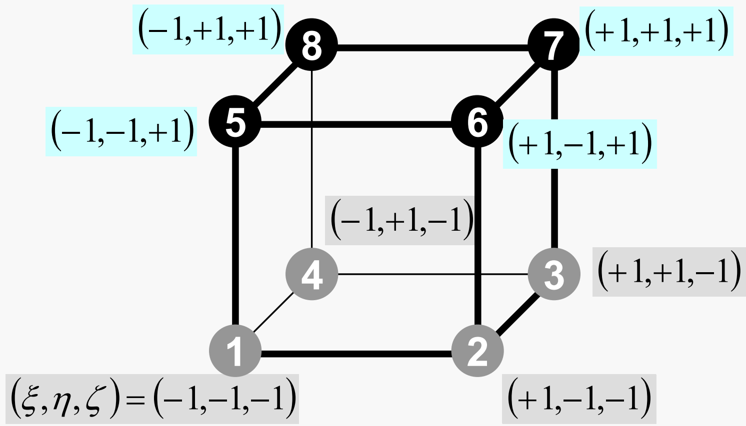
$$\frac{\partial N_3}{\partial \xi}(\xi_i, \eta_j, \zeta_k) = +\frac{1}{8}(1+\eta_j)(1-\zeta_k)$$

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

First Order Derivative of Shape Functions at (ξ_i, η_j, ζ_k)

MAT_ASS_MAIN (3/6)

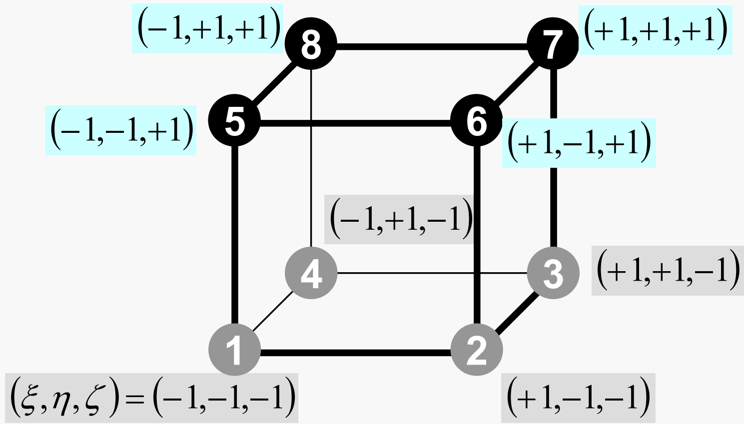
```
    PNQ(jp,kp,1)= - O8th * EM1 * TM1
    PNQ(jp,kp,2)= + O8th * EM1 * TM1
    PNQ(jp,kp,3)= + O8th * EP1 * TM1
    PNQ(jp,kp,4)= - O8th * EP1 * TM1
    PNQ(jp,kp,5)= - O8th * EM1 * TP1
    PNQ(jp,kp,6)= + O8th * EM1 * TP1
    PNQ(jp,kp,7)= + O8th * EP1 * TP1
    PNQ(jp,kp,8)= - O8th * EP1 * TP1
    PNE(ip,kp,1)= - O8th * QM1 * TM1
    PNE(ip,kp,2)= - O8th * QP1 * TM1
    PNE(ip,kp,3)= + O8th * QP1 * TM1
    PNE(ip,kp,4)= + O8th * QM1 * TM1
    PNE(ip,kp,5)= - O8th * QM1 * TP1
    PNE(ip,kp,6)= - O8th * QP1 * TP1
    PNE(ip,kp,7)= + O8th * QP1 * TP1
    PNE(ip,kp,8)= + O8th * QM1 * TP1
    PNT(ip,jp,1)= - O8th * QM1 * EM1
    PNT(ip,jp,2)= - O8th * QP1 * EM1
    PNT(ip,jp,3)= - O8th * QP1 * EP1
    PNT(ip,jp,4)= - O8th * QM1 * EP1
    PNT(ip,jp,5)= + O8th * QM1 * EM1
    PNT(ip,jp,6)= + O8th * QP1 * EM1
    PNT(ip,jp,7)= + O8th * QP1 * EP1
    PNT(ip,jp,8)= + O8th * QM1 * EP1
  enddo
  enddo
  enddo

  do icel= 1, ICELTOT
    COND0= 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

      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= O8th * (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 )
```

Node ID (Global)



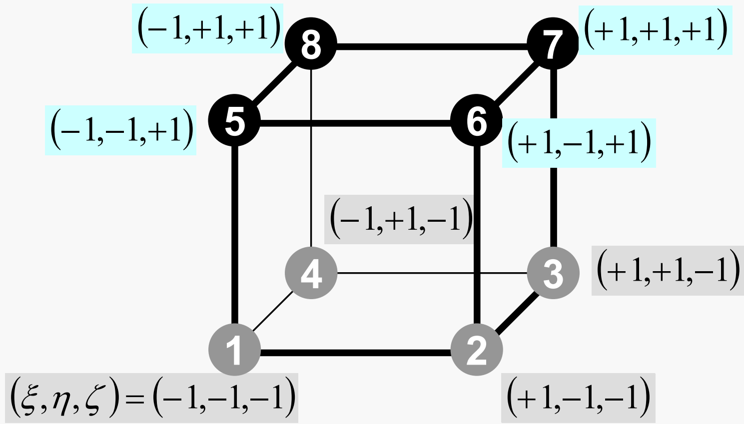
MAT_ASS_MAIN (4/6)

```
      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= O8th * (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 )
```

X-Coordinates of 8 nodes

Y-Coordinates of 8 nodes

Z-Coordinates of 8 nodes

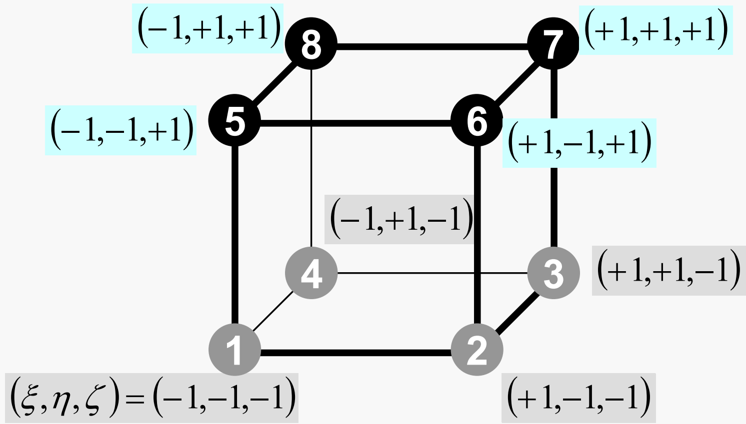


MAT_ASS_MAIN (4/6)

```
      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= O8th * (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 )
```

X-Coordinates of 8 nodes

Y-Coordinates of 8 nodes



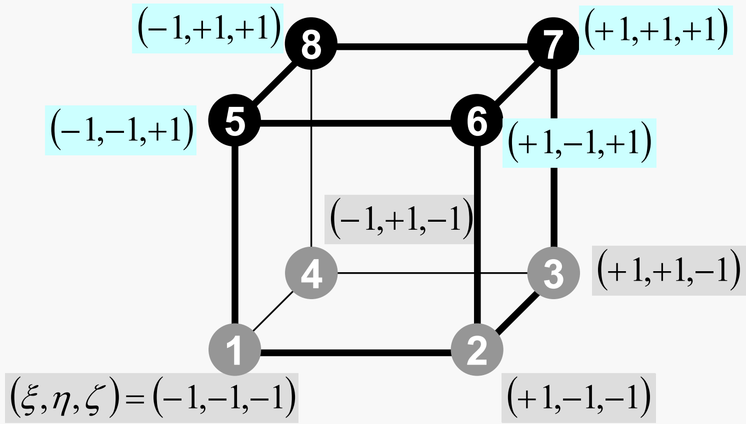
$$\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(\lambda\frac{\partial T}{\partial z}\right)+\dot{Q}(x,y,z)=0$$

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

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= O8th * (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= O8th * (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 )
```

JACOBI (1/4)

```
      subroutine 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 )

!C
!C    calculates JACOBIAN & INVERSE JACOBIAN
!C             dNi/dx, dNi/dy & dNi/dz
!C

      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 PNX(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:

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

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

$[J]$: Jacobi matrix, Jacobian

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

- Components of Jacobian:

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

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

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

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

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

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

JACOBI (2/4)

```
!C
!C==   DETERMINANT of the JACOBIAN

        dXdQ =                                                     &
     &          + PNQ(jp,kp,1) * X1 + PNQ(jp,kp,2) * X2            &
     &          + PNQ(jp,kp,3) * X3 + PNQ(jp,kp,4) * X4            &
     &          + PNQ(jp,kp,5) * X5 + PNQ(jp,kp,6) * X6            &
     &          + PNQ(jp,kp,7) * X7 + PNQ(jp,kp,8) * X8
        dYdQ =                                                     &
     &          + PNQ(jp,kp,1) * Y1 + PNQ(jp,kp,2) * Y2            &
     &          + PNQ(jp,kp,3) * Y3 + PNQ(jp,kp,4) * Y4            &
     &          + PNQ(jp,kp,5) * Y5 + PNQ(jp,kp,6) * Y6            &
     &          + PNQ(jp,kp,7) * Y7 + PNQ(jp,kp,8) * Y8
        dZdQ =                                                     &
     &          + PNQ(jp,kp,1) * Z1 + PNQ(jp,kp,2) * Z2            &
     &          + PNQ(jp,kp,3) * Z3 + PNQ(jp,kp,4) * Z4            &
     &          + PNQ(jp,kp,5) * Z5 + PNQ(jp,kp,6) * Z6            &
     &          + PNQ(jp,kp,7) * Z7 + PNQ(jp,kp,8) * Z8
        dXdE =                                                     &
     &          + PNE(ip,kp,1) * X1 + PNE(ip,kp,2) * X2            &
     &          + PNE(ip,kp,3) * X3 + PNE(ip,kp,4) * X4            &
     &          + PNE(ip,kp,5) * X5 + PNE(ip,kp,6) * X6            &
     &          + PNE(ip,kp,7) * X7 + PNE(ip,kp,8) * X8
        dYdE =                                                     &
     &          + PNE(ip,kp,1) * Y1 + PNE(ip,kp,2) * Y2            &
     &          + PNE(ip,kp,3) * Y3 + PNE(ip,kp,4) * Y4            &
     &          + PNE(ip,kp,5) * Y5 + PNE(ip,kp,6) * Y6            &
     &          + PNE(ip,kp,7) * Y7 + PNE(ip,kp,8) * Y8
        dZdE =                                                     &
     &          + PNE(ip,kp,1) * Z1 + PNE(ip,kp,2) * Z2            &
     &          + PNE(ip,kp,3) * Z3 + PNE(ip,kp,4) * Z4            &
     &          + PNE(ip,kp,5) * Z5 + PNE(ip,kp,6) * Z6            &
     &          + PNE(ip,kp,7) * Z7 + PNE(ip,kp,8) * Z8
```

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

$$\mathrm{dXdQ} = \frac{\partial x}{\partial \xi} = J_{11}$$

$$\mathrm{dYdQ} = \frac{\partial y}{\partial \xi} = J_{12}$$

$$\mathrm{dZdQ} = \frac{\partial z}{\partial \xi} = J_{13}$$

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 × 3))

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

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
          PNX(ip,jp,kp,1)= a11*PNQ(jp,kp,1) + a12*PNE(ip,kp,1) + a13*PNT(ip,jp,1)
          PNX(ip,jp,kp,2)= a11*PNQ(jp,kp,2) + a12*PNE(ip,kp,2) + a13*PNT(ip,jp,2)
          PNX(ip,jp,kp,3)= a11*PNQ(jp,kp,3) + a12*PNE(ip,kp,3) + a13*PNT(ip,jp,3)
          PNX(ip,jp,kp,4)= a11*PNQ(jp,kp,4) + a12*PNE(ip,kp,4) + a13*PNT(ip,jp,4)
          PNX(ip,jp,kp,5)= a11*PNQ(jp,kp,5) + a12*PNE(ip,kp,5) + a13*PNT(ip,jp,5)
          PNX(ip,jp,kp,6)= a11*PNQ(jp,kp,6) + a12*PNE(ip,kp,6) + a13*PNT(ip,jp,6)
          PNX(ip,jp,kp,7)= a11*PNQ(jp,kp,7) + a12*PNE(ip,kp,7) + a13*PNT(ip,jp,7)
          PNX(ip,jp,kp,8)= a11*PNQ(jp,kp,8) + a12*PNE(ip,kp,8) + a13*PNT(ip,jp,8)

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

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

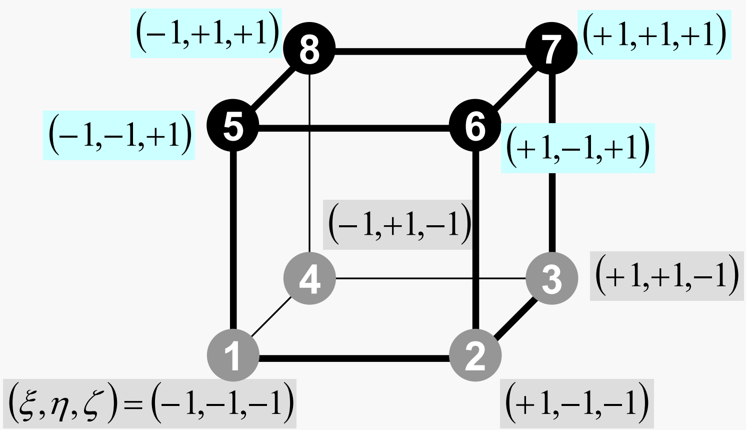
Non-Zero Off-Diagonal Block in Global Matix

$A_{ip,jp}$

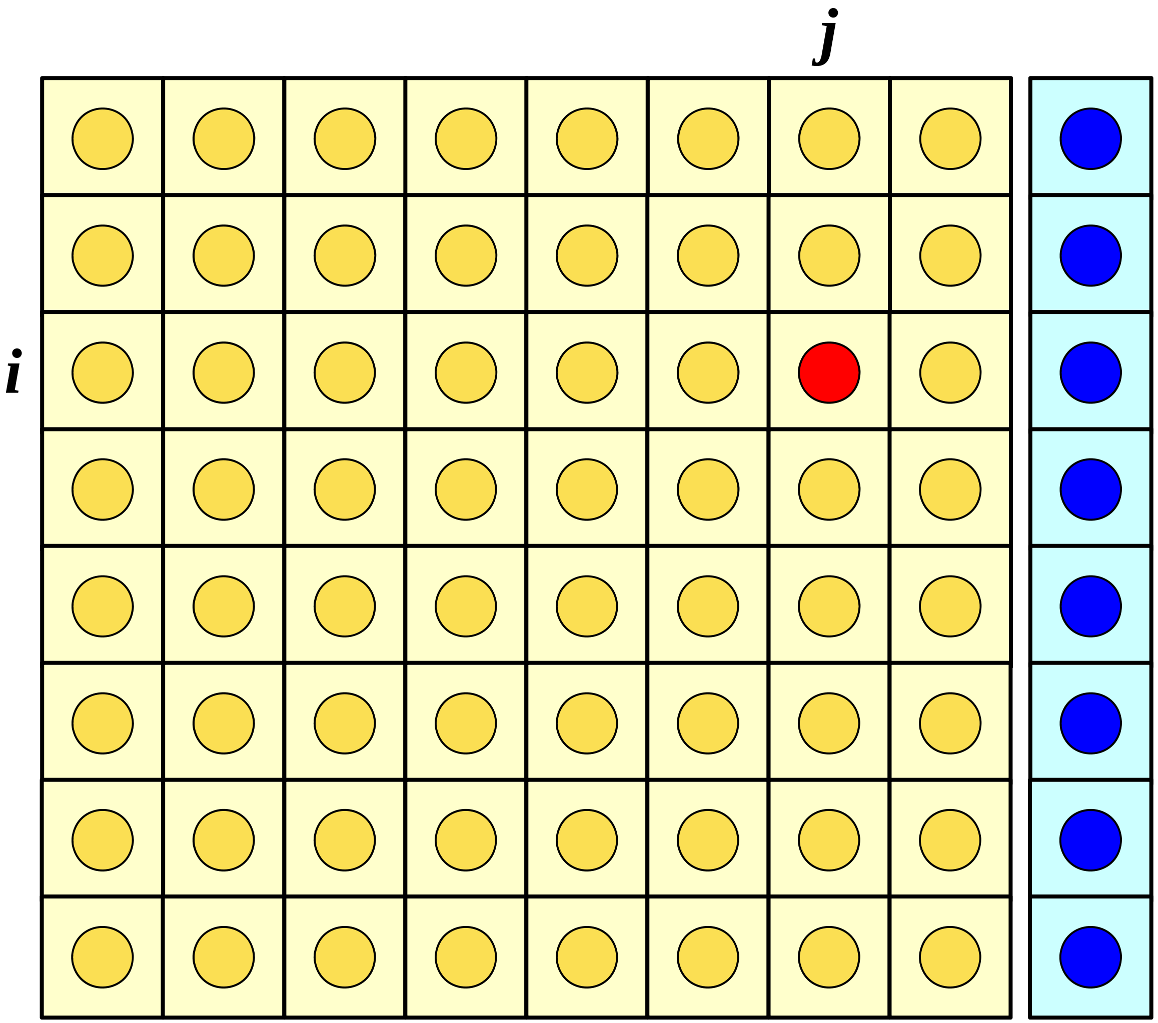
kk: address in “item”

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

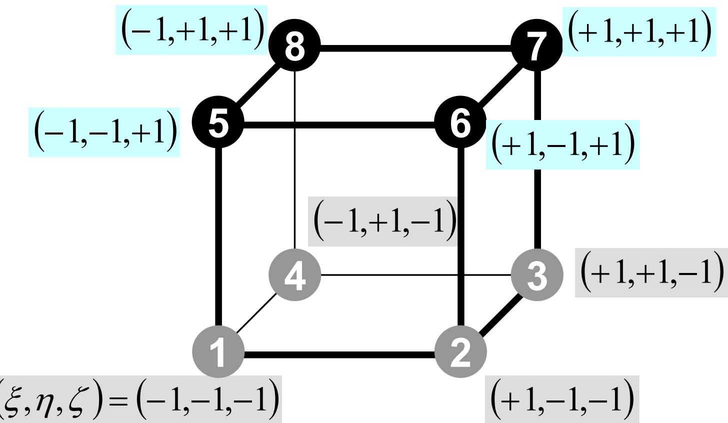
Node ID (ip,jp) starting from 1



Element Matrix: 8x8



$$\lfloor k_{ij} \rfloor \quad (i, j = 1 \ldots 8)$$



MAT_ASS_MAIN (5/6)

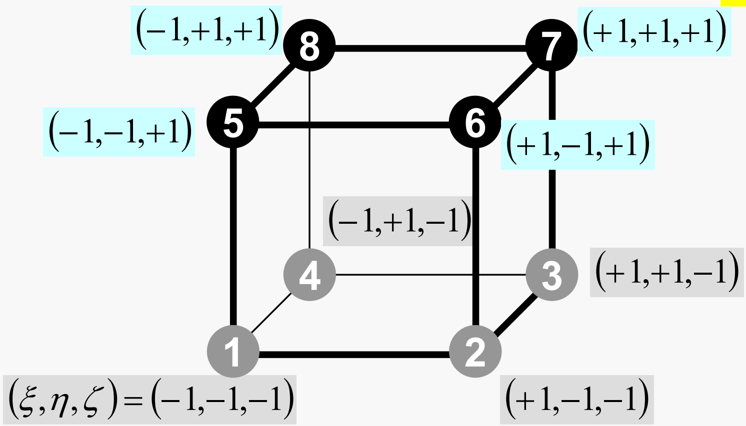
```
!C
!C== CONSTRUCT the GLOBAL MATRIX
        do ie= 1, 8
          ip = nodLOCAL(ie)
        do je= 1, 8
          jp = nodLOCAL(je)

          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
```

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)

```
          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 * COND0 *
     &                      (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
```

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

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
```

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

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

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

MAT_ASS_MAIN (6/6)

```
        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 * COND0 *
     &                     (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
```

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

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

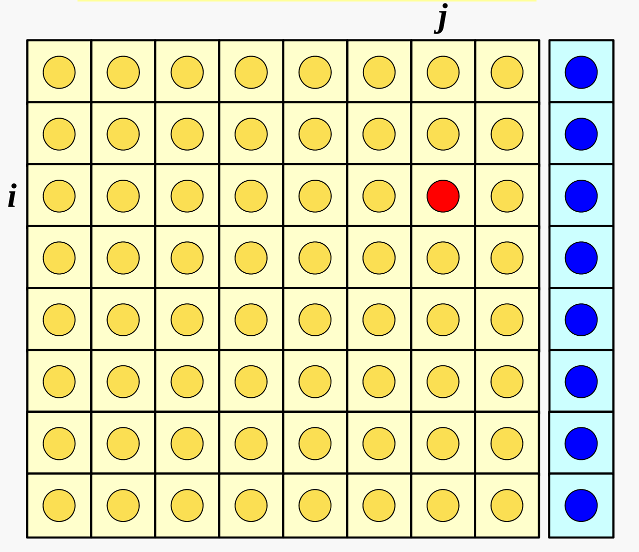
$$= \sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{k=1}^{N}\left[W_i \cdot W_j \cdot W_k \cdot f(\xi_i,\eta_j,\zeta_k)\right]$$

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

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
```

$\lfloor k_{ij} \rfloor \quad (i, j = 1 \ldots 8)$



係数行列：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 * COND0 *
 &                      (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
```

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

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

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

$$QVC=|x_C+y_C|$$

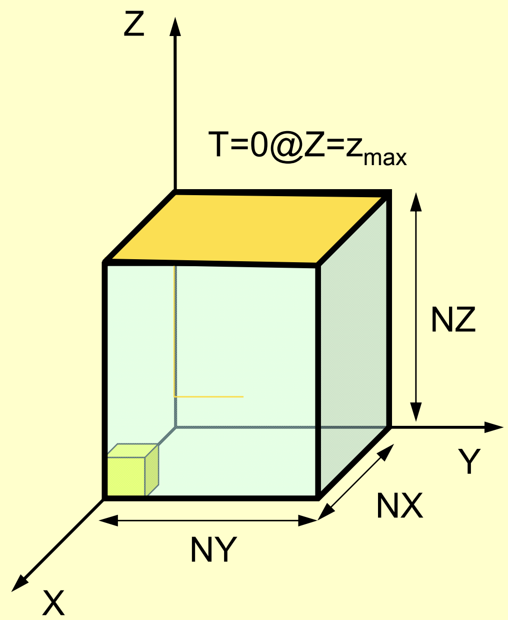
$$QV0=\int_V QVOL[N]^T dV$$

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

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   if corresponding non-zero
                                     off-diagonal node is "marked"
      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
```

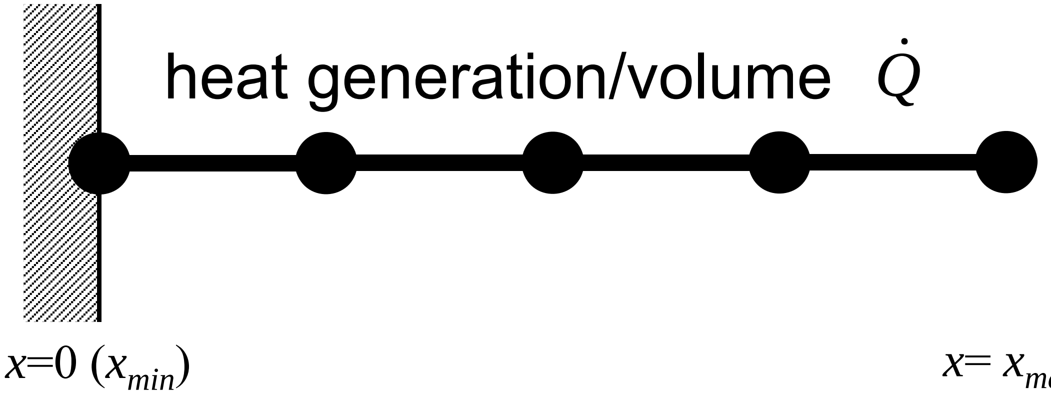
If the node “in” is included in the node group “Zmax”

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

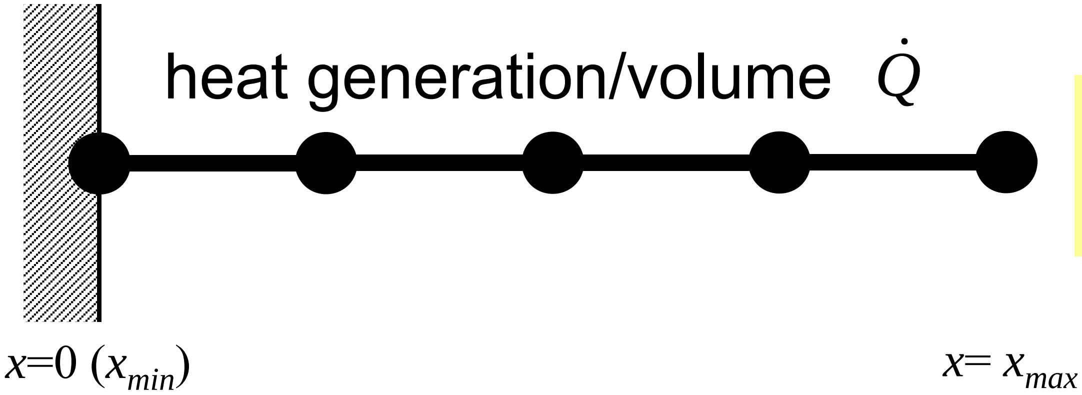


$$\frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right)+\dot{Q}=0$$

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

1D

(Linear) Equation at x=0
$T_1 = 0$ (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: λ
- Heat Generation Rate/Volume/Time [QL^{-3}T^{-1}] $\dot{Q}$
- Boundary Conditions
  - $x=0$: $T=0$ (Fixed Temperature)
  - $x=x_{max}$: $\dfrac{\partial T}{\partial x} = 0$ (Insulated)

1D

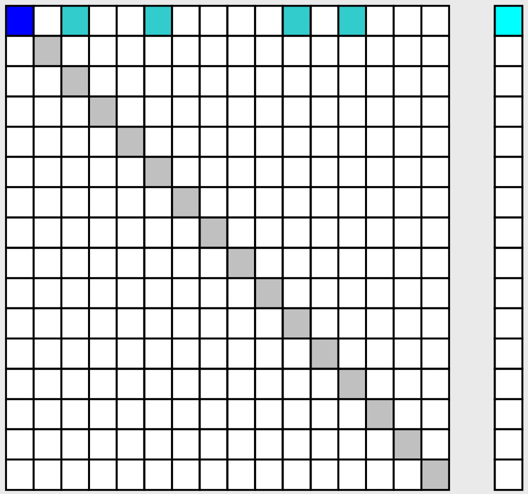
Program: 1d.f (6/6)

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.



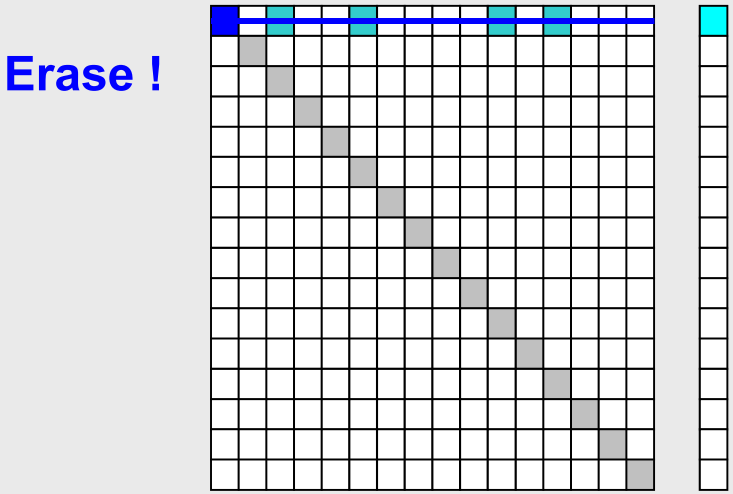
Program: 1d.f (6/6)

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.



Program: 1d.f (6/6)

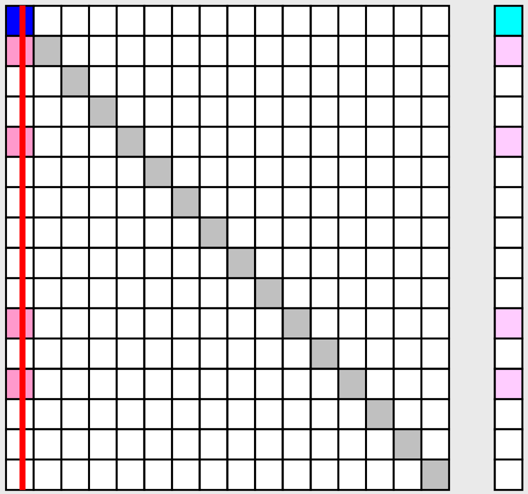
Dirichlet B.C. @ X=0

```
!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)

1D

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===
```

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

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

1D

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===
```

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

$$= Rhs_j - Amat_{k_s} \phi_{Item[k_s]}$$

$$= Rhs_j - Amat_{k_s} \phi_{\min} \quad where\ Item[k_s] = 1$$

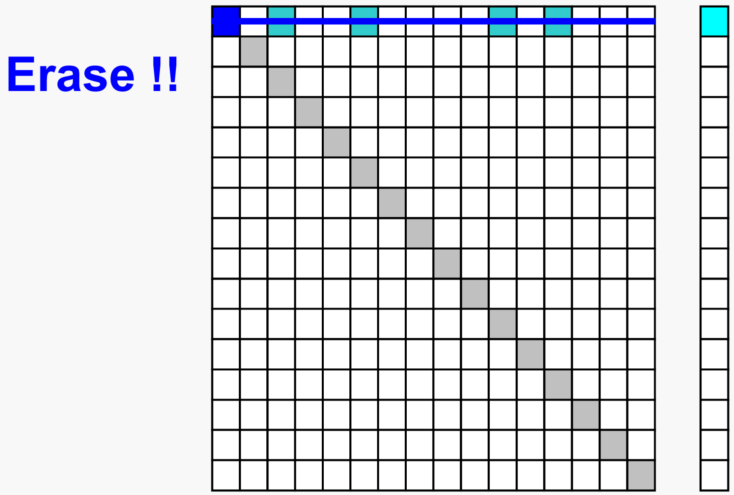
Column components of boundary nodes (Dirichlet B.C.) are moved to RHS and eliminated for keeping symmetrical feature of the matrix.

1D

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

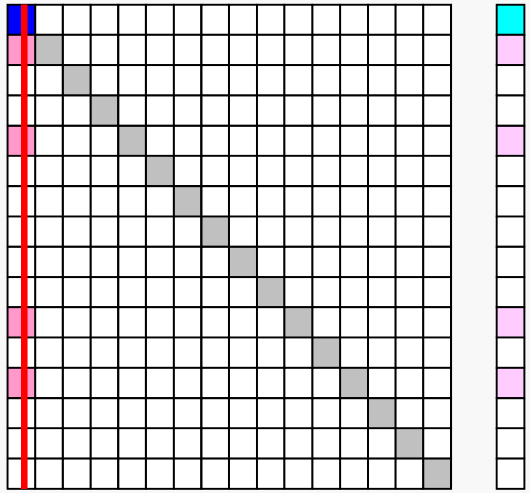


Same as 1D case

MAT_ASS_BC (2/2)

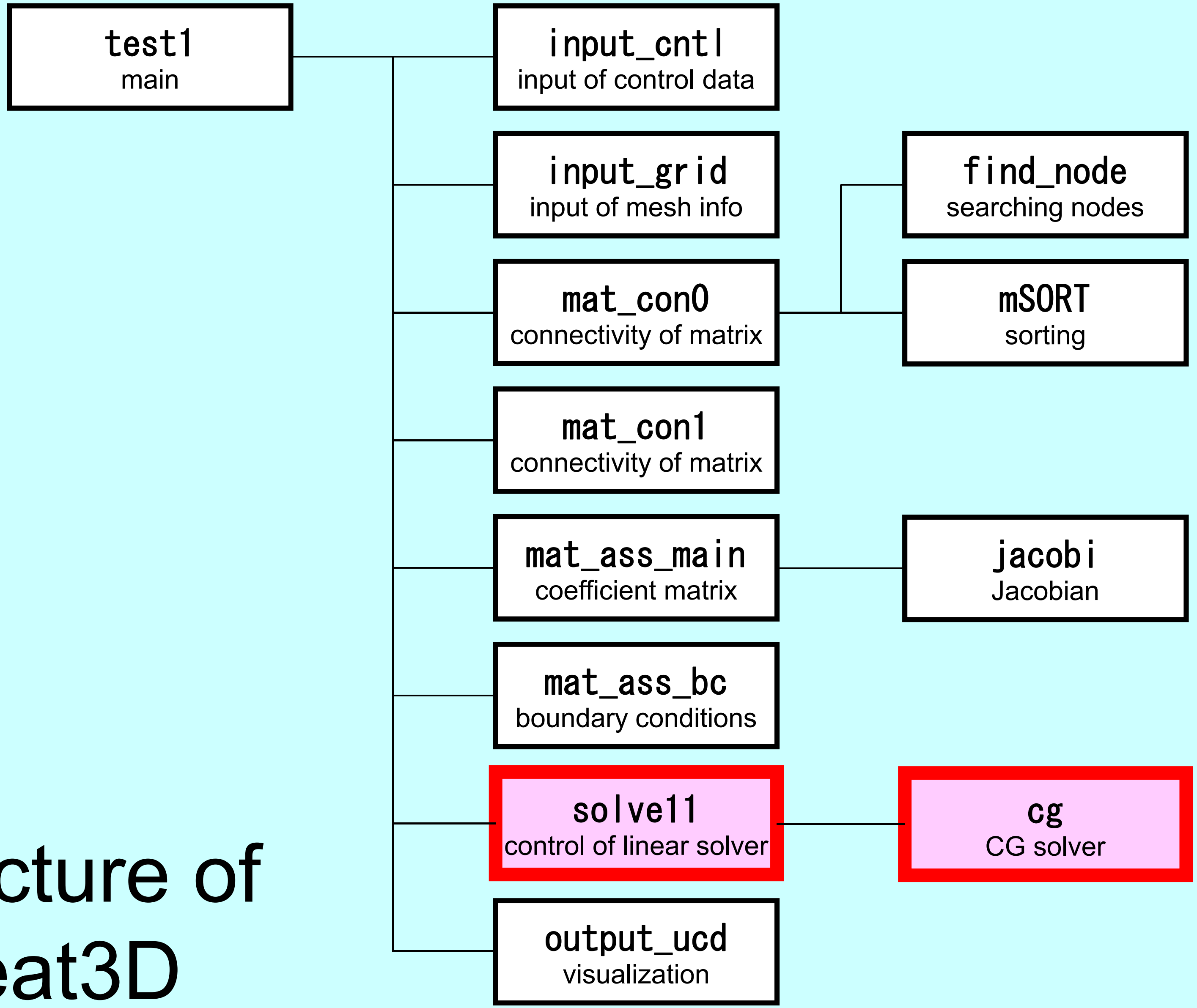
```
      do in= 1, N
        if (IWKX(in,1).eq.1) then
          B(in)= 0.d0
          D(in)= 1.d0

          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



Structure of heat3D

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
```

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)                Max. Iterations for CG
        RESID     = pfemRarray(1)                Convergence Criteria for CG
!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
```

Preconditioned CG Solver

Diagonal Scaling/Point Jacobi Preconditioning

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

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

Diagonal Scaling, Point-Jacobi

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

- **`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)

```
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}
```

```
module solver_CG
contains
      ID,
integer(kind=kint ), intent(inout):: ITER, ERRO
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
```

```
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}
```

Reciprocal numbers（逆数）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 $r^{(0)} = b - [A]x^{(0)}$

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

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|²
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, …
  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===
      RHO0= 0.d0
      do i= 1, N
        RHO0= RHO0 + WW(i,R)*WW(i,Z)
      enddo
      RHO= RHO0
!C===
!C +-----------------------------+
!C | {p} = {z} if      ITER=1    |
!C | BETA= RHO / RHO1  otherwise |
!C +-----------------------------+
!C===
      if ( ITER.eq.1 ) then
        do i= 1, N
          WW(i,P)= WW(i,Z)
        enddo
       else
         BETA= RHO / RHO1
         do i= 1, N
           WW(i,P)= WW(i,Z) + BETA*WW(i,P)
         enddo
      endif
!C===
```

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

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= C10
      ALPHA= RHO / C1
!C===
```

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

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===
        DNRM20= 0.d0
        do i= 1, N
          DNRM20= DNRM20 + WW(i,R)**2
        enddo
        DNRM2= DNRM20
        RESID= dsqrt(DNRM2/BNRM2)

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

        RHO1 = RHO
      enddo
!C===
```

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

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===
        DNRM20= 0.d0
        do i= 1, N
          DNRM20= DNRM20 + WW(i,R)**2
        enddo
        DNRM2= DNRM20
        RESID= dsqrt(DNRM2/BNRM2)

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

        RHO1 = 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

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