

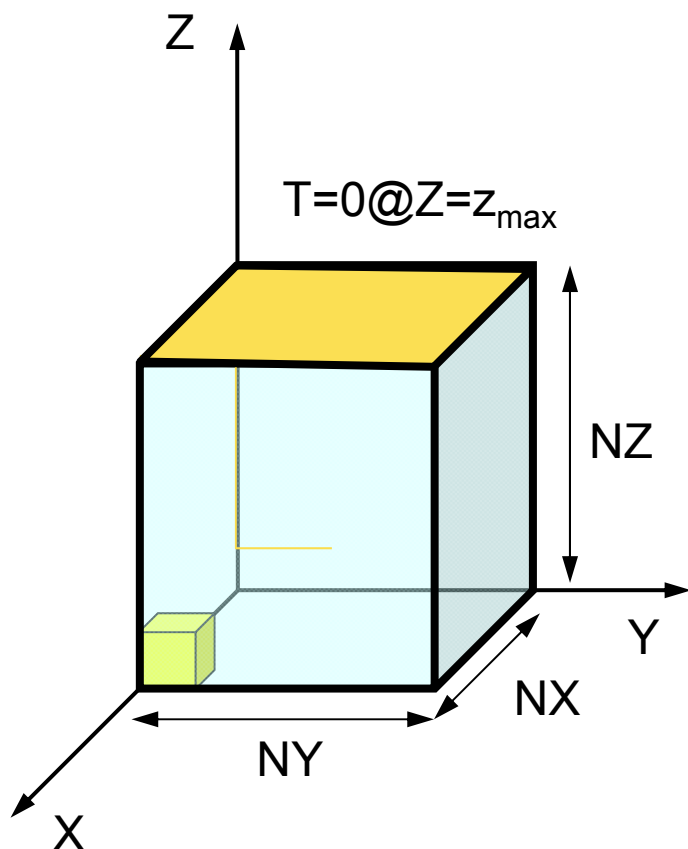
有限要素法による
三次元定常熱伝導解析プログラム
C言語編

中島 研吾

東京大学情報基盤センター

対象とする問題：三次元定常熱伝導

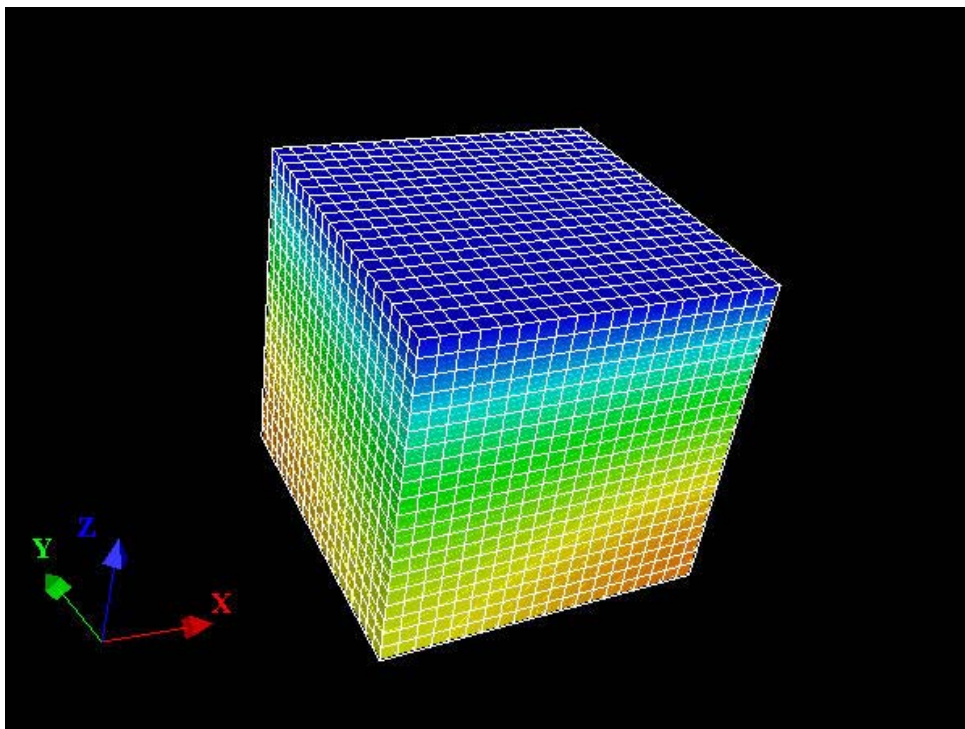
$$\frac{\partial}{\partial x} \left(\lambda \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left(\lambda \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left(\lambda \frac{\partial T}{\partial z} \right) + \dot{Q}(x, y, z) = 0$$



- 定常熱伝導＋発熱
- 一様な熱伝導率 λ
- 直方体
 - 一辺長さ1の立方体（六面体）要素
 - 各方向に $NX \cdot NY \cdot NZ$ 個
- 境界条件
 - $T=0@Z=z_{\max}$
- 体積当たり発熱量は位置（メッシュの中心の座標 x_c, y_c ）に依存
 - $\dot{Q}(x, y, z) = QVOL|x_c + y_c|$

対象とする問題：三次元定常熱伝導

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



- 原点から遠い部分が高温
- 体積当たり発熱量は位置（メッシュの中心の座標）に依存
 - $\dot{Q}(x, y, z) = |x_c + y_c|$

movie

有限要素法の処理

- 支配方程式
- ガラーキン法：弱形式
- 要素単位の積分
 - 要素マトリクス生成
- 全体マトリクス生成
- 境界条件適用
- 連立一次方程式

有限要素法の処理：プログラム

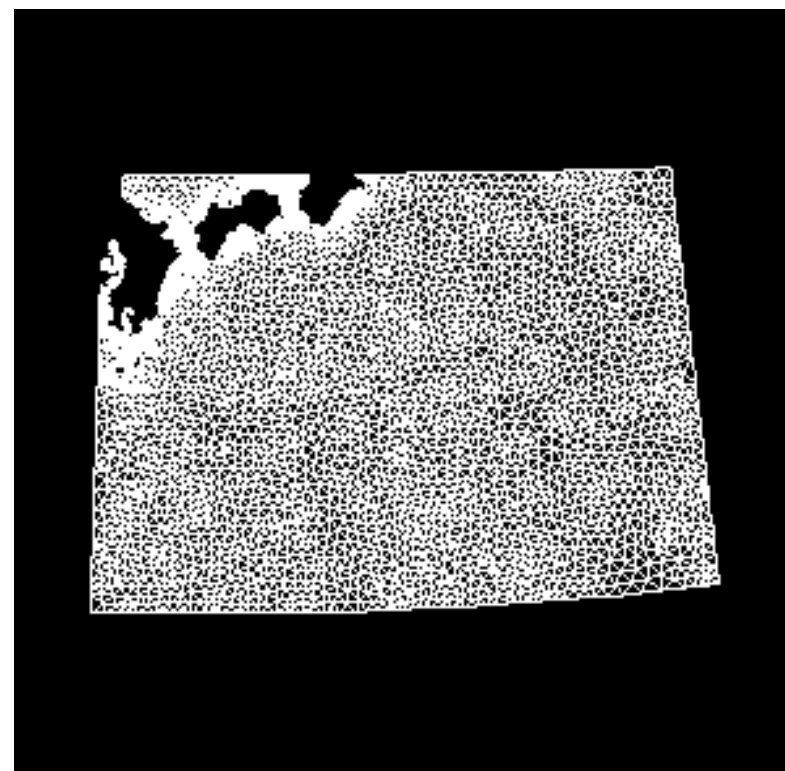
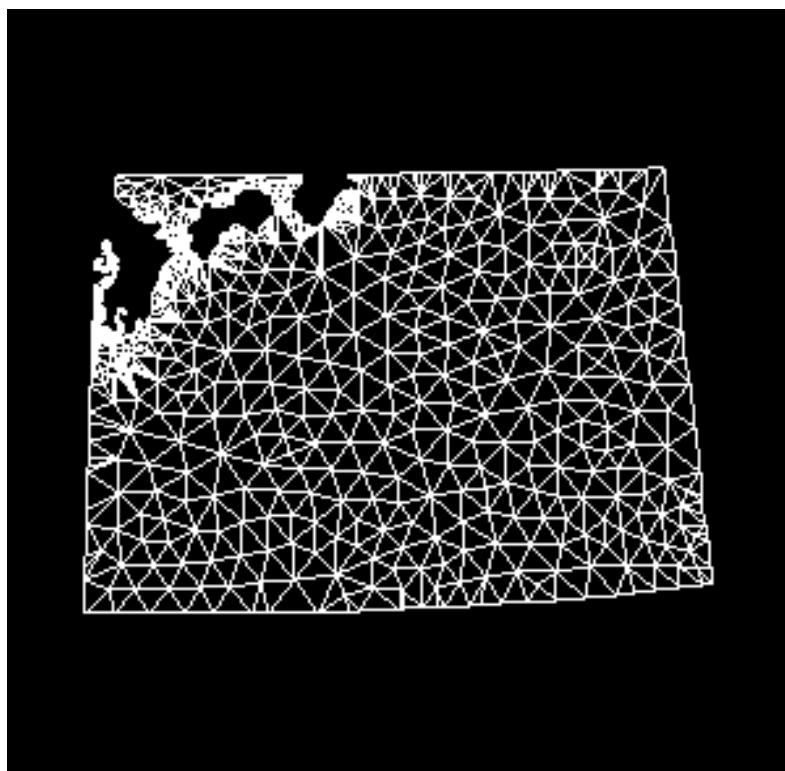
- 初期化
 - 制御変数読み込み
 - 座標読み込み⇒要素生成 (N:節点数, ICELTOT : 要素数)
 - 配列初期化 (全体マトリクス, 要素マトリクス)
 - 要素⇒全体マトリクスマッピング (Index, Item)
- マトリクス生成
 - 要素単位の処理 (do icel= 1, ICELTOT)
 - 要素マトリクス計算
 - 全体マトリクスへの重ね合わせ
 - 境界条件の処理
- 連立一次方程式
 - 共役勾配法 (CG)

- 三次元要素の定式化
- 三次元熱伝導方程式
 - ガラーキン法
 - 要素マトリクス生成

- プログラムの実行
- データ構造
- プログラムの構成

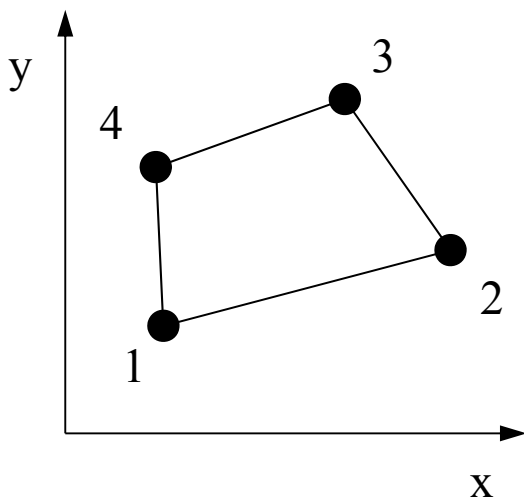
二次元への拡張：三角形要素

- 任意の形状を扱うことができる。
- 特に一次要素は精度が悪く，一部の問題を除いてあまり使用されない。



二次元への拡張：四角形要素

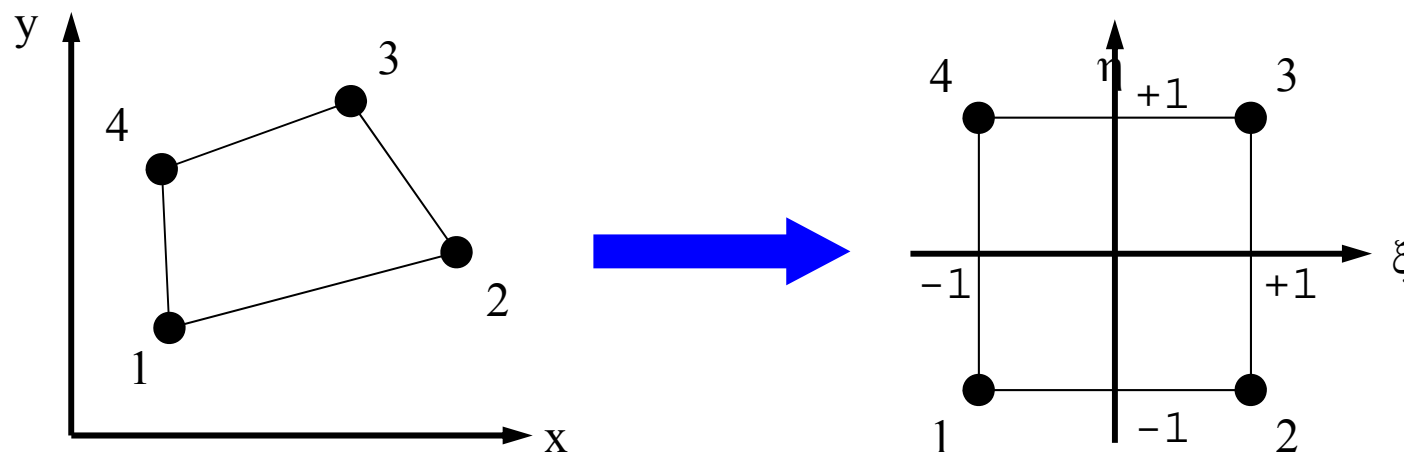
- 一次元要素と同じ形状関数を x, y 軸に適用することによって、四角形要素の定式化は可能である。
 - 三角形と比較して特に低次要素の精度はよい
- しかしながら、各辺が座標軸に平行な長方形でなければならない
 - 差分法と変わらない



- このような形状を扱うことができない。

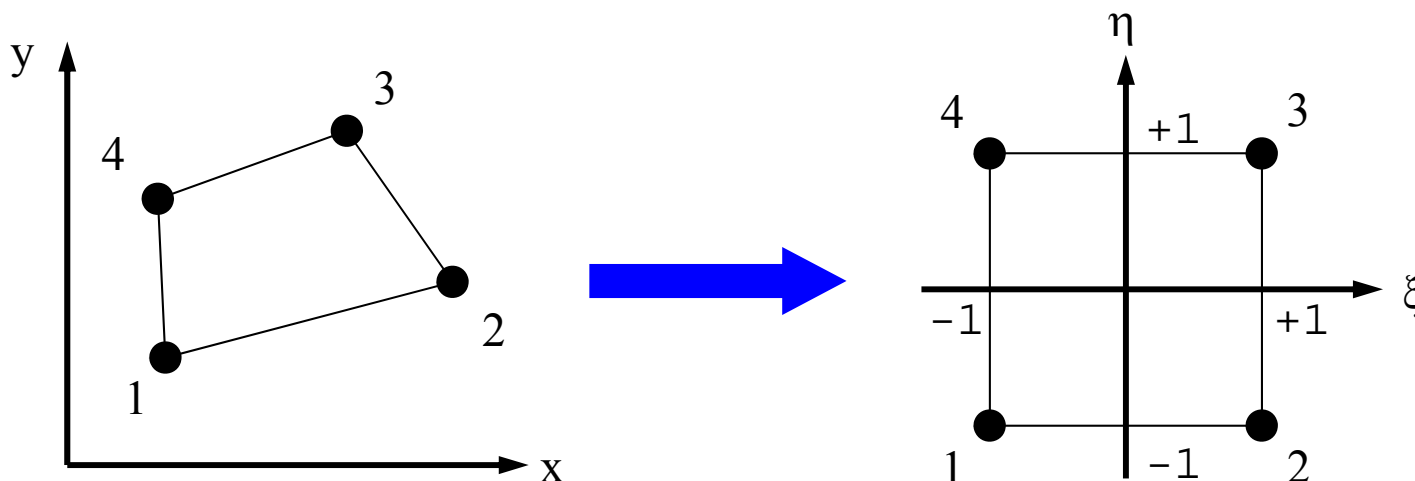
アイソパラメトリック要素 (1/3)

- 各要素を, 自然座標系 (ξ, η) の正方形要素 $[\pm 1, \pm 1]$ に変換する。



- 各要素の全体座標系 (global coordinate) (x, y) における座標成分を, 自然座標系における形状関数 $[N]$ (従属変数の内挿に使うのと同じ $[N]$) を使用して変換する場合, このような要素を**アイソパラメトリック要素 (isoparametric element)** という

アイソパラメトリック要素 (2/3)

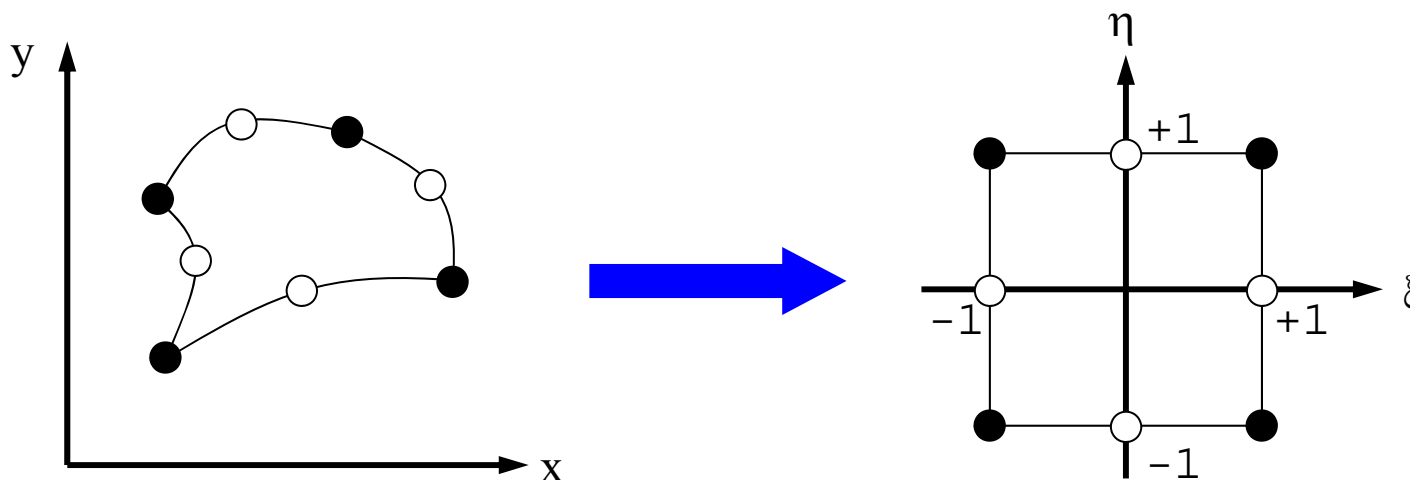


- 各節点の座標 : $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$
- 各節点における温度 : T_1, T_2, T_3, T_4

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

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

アイソパラメトリック要素 (3/3)



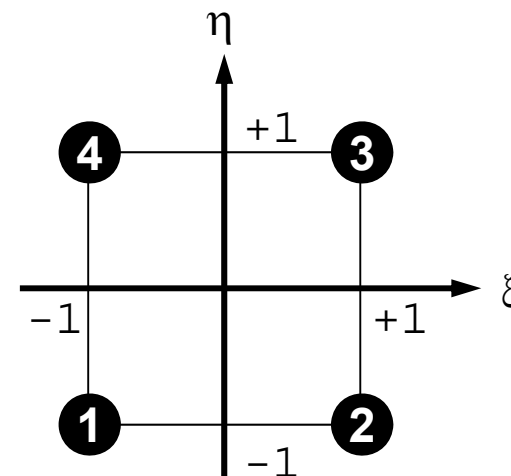
- 高次の補間関数を使えば，曲線，曲面も扱うことが可能となる。
- そういう意味で「自然座標系」と呼んでいる。

Sub-Parametric
Super-Parametric

2D自然座標系の形状関数 (1/3)

- 自然座標系における正方形上の内挿多項式は下式で与えられる：

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



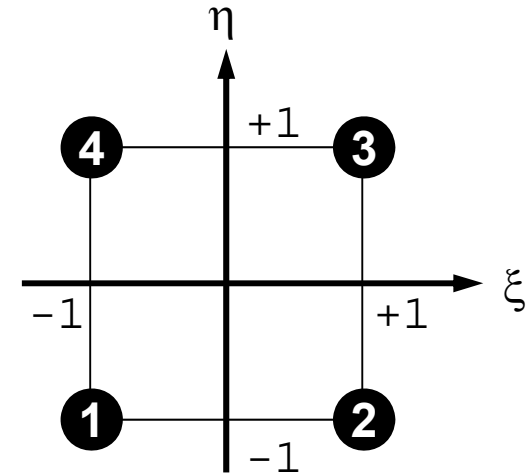
- 各節点での条件より：

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

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

2D自然座標系の形状関数 (2/3)

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



2D自然座標系の形状関数 (3/3)

- 元の式に代入して, T_i について整理すると以下のようなになる:

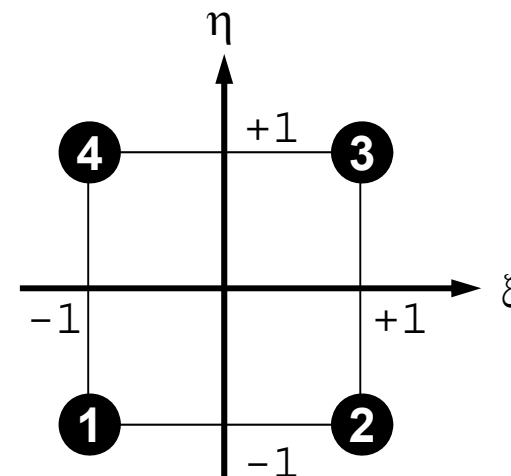
$$T = N_1T_1 + N_2T_2 + N_3T_3 + N_4T_4$$

- 形状関数 N_i は以下のようなになる:

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

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

- 双一次 (bi-linear) 要素とも呼ばれる。
- 各節点における N_i の値を計算してみよ



三次元への拡張

- 四面体要素：二次元における三角形要素
 - 任意の形状を扱うことができる。
 - 特に一次要素は精度が悪く，一部の問題を除いてあまり使用されない。
 - 高次の四面体要素は広く使用されている・・・
- 本講義では低次六面体要素（アイソパラメトリック要素）を使用する。
 - tri-linear
- 自由度：温度@各節点上

3D自然座標系の形状関数

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

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

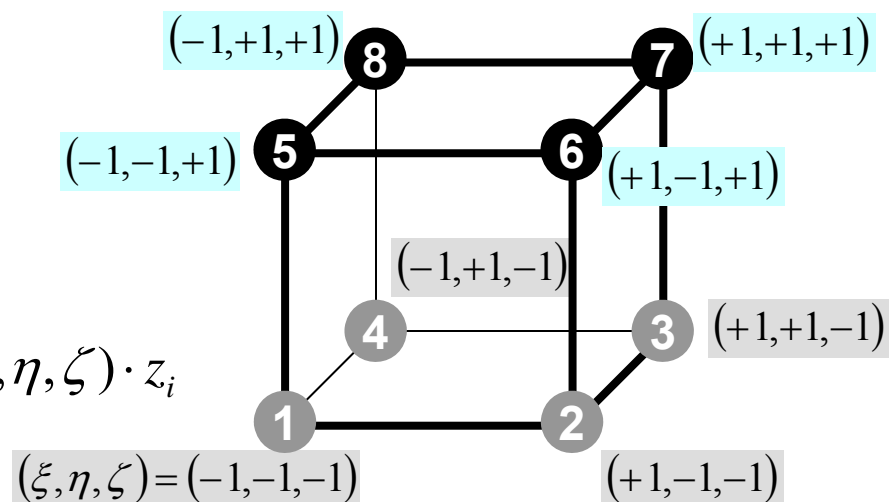
$$N_3(\xi, \eta, \zeta) = \frac{1}{8}(1+\xi)(1+\eta)(1-\zeta) \quad N_7(\xi, \eta, \zeta) = \frac{1}{8}(1+\xi)(1+\eta)(1+\zeta)$$

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

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

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

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



- 三次元要素の定式化
- 三次元熱伝導方程式
 - ガラーキン法
 - 要素マトリクス生成

- プログラムの実行
- データ構造
- プログラムの構成

ガラーキン法の適用 (1/3)

- 以下のような三次元熱伝導方程式を考慮する（熱伝導率一定）：

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

$T = [N]\{\phi\}$ 要素内の温度分布
(マトリクス形式), 節点における温度を ϕ としてある。

- ガラーキン法に従い, 重み関数を $[N]$ とすると, 各要素において以下の積分方程式が得られる:

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

ガラーキン法の適用 (2/3)

- 三次元のグリーンの定理

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

- 前式の2階微分の部分に適用 (表面積分省略) :

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

- これに以下を代入する :

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

ガラーキン法の適用 (3/3)

- 体積あたり発熱量の項 \dot{Q} を加えて次式が得られる :

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

- この式を弱形式 (weak form) と呼ぶ。元の微分方程式では2階の微分が含まれていたが、上式では、グリーンの定理によって1階微分に低減されている。
 - 弱形式によって近似関数 (形状関数, 内挿関数) に対する要求が弱くなっている : すなわち線形関数で2階微分の効果を記述できる。
 - 項が増えただけで、次元と同じ

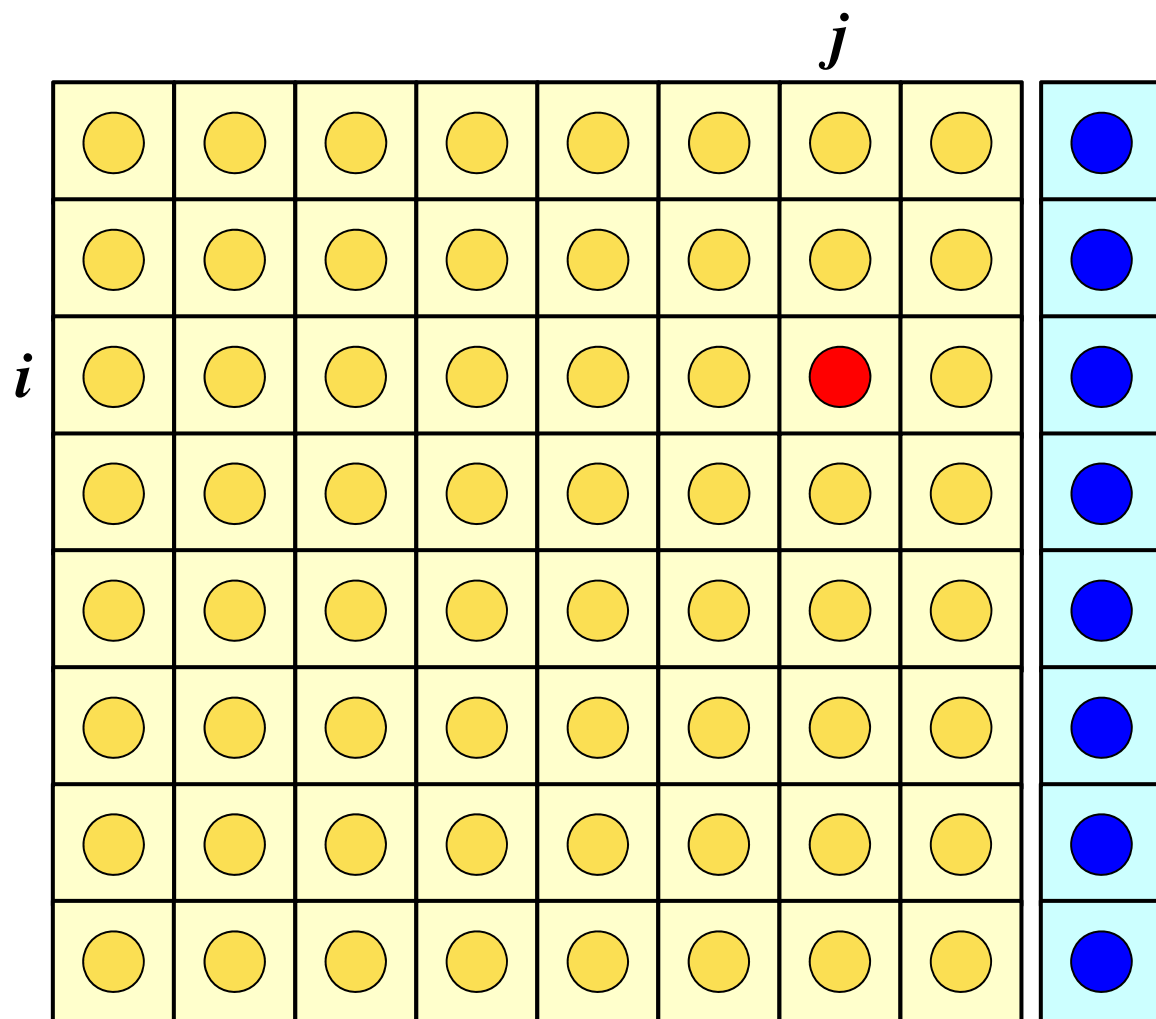
境界条件を考慮した弱形式：各要素

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

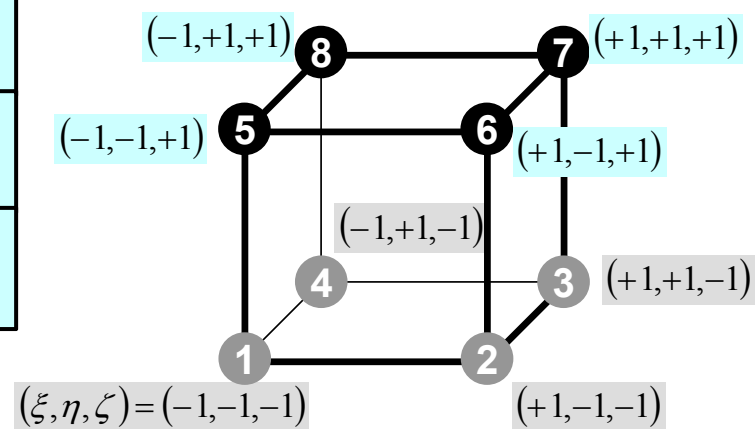
$$[k]^{(e)} = \int_V \lambda \left([N_{,x}]^T [N_{,x}] \right) dV + \int_V \lambda \left([N_{,y}]^T [N_{,y}] \right) dV \\ + \int_V \lambda \left([N_{,z}]^T [N_{,z}] \right) dV$$

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

要素マトリクス : 8×8 行列

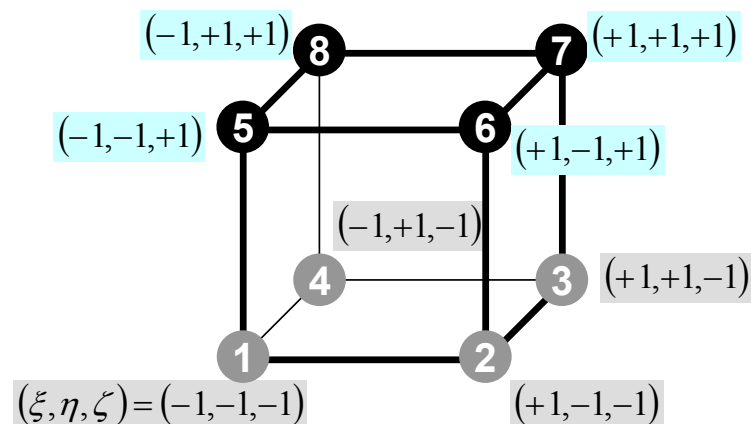


$$[k_{ij}] \quad (i, j = 1 \dots 8)$$



要素マトリクス： k_{ij}

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



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

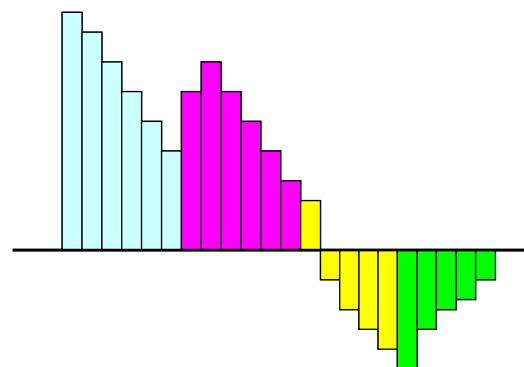


$$k_{ij} = - \int_V \{ \lambda \cdot N_{i,x} \cdot N_{j,x} + \lambda \cdot N_{i,y} \cdot N_{j,y} + \lambda \cdot N_{i,z} \cdot N_{j,z} \} dV$$

数値的に積分を実施する方法

- 台形公式
- シンプソンの公式
- ガウスの積分公式（ガウス＝ルジャンドル（Gauss-Legendre）とも呼ばれる，精度良い）
- 不定積分を解析的に求めるのではなく，有限な数のサンプル点における値を利用する

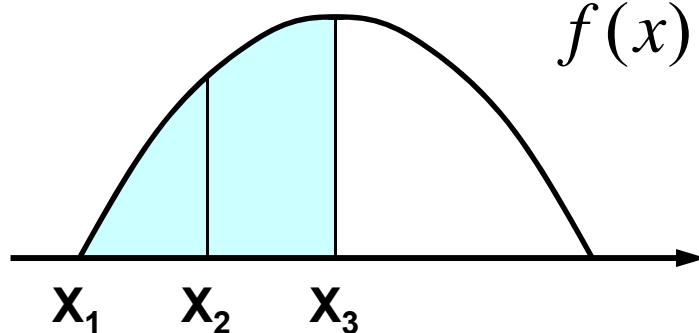
$$\int_{x_1}^{x_2} f(x) dx \Rightarrow \sum_{k=1}^m [w_k \cdot f(x_k)]$$



ガウス積分公式：一次元の例

シンプソンの公式より精度良い

Simpson's

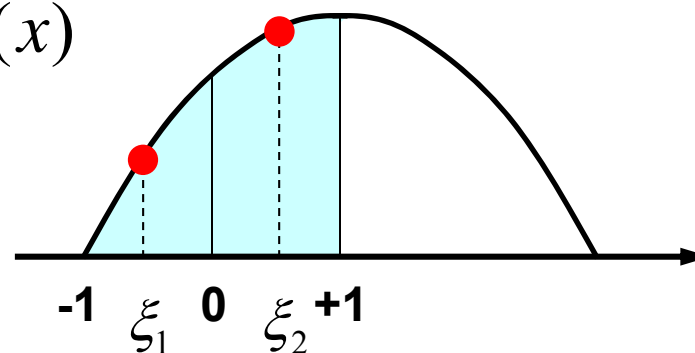


$$X_1 = 0, \quad X_2 = \frac{\pi}{4}, \quad X_3 = \frac{\pi}{2}$$

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

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

Gauss



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

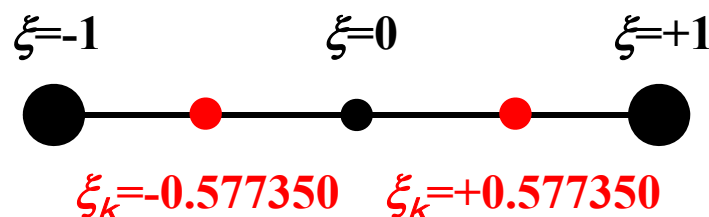
$$S = \int_0^{\pi/2} f(x) dx = \int_{-1}^{+1} f(\xi) h d\xi$$

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

ガウスの積分公式

- 無次元化された自然座標系 $[-1,+1]$ を対象とする。
- m 個の積分点を使用すると $(2m-1)$ 次の関数までは近似可能（従って1次, 2次の内挿関数（形状関数）を使用するときは, $m=2$ で十分）

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



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

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

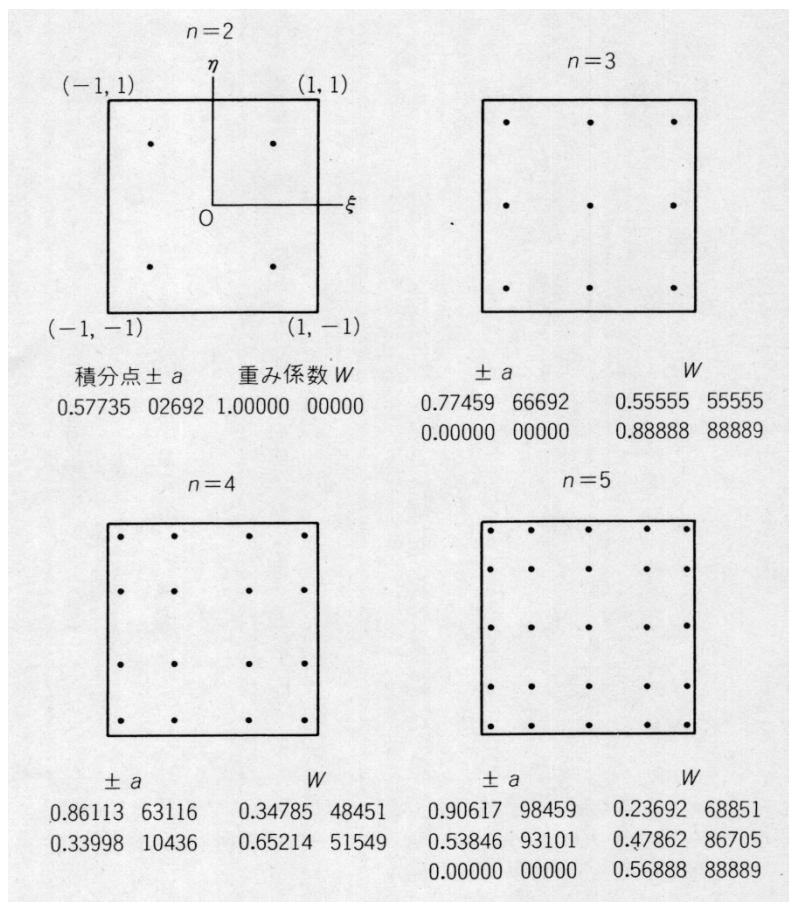
$$m = 3 \quad \xi_k = 0.00, w_k = 8/9$$

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

Gaussian Quadrature

ガウスの積分公式

- 自然座標系 (ξ, η, ζ) 上で定義 \Rightarrow ガウス積分公式が使える (三次元)



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

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

L, M, N : ξ, η, ζ 方向の積分点数

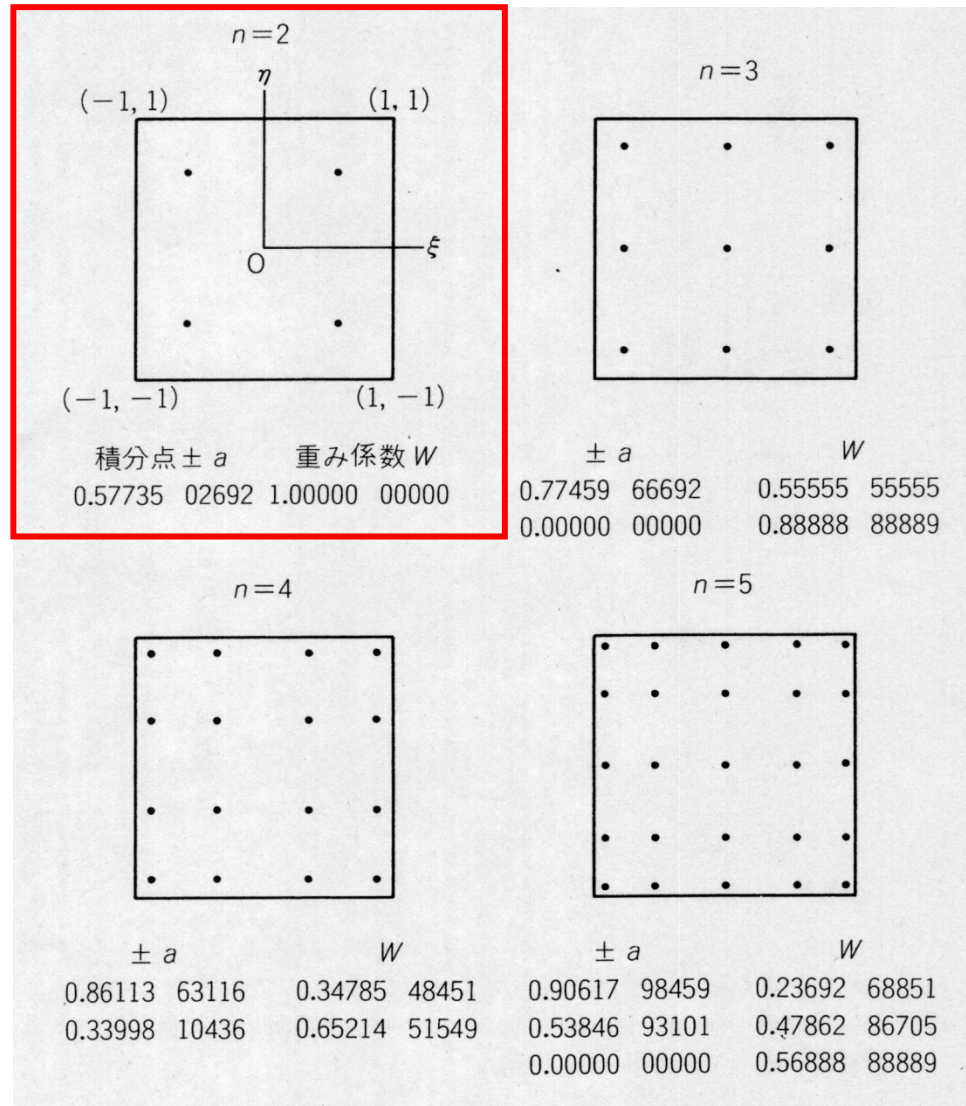
(ξ_i, η_j, ζ_k) : 積分点の座標値

W_i, W_j, W_k : 積分点での重み係数

Gaussian Quadrature

ガウスの積分公式

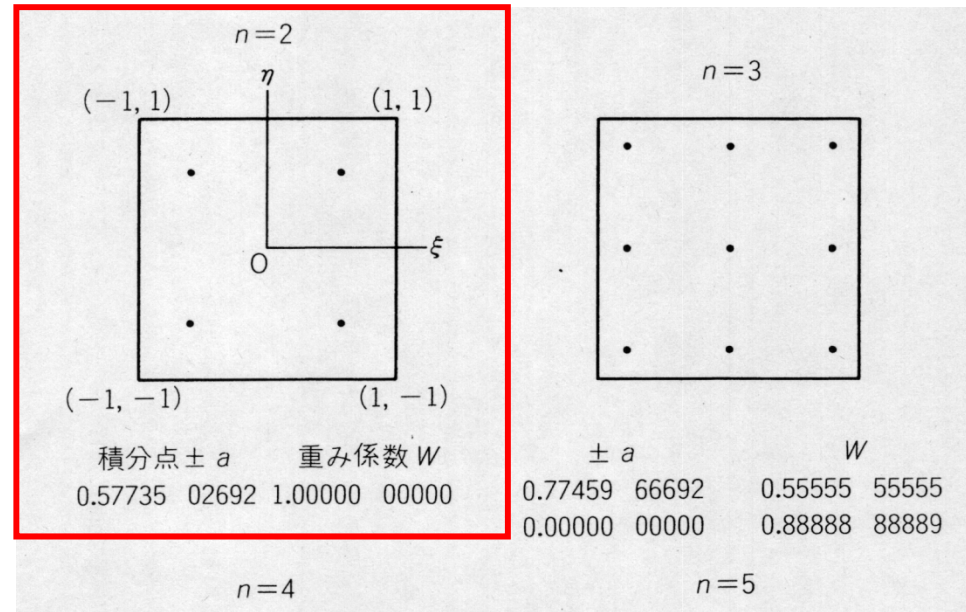
この組み合わせがよく使われる, 二次元では4点における f の値を計算して足せば良い(三次元では8点)



Gaussian Quadrature

ガウスの積分公式

この組み合わせがよく使われる, 二次元では4点におけるfの値を計算して足せば良い(三次元では8点)



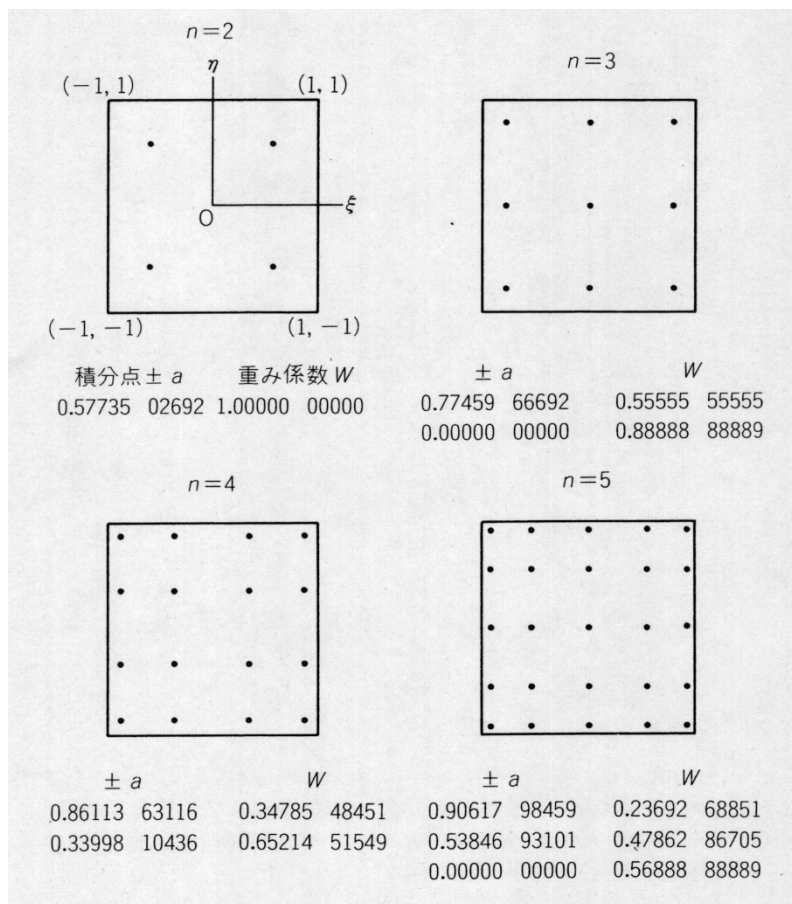
$$I = \int_{-1}^{+1} \int_{-1}^{+1} f(\xi, \eta) d\xi d\eta = \sum_{i=1}^m \sum_{j=1}^n [W_i \cdot W_j \cdot f(\xi_i, \eta_j)]$$

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

0.33998 0.10436 0.65214 0.51549 0.53846 0.35101 0.47802 0.6703
0.00000 0.00000 0.56888 0.88889

あとは積分を求めれば良い

- 自然座標系 (ξ, η, ζ) 上で定義 \Rightarrow ガウス積分公式が使える (三次元) . . . しかし, 微分が



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

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

L, M, N : ξ, η, ζ 方向の積分点数

(ξ_i, η_j, ζ_k) : 積分点の座標値

W_i, W_j, W_k : 積分点での重み係数

自然座標系における偏微分 (1/4)

- 偏微分の公式より以下のようなになる：

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

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

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

$\left[\frac{\partial N_i}{\partial \xi}, \frac{\partial N_i}{\partial \eta}, \frac{\partial N_i}{\partial \zeta} \right]$ は定義より簡単に求められるが

$\left[\frac{\partial N_i}{\partial x}, \frac{\partial N_i}{\partial y}, \frac{\partial N_i}{\partial z} \right]$ を実際の計算で使用する

自然座標系における偏微分 (2/4)

- マトリックス表示すると：

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

$[J]$: ヤコビのマトリクス
(Jacobi matrix
Jacobian)

自然座標系における偏微分 (3/4)

- N_i の定義より簡単に求められる

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

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

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

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

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

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

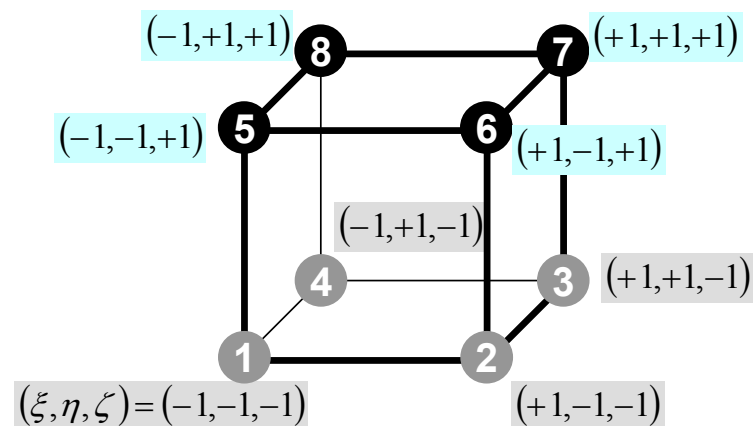
自然座標系における偏微分 (4/4)

- 従って下記のように偏微分を計算できる
 - ヤコビアン (3×3行列) の逆行列を求める

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

要素単位での積分

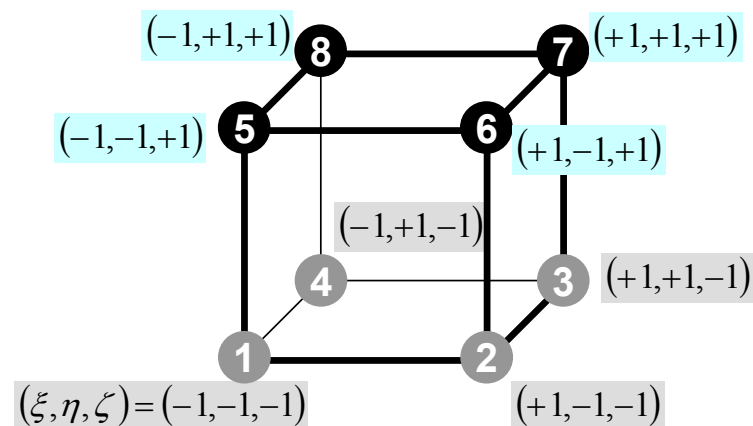
$$[k_{ij}] \quad (i, j = 1 \dots 8)$$



$$\begin{aligned}
 k_{ij} &= - \int_V \left\{ \lambda \cdot N_{i,x} \cdot N_{j,x} + \lambda \cdot N_{i,y} \cdot N_{j,y} + \lambda \cdot N_{i,z} \cdot N_{j,z} \right\} dV \\
 &= - \int_V \left\{ \lambda \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \lambda \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + \lambda \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right\} dV
 \end{aligned}$$

自然座標系での積分

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



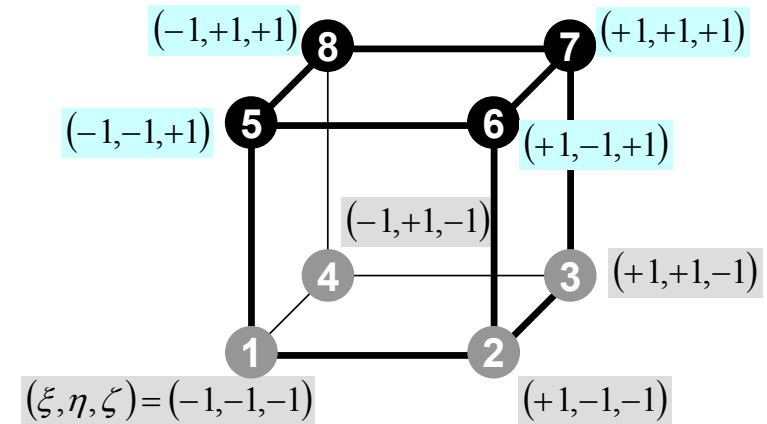
$$- \int_V \left\{ \lambda \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \lambda \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + \lambda \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right\} dV =$$

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

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

ガウスの積分公式

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



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

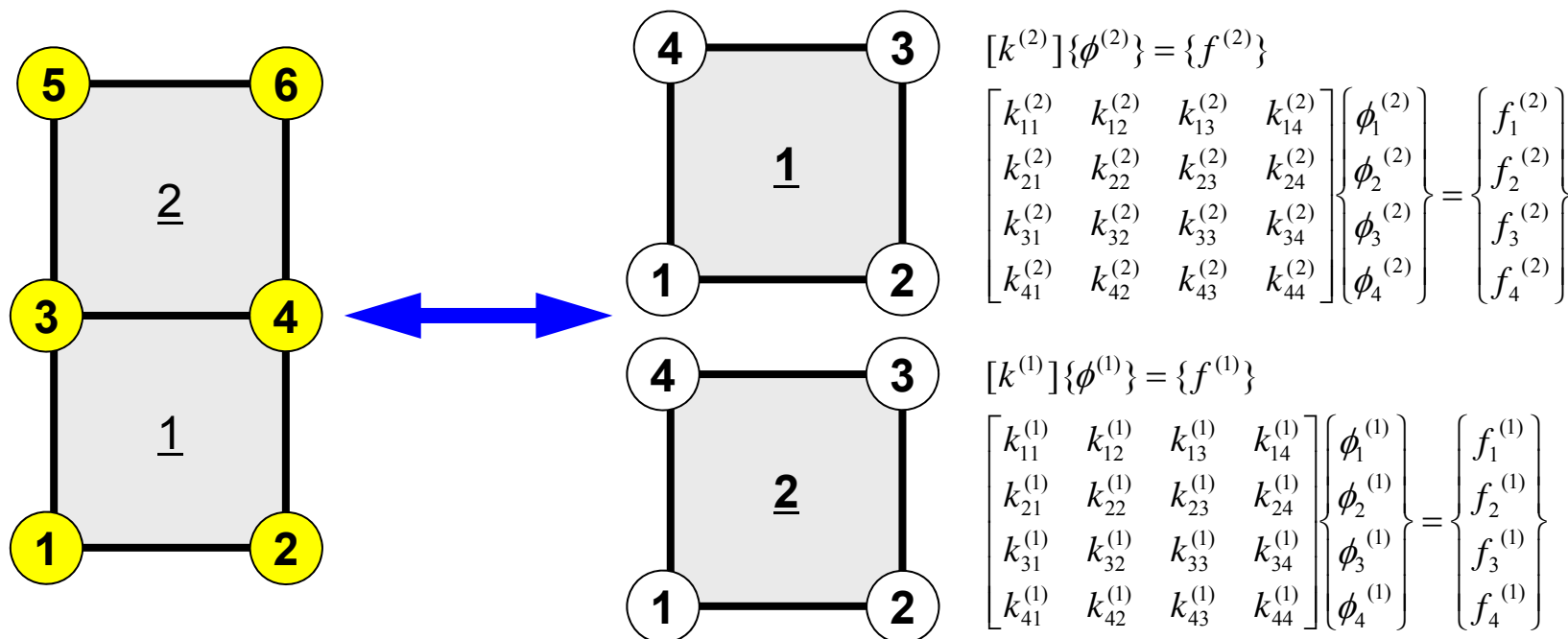
$$I = \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} f(\xi, \eta, \zeta) d\xi d\eta d\zeta$$

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

残りの手順

- ここまでで、要素ごとの積分が可能となる。
- あとは：
 - 全体マトリクスへの重ね合わせ
 - 境界条件処理
 - 連立一次方程式を解く . . .
- 詳細はプログラムの内容を解説しながら説明する。

要素 ⇒ 全体マトリクス重ね合わせ



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

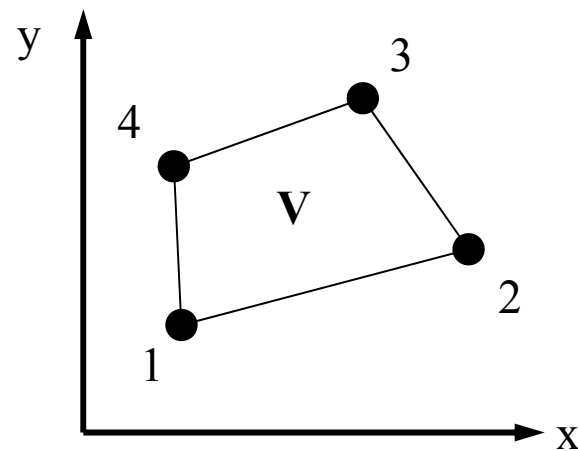
$$\begin{bmatrix} D_1 & X & X & X & & & \\ X & D_2 & X & X & & & \\ X & X & D_3 & X & X & X & \\ X & X & X & D_4 & X & X & \\ & & X & X & D_5 & X & \\ & & X & X & X & D_6 & \end{bmatrix} \begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \end{Bmatrix} = \begin{Bmatrix} B_1 \\ B_2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

- 三次元要素の定式化
- 三次元弾性力学方程式
 - ガラーキン法
 - 要素マトリクス生成
 - 演習

- プログラムの実行
- データ構造
- プログラムの構成

演習

- ガウスの積分公式を使用して以下の四角形の面積を求めよ（プログラムを作って、計算機で計算してください）



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

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

ヒント (1/2)

- 座標値によってヤコビアン（ヤコビの行列）を計算
- ガウスの積分公式（ $n=2$ ）に代入する。

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

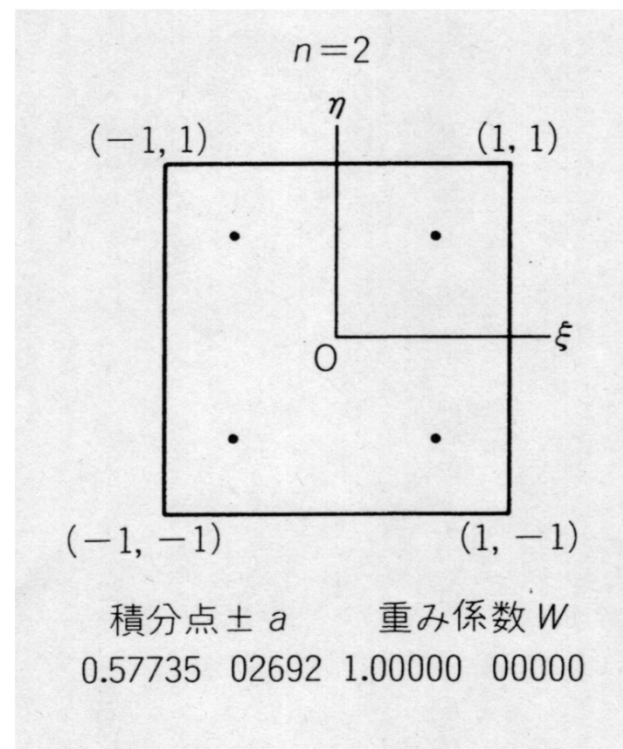
```

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

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

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

```



ヒント (2/2)

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

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

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

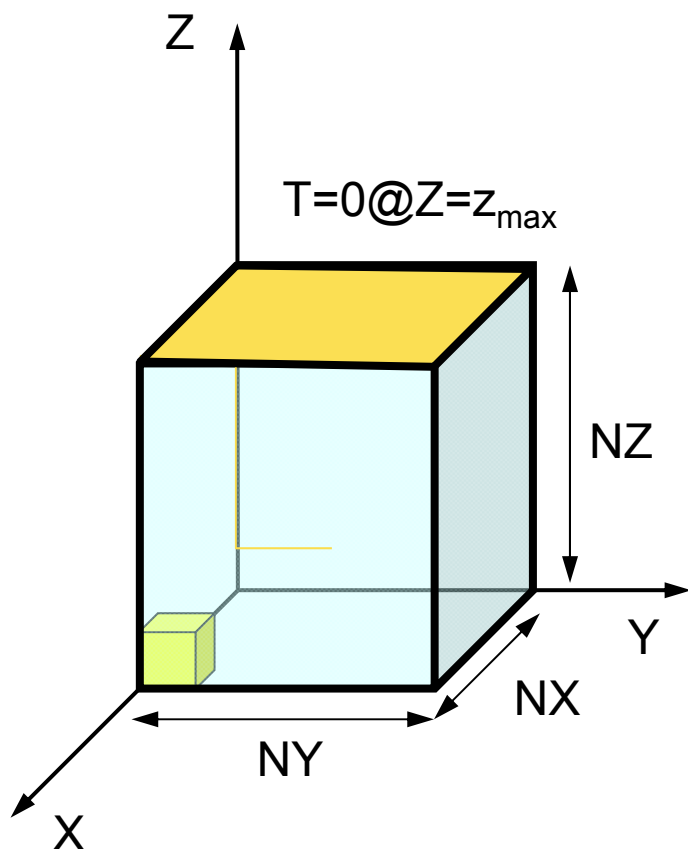
$$N_1(\xi, \eta) = \frac{1}{4}(1 - \xi)(1 - \eta), \quad N_2(\xi, \eta) = \frac{1}{4}(1 + \xi)(1 - \eta),$$

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

- 三次元要素の定式化
- 三次元弾性力学方程式
 - ガラーキン法
 - 要素マトリクス生成
- プログラムの実行
- データ構造
- プログラムの構成

対象とする問題：三次元定常熱伝導

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



- 定常熱伝導＋発熱
- 一様な熱伝導率 λ
- 直方体
 - 一辺長さ1の立方体（六面体）要素
 - 各方向に $NX \cdot NY \cdot NZ$ 個
- 境界条件
 - $T=0@Z=z_{\max}$
- 体積当たり発熱量は位置（メッシュの中心の座標 x_c, y_c ）に依存
 - $\dot{Q}(x, y, z) = QVOL|x_c + y_c|$

インストール (Cygwinでは*.exe)

インストール

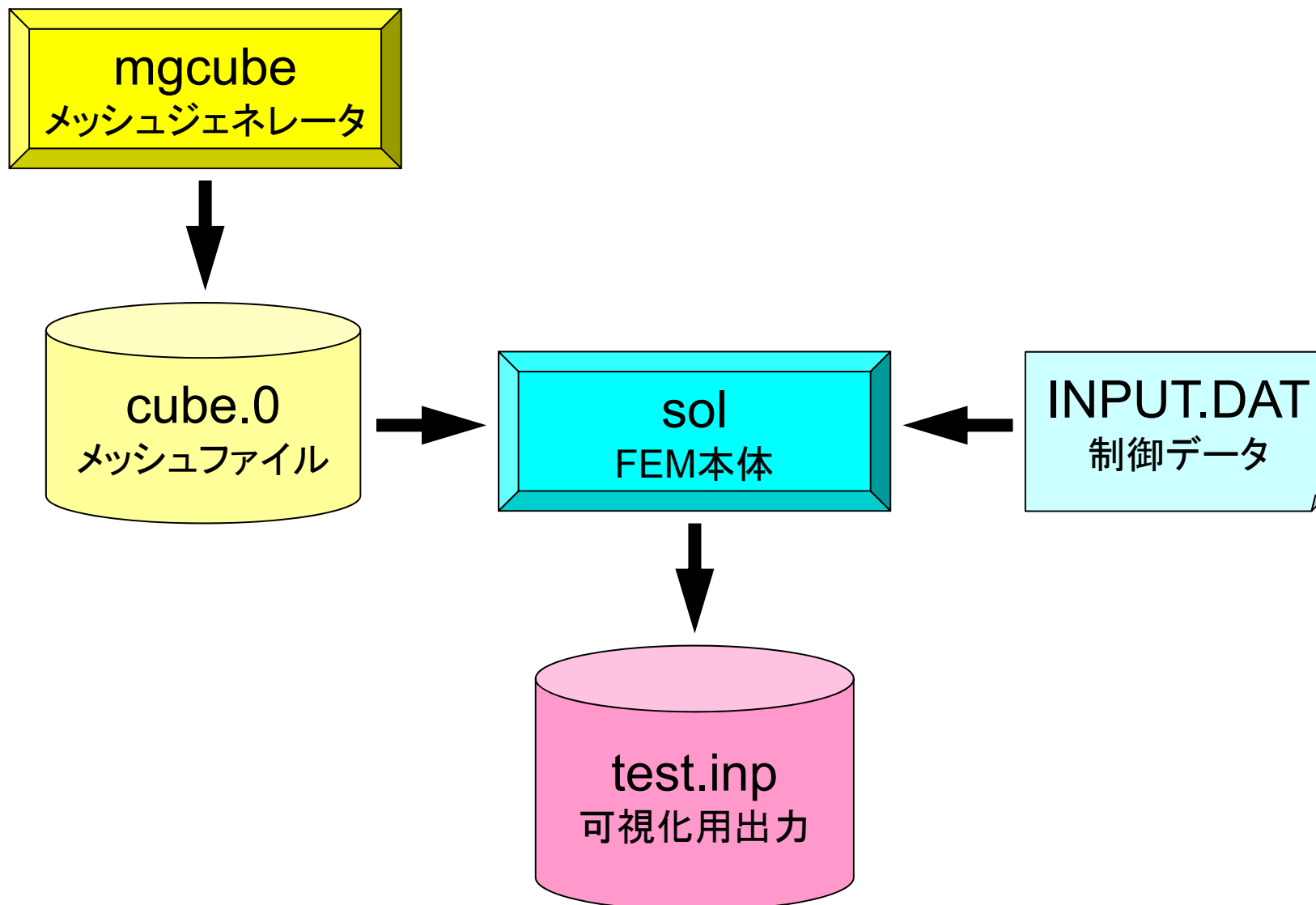
```
>$ cd <$P-TOP>/fem3d/src  
>$ make  
>$ ls ../run/sol  
sol
```

メッシュジェネレータインストール

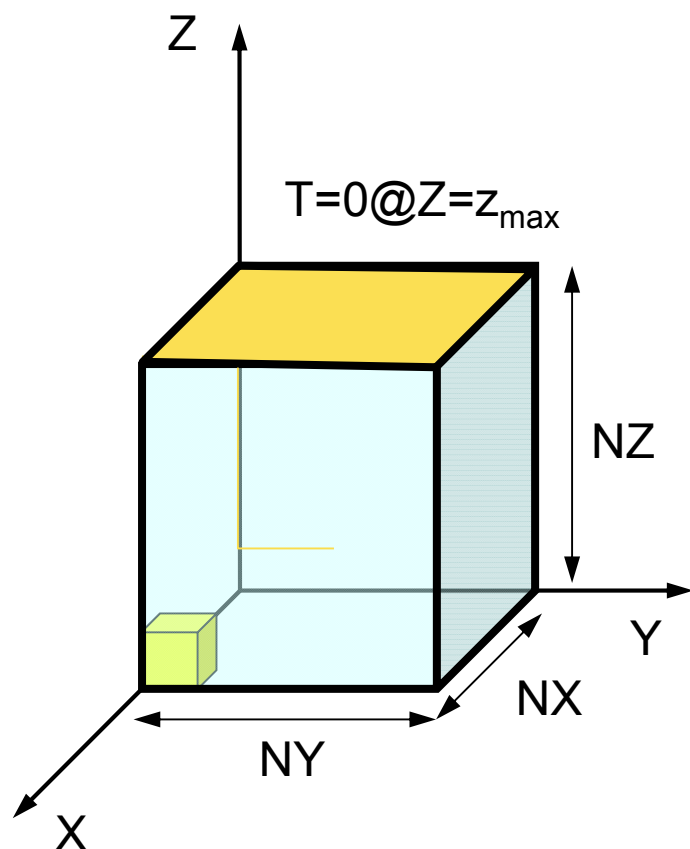
```
>$ cd <$P-TOP>/fem3d/run  
>$ cc -O3 mgcube.c -o mgcube
```

計算の流れ

メッシュ生成⇒計算, ファイル名称固定



メッシュ生成 (Cygwinでは mgcube.exe)



```
>$ cd <$P-TOP>/fem3d/run
```

```
>$ ./mgcube
```

```
    NX, NY, NZ
```

```
    20 20 20
```

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

```
    cube.0
```

← 各辺長さを
訊いてくる
← このように
入れてみる

生成を確認

制御ファイル：INPUT.DAT

INPUT.DAT

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

- fname : メッシュファイル名
- ITER : 反復回数上限
- COND : 熱伝導率
- QVOL : 体積当たり発熱量係数
- RESID : 反復法の収束判定値

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

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

実行 (Cygwinではsol.exe)

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

```
>$ ls test.inp          生成を確認  
test.inp
```

ParaView

- <http://www.paraview.org/>
- ファイルを開く
- 図の表示
- イメージファイルの保存
- <http://nkl.cc.u-tokyo.ac.jp/class/HowtouseParaView.pdf>

UCD Format (1/3)

Unstructured Cell Data

要素の種類

点

線

三角形

四角形

四面体

角錐

三角柱

六面体

二次要素

線2

三角形2

四角形2

四面体2

角錐2

三角柱2

六面体2

キーワード

pt

line

tri

quad

tet

pyr

prism

hex

line2

tri2

quad2

tet2

pyr2

prism2

hex2

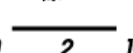
点



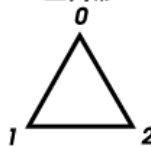
線



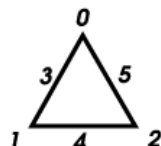
線2



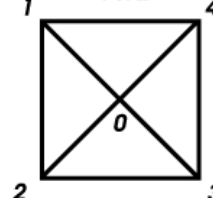
三角形



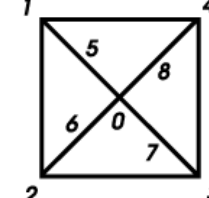
三角形2



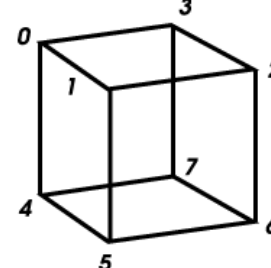
四角錐



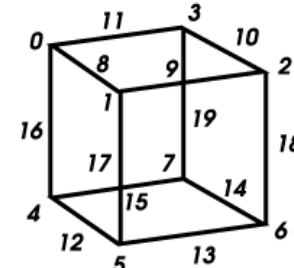
四角錐2



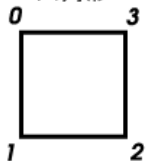
六面体



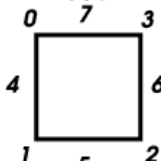
六面体2



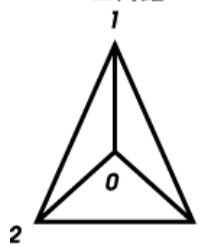
四角形



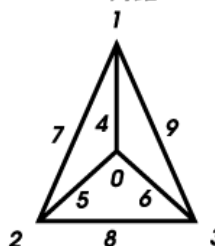
四角形2



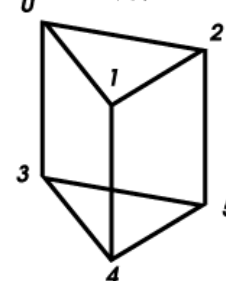
三角錐



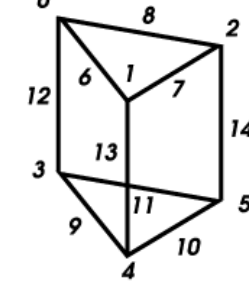
三角錐2



三角柱



三角柱2



UCD Format (2/3)

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

UCD Format (3/3): Old Format

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

(節点番号1) (X座標) (Y座標) (Z座標)

(節点番号2) (X座標) (Y座標) (Z座標)

·
·
·

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

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

·
·
·

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

(節点データ成分1のラベル), (単位)

(節点データ成分2のラベル), (単位)

·
·
·

(各節点データ成分のラベル), (単位)

(節点番号1) (節点データ1) (節点データ2)

(節点番号2) (節点データ1) (節点データ2)

·
·
·

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

(要素データ成分1のラベル), (単位)

(要素データ成分2のラベル), (単位)

·
·
·

(各要素データ成分のラベル), (単位)

(要素番号1) (要素データ1) (要素データ2)

(要素番号2) (要素データ1) (要素データ2)

·
·
·

- 三次元要素の定式化
- 三次元弾性力学方程式
 - ガラーキン法
 - 要素マトリクス生成

- プログラムの実行
- データ構造
- プログラムの構成

メッシュファイル構成：cube.0

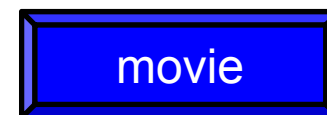
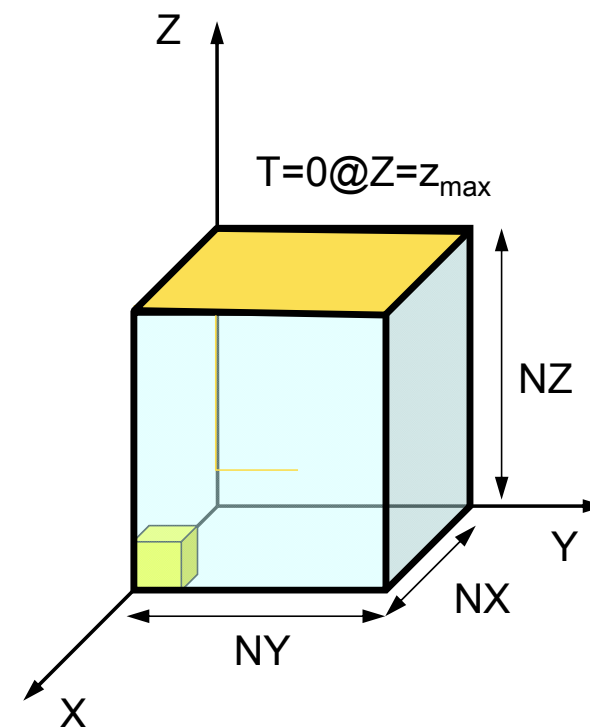
番号は「1」から始まっている

- 節点データ
 - 節点数
 - 節点番号, 座標
- 要素データ
 - 要素数
 - 要素タイプ
 - 要素番号, 材料番号, コネクティビティ
- 節点グループデータ
 - グループ数
 - グループ内節点数
 - グループ名
 - グループ内節点

cube.0 : 節点データ (NX=NY=NZ=4)

節点ID	X座標	Y座標	Z座標
125			
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

=5*5*5 (節点数)

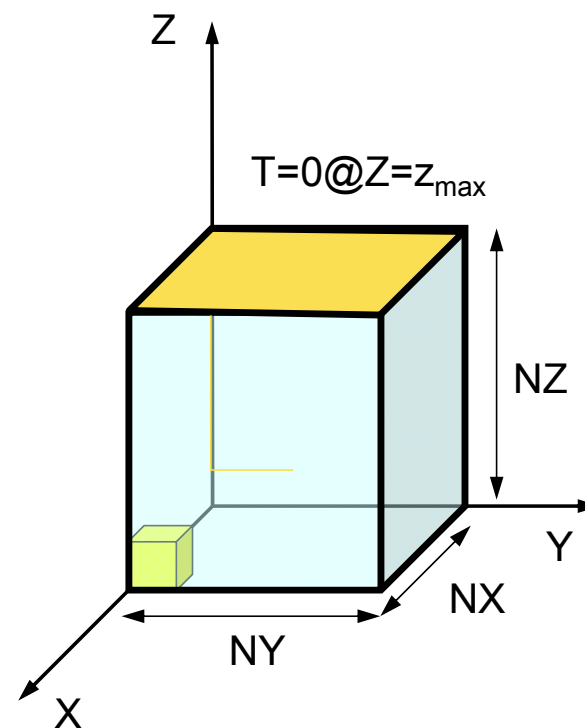


cube.0 : 要素データ (1/2)

64 =4*4*4 (要素数)

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	361	361	361	361	361	361
361	361	361	361	361	361	361	361	361	361

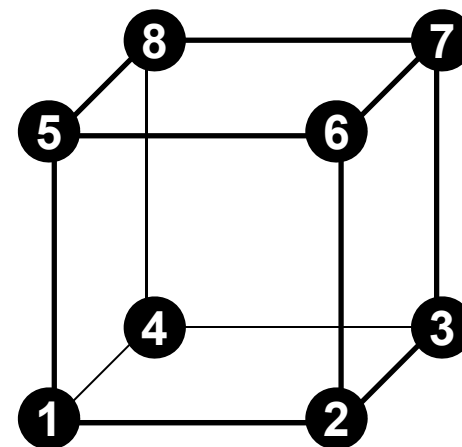
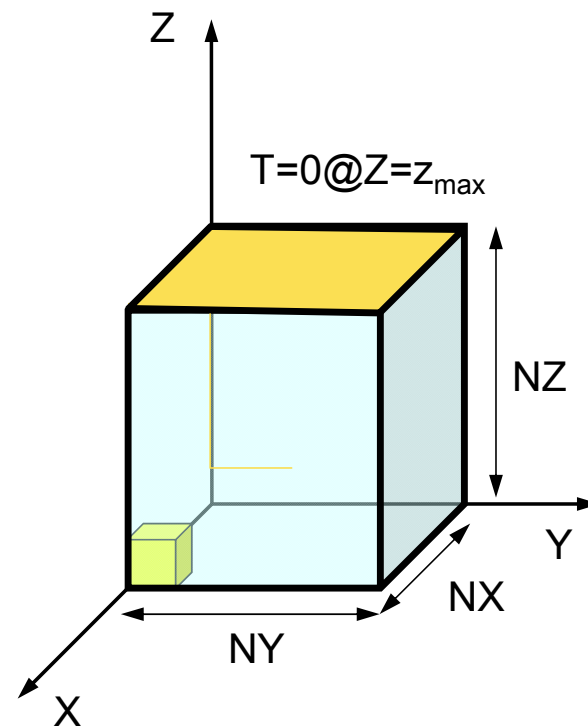
要素タイプ : 361
 三次元, 六面体, 一次



cube.0 : 要素データ (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

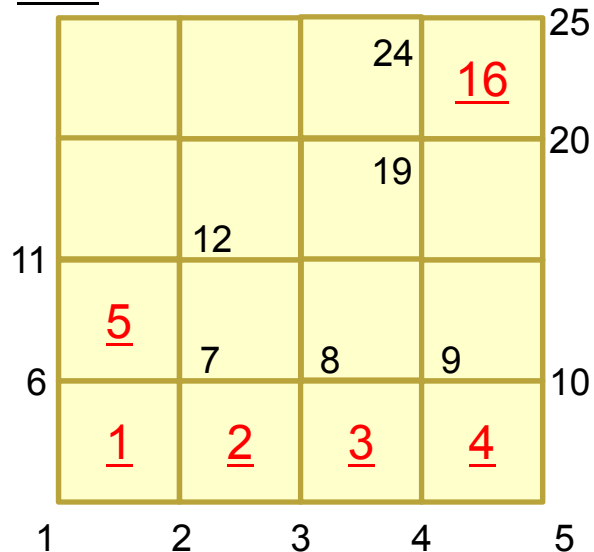
要素ID 材料ID 8節点ID



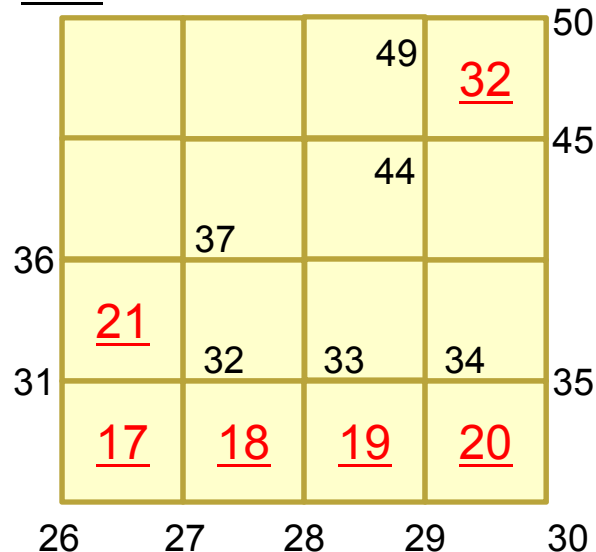
$NX=NY=NZ=4$, $NXP1=NYP1=NZP1=5$

$ICELTOT=64$, $INODTOT=125$, $IBNODTOT=25$

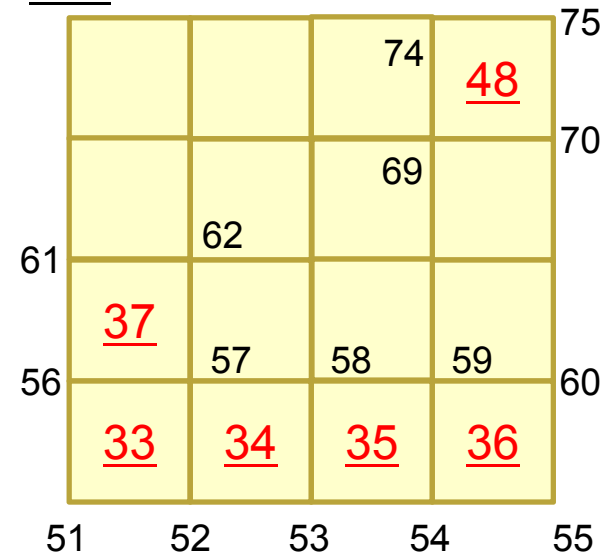
k=1



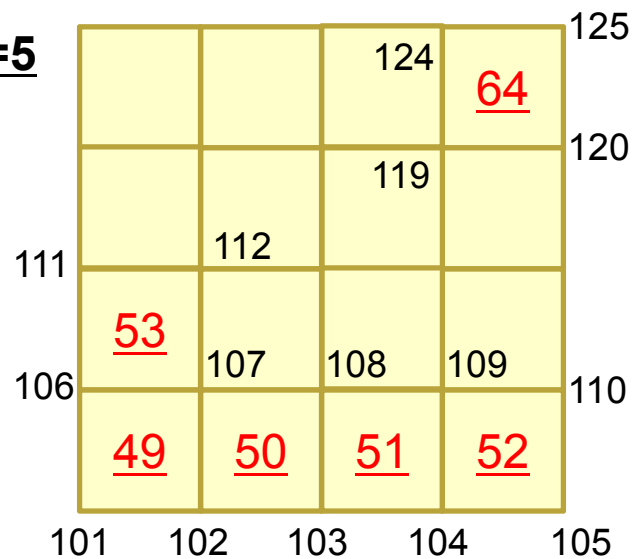
k=2



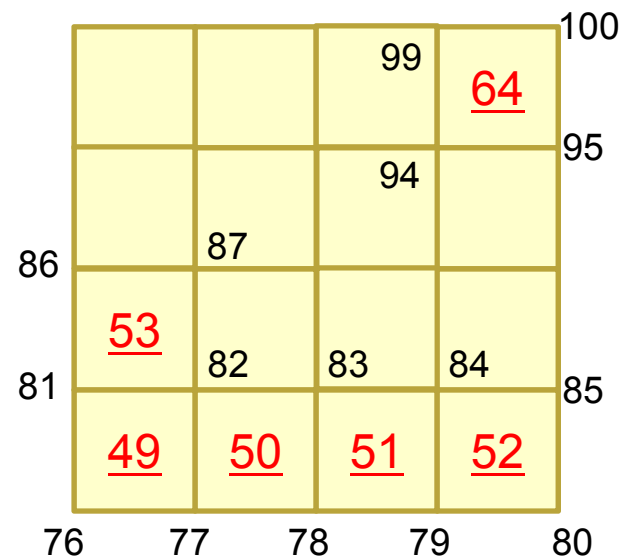
k=3



k=5



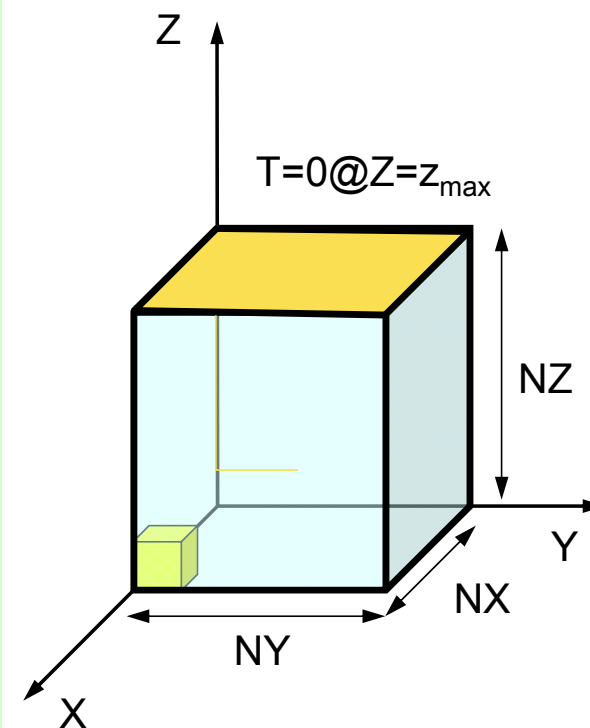
k=4



cube.0 : 節点グループデータ

	4	25	50	75	100	グループ総数 各グループ節点数 (累積)				
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					

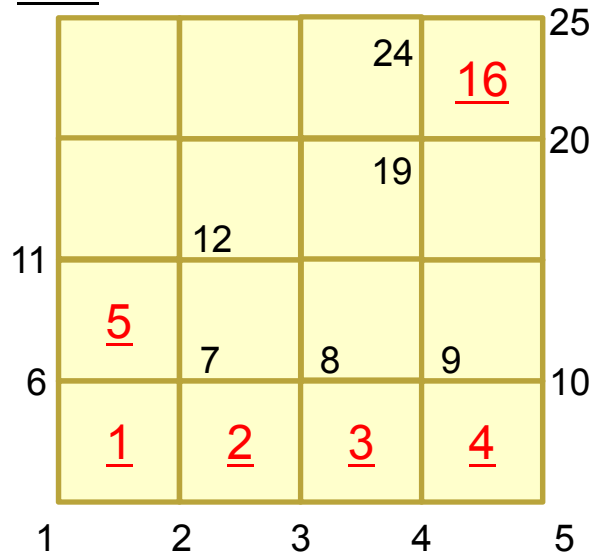
(以下 使用せず)



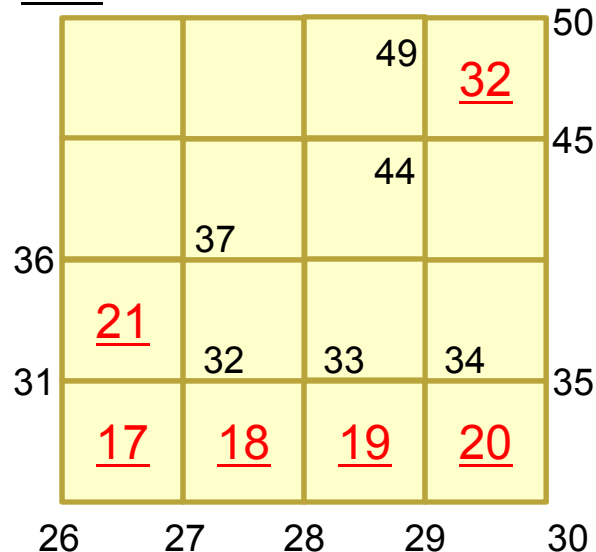
$NX=NY=NZ=4, NXP1=NYP1=NZP1=5$

$ICELTOT= 64, INODTOT= 125, IBNODTOT= 25$

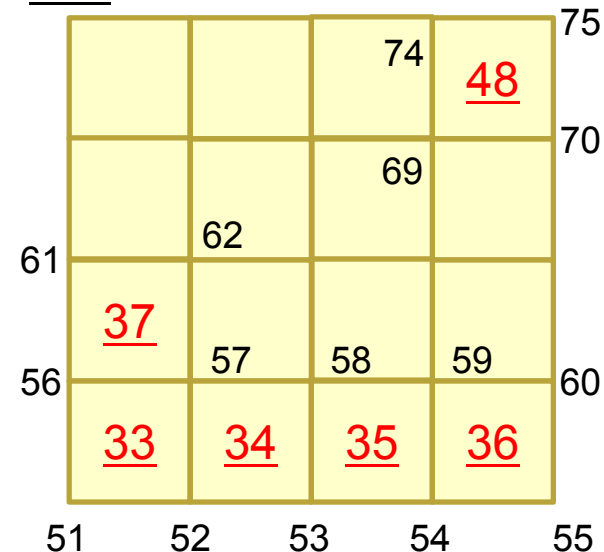
k=1



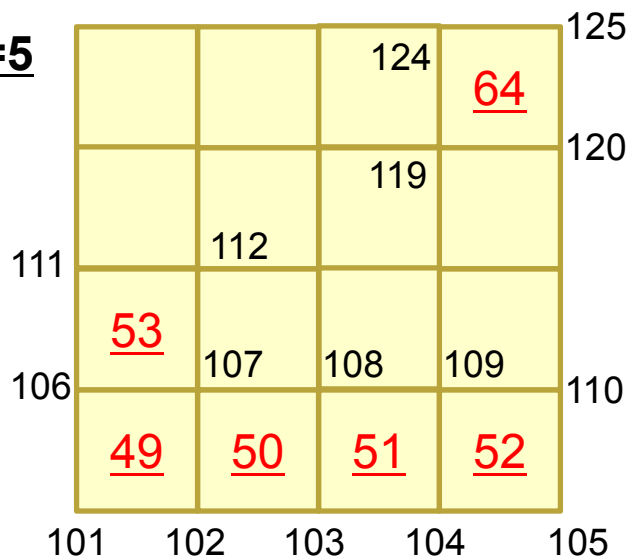
k=2



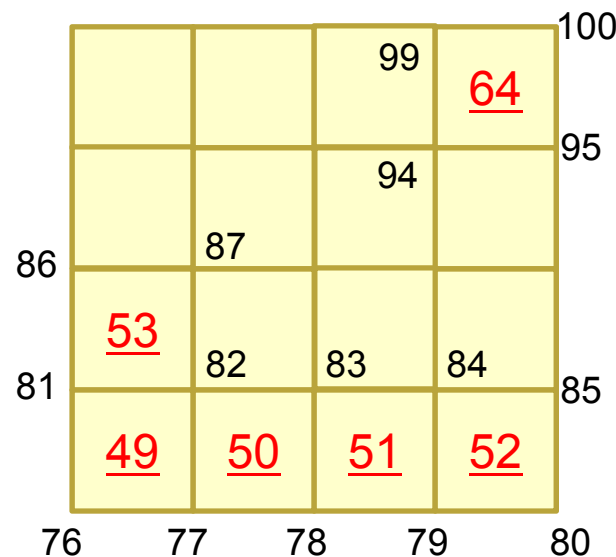
k=3



k=5



k=4



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

メッシュ生成

- 実は技術的には大きな課題
 - 複雑形状
 - 大規模メッシュ
- 並列化が難しい
- 市販のメッシュ生成アプリケーション
 - FEMAP
 - CADデータとのインタフェース



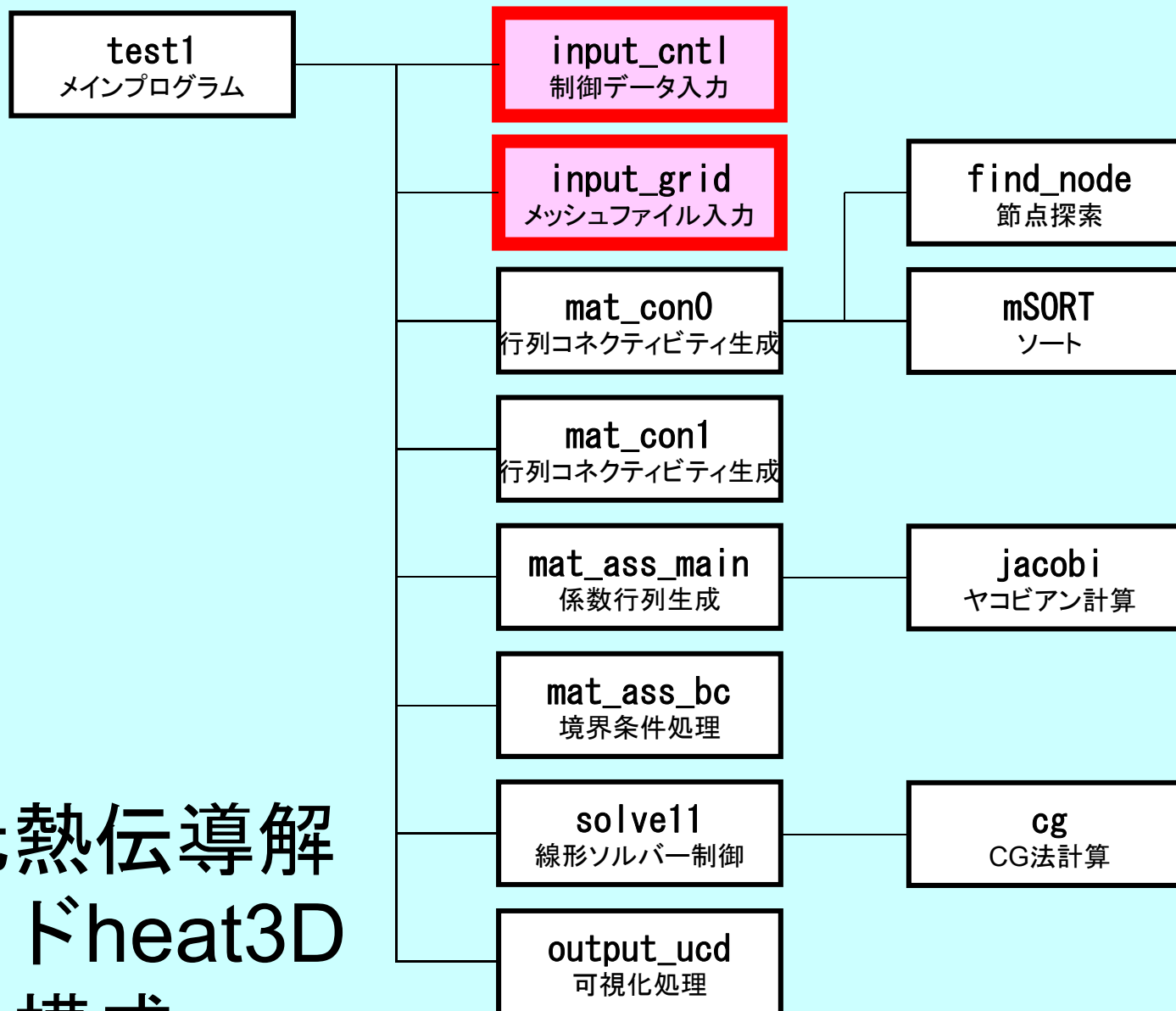
movie

- 三次元要素の定式化
- 三次元弾性力学方程式
 - ガラーキン法
 - 要素マトリクス生成

- プログラムの実行
- データ構造
- プログラムの構成

有限要素法の処理：プログラム

- 初期化
 - 制御変数読み込み
 - 座標読み込み⇒要素生成 (N:節点数, NE:要素数)
 - 配列初期化 (全体マトリクス, 要素マトリクス)
 - 要素⇒全体マトリクスマッピング (Index, Item)
- マトリクス生成
 - 要素単位の処理 (do icel= 1, NE)
 - 要素マトリクス計算
 - 全体マトリクスへの重ね合わせ
 - 境界条件の処理
- 連立一次方程式
 - 共役勾配法 (CG)



三次元熱伝導解析コードheat3Dの構成

全体処理

```
/**
    program heat3D
**/
#include <stdio.h>
#include <stdlib.h>
FILE* fp_log;
#define GLOBAL_VALUE_DEFINE
#include "pfem_util.h"
// #include "solver11.h"
extern void INPUT_CNTL ();
extern void INPUT_GRID ();
extern void MAT_CONO ();
extern void MAT_CON1 ();
extern void MAT_ASS_MAIN ();
extern void MAT_ASS_BC ();
extern void SOLVE11 ();
extern void OUTPUT_UCD ();
int main()
{
    INPUT_CNTL ();
    INPUT_GRID ();

    MAT_CONO ();
    MAT_CON1 ();

    MAT_ASS_MAIN ();
    MAT_ASS_BC ();

    SOLVE11 ();

    OUTPUT_UCD ();
}
```

Global変数表 : pfem_util.h (1/3)

変数名	種別	サイズ	I/O	内 容
fname	C	[80]	I	メッシュファイル名
N, NP	I		I	節点数
ICELTOT	I		I	要素数
NODGRPtot	I		I	節点グループ数
XYZ	R	[N][3]	I	節点座標
ICELNOD	I	[ICELTOT][8]	I	要素コネクティビティ
NODGRP_INDEX	I	[NODGRPtot+1]	I	各節点グループに含まれる節点数 (累積)
NODGRP_ITEM	I	[NODGRP_INDEX[NODGRPtot+1]]	I	節点グループに含まれる節点
NODGRP_NAME	C80	[NODGRP_INDEX[NODGRPtot+1]]	I	節点グループ名
NLU	I		O	各節点非対角成分数
NPLU	I		O	非対角成分総数
D	R	[N]	O	全体行列 : 対角ブロック
B, X	R	[N]	O	右辺ベクトル, 未知数ベクトル

Global変数表 : pfem_util.h (2/3)

変数名	種別	サイズ	I/O	内 容
AMAT	R	[N]	O	全体行列 : 非零非対角成分
indexLU	I	[N+1]	O	全体行列 : 非零非対角成分数
itemLU	I	[NPLU]	O	全体行列 : 非零非対角成分 (列番号)
INLU	I	[N]	O	各節点の非零非対角成分数
IALU	I	[N] [NLU]	O	各節点の非零非対角成分数 (列番号)
IWKX	I	[N] [2]	O	ワーク用配列
ITER, ITERactual	I		I	反復回数の上限, 実際の反復回数
RESID	R		I	打ち切り誤差 (1.e-8に設定)
pfemIarray	I	[100]	O	諸定数 (整数)
pfemRarray	R	[100]	O	諸定数 (実数)

Global変数表 : pfem_util.h (3/3)

変数名	種別	サイズ	I/O	内 容
08th	R		I	=0.125
PNQ, PNE, PNT	R	[2][2][8]	O	各ガウス積分点における $\frac{\partial N_i}{\partial \xi}, \frac{\partial N_i}{\partial \eta}, \frac{\partial N_i}{\partial \zeta} (i=1\sim 8)$
POS, WEI	R	[2]	O	各ガウス積分点の座標, 重み係数
NCOL1, NCOL2	I	[100]	O	ソート用ワーク配列
SHAPE	R	[2][2][2][8]	O	各ガウス積分点における形状関数 $N_i (i=1\sim 8)$
PNX, PNY, PNZ	R	[2][2][2][8]	O	各ガウス積分点における $\frac{\partial N_i}{\partial x}, \frac{\partial N_i}{\partial y}, \frac{\partial N_i}{\partial z} (i=1\sim 8)$
DETJ	R	[2][2][2]	O	各ガウス積分点におけるヤコビアン行列式
COND, QVOL	R		I	熱伝導率, 体積当たり発熱量係数

制御ファイル入力 : INPUT_CNTL

```
/**
** INPUT_CNTL
**/
#include <stdio.h>
#include <stdlib.h>
#include "pfem_util.h"
/** **/
void INPUT_CNTL ()
{
    FILE *fp;

    if( (fp=fopen("INPUT.DAT","r")) == NULL) {
        fprintf(stdout,"input file cannot be opened!\n");
        exit(1);
    }
    fscanf(fp,"%s",fname);
    fscanf(fp,"%d",&ITER);
    fscanf(fp,"%lf %lf", &COND, &QVOL);
    fscanf(fp,"%lf", &RESID);
    fclose(fp);

    pfemRarray[0]= RESID;
    pfemIarray[0]= ITER;
}
```

INPUT.DAT

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

メッシュ入力 : INPUT_GRID (1/3)

```
#include <stdio.h>
#include <stdlib.h>
#include "pfem_util.h"
#include "allocate.h"
void INPUT_GRID()
{
    FILE *fp;
    int i, j, k, ii, kk, nn, icel, iS, iE;
    int NTYPE, IMAT;

    if( (fp=fopen(fname, "r")) == NULL) {
        fprintf(stdout, "input file cannot be opened!¥n");
        exit(1);
    }

    /**
    NODE
    **/
    fscanf(fp, "%d", &N);

    NP=N;
    XYZ=(KREAL**) allocate_matrix(sizeof(KREAL), N, 3);

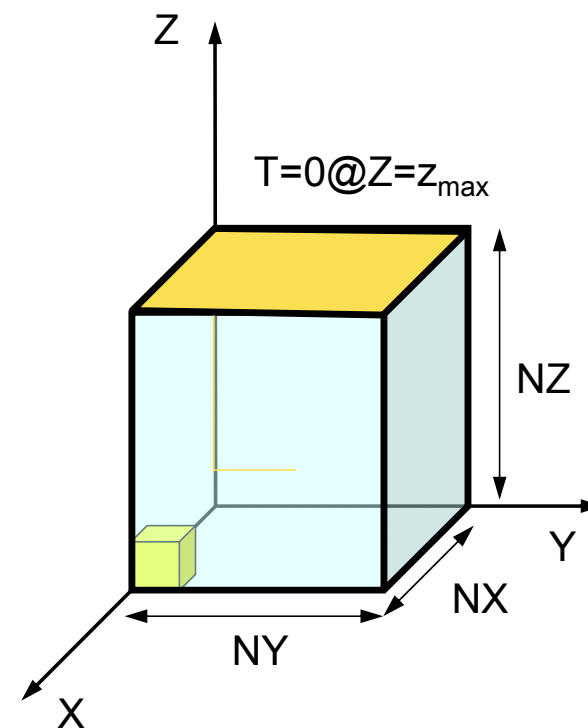
    for(i=0; i<N; i++) {
        for(j=0; j<3; j++) {
            XYZ[i][j]=0.0;
        }
    }

    for(i=0; i<N; i++) {
        fscanf(fp, "%d %lf %lf %lf", &ii, &XYZ[i][0], &XYZ[i][1], &XYZ[i][2]);
    }
}
```


cube.0 : 節点データ (NX=NY=NZ=4)

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関数

```
#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 );
}
```

allocateをFORTRAN並みに
簡単にやるための関数

メッシュ入力 : INPUT_GRID (2/3)

```
/**  
ELEMENT  
**/  
fscanf(fp, "%d", &ICELTOT);  
  
ICELNOD=(KINT**) allocate_matrix(sizeof(KINT), ICELTOT, 8);  
for (i=0; i<ICELTOT; i++) fscanf(fp, "%d", &NTYPE);  
  
for (icel=0; icel<ICELTOT; icel++) {  
    fscanf(fp, "%d %d %d %d %d %d %d %d %d %d", &i, &IMAT,  
           &ICELNOD[icel][0], &ICELNOD[icel][1], &ICELNOD[icel][2], &ICELNOD[icel][3],  
           &ICELNOD[icel][4], &ICELNOD[icel][5], &ICELNOD[icel][6], &ICELNOD[icel][7]);  
}
```

ICELNOD[i][j]の中身としては「1」から始まる通し節点番号がそのまま読み込まれている。要素番号は「0」から番号付け。

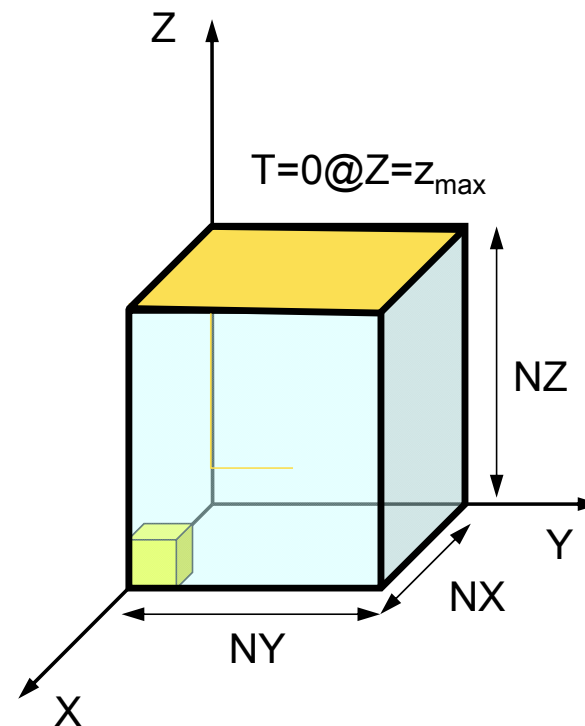
cube.0 : 要素データ (1/2)

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	361	361	361	361	361	361
361	361	361	361	361	361	361	361	361	361

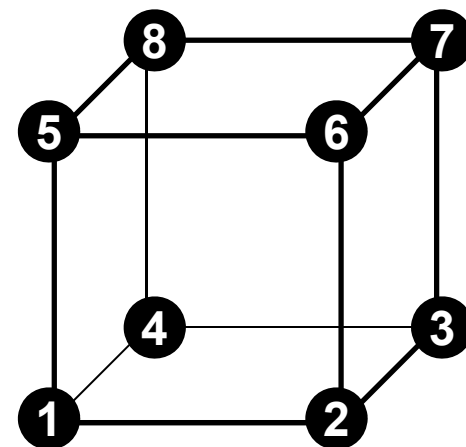
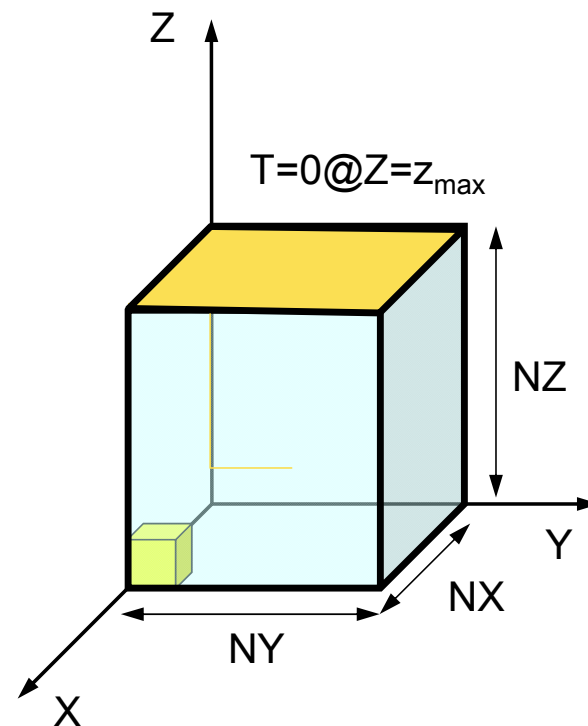
要素タイプ : 361
 三次元, 六面体, 一次



cube.0 : 要素データ (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]



メッシュ入力 : INPUT_GRID (3/3)

```
/**
  NODE grp. info.
**/
fscanf(fp, "%d", &NODGRPtot);

NODGRP_INDEX=(KINT* ) allocate_vector (sizeof(KINT), NODGRPtot+1);
NODGRP_NAME =(CHAR80*) allocate_vector (sizeof(CHAR80), NODGRPtot);
for(i=0;i<NODGRPtot+1;i++) NODGRP_INDEX[i]=0;

for(i=0;i<NODGRPtot;i++) fscanf(fp, "%d", &NODGRP_INDEX[i+1]);
nn=NODGRP_INDEX[NODGRPtot];
NODGRP_ITEM=(KINT*) allocate_vector (sizeof(KINT), nn);

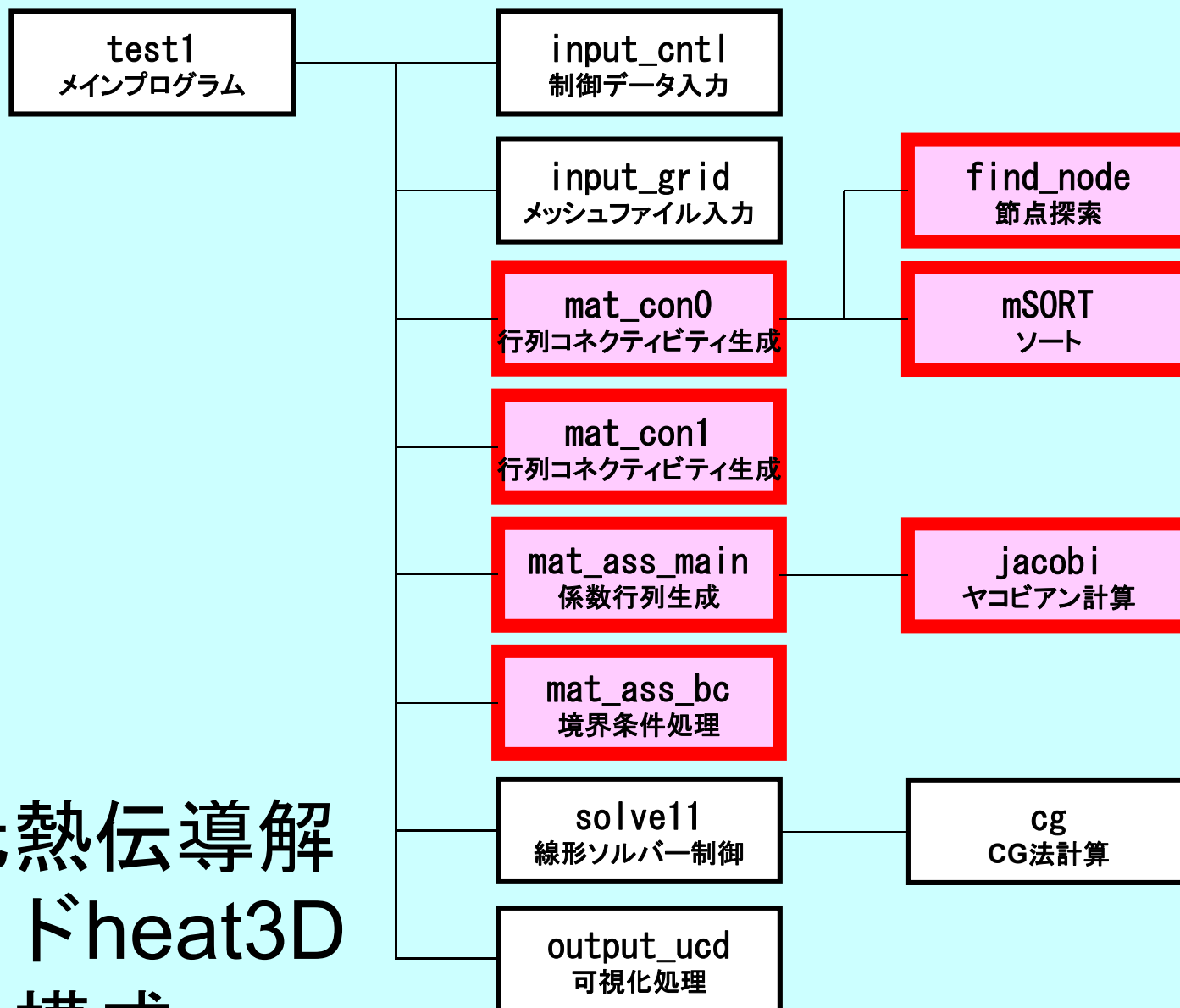
for (k=0;k<NODGRPtot;k++) {
    iS= NODGRP_INDEX[k];
    iE= NODGRP_INDEX[k+1];
    fscanf(fp, "%s", NODGRP_NAME[k]. name);
    nn= iE - iS;
    if( nn != 0 ) {
        for (kk=iS;kk<iE;kk++) fscanf(fp, "%d", &NODGRP_ITEM[kk]);
    }
}

fclose(fp);
}
```

節点グループの中身も「1」から始まる通し節点番号がそのまま読み込まれている。

cube.0 : 節点グループデータ

4 25	50	75	100								NODGRPtot NODGRP_INDEX[1-4]
Xmin 1	6	11	16	21	26	31	36	41	46		NODGRP_NAME[1] NODGRP_ITEM[0-24]
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		NODGRP_NAME[2] NODGRP_ITEM[25-49]
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		NODGRP_NAME[3] NODGRP_ITEM[50-74]
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		NODGRP_NAME[4] NODGRP_ITEM[75-99]
111	112	113	114	115	116	117	118	119	120		
121	122	123	124	125							



三次元熱伝導解析コードheat3Dの構成

Global変数表 : pfem_util.h (1/3)

変数名	種別	サイズ	I/O	内 容
fname	C	[80]	I	メッシュファイル名
N, NP	I		I	節点数
ICELTOT	I		I	要素数
NODGRPtot	I		I	節点グループ数
XYZ	R	[N][3]	I	節点座標
ICELNOD	I	[ICELTOT][8]	I	要素コネクティビティ
NODGRP_INDEX	I	[NODGRPtot+1]	I	各節点グループに含まれる節点数 (累積)
NODGRP_ITEM	I	[NODGRP_INDEX[NODGRPtot+1]]	I	節点グループに含まれる節点
NODGRP_NAME	C80	[NODGRP_INDEX[NODGRPtot+1]]	I	節点グループ名
NLU	I		O	各節点非対角成分数
NPLU	I		O	非対角成分総数
D	R	[N]	O	全体行列 : 対角ブロック
B, X	R	[N]	O	右辺ベクトル, 未知数ベクトル

Global変数表 : pfem_util.h (2/3)

変数名	種別	サイズ	I/O	内 容
AMAT	R	[N]	O	全体行列：非零非対角成分
indexLU	I	[N+1]	O	全体行列：非零非対角成分数
itemLU	I	[NPLU]	O	全体行列：非零非対角成分（列番号）
INLU	I	[N]	O	各節点の非零非対角成分数
IALU	I	[N][NLU]	O	各節点の非零非対角成分数（列番号）
IWKX	I	[N][2]	O	ワーク用配列
ITER, ITERactual	I		I	反復回数の上限, 実際の反復回数
RESID	R		I	打ち切り誤差（1.e-8に設定）
pfemIarray	I	[100]	O	諸定数（整数）
pfemRarray	R	[100]	O	諸定数（実数）

Global変数表 : pfem_util.h (3/3)

変数名	種別	サイズ	I/O	内 容
08th	R		I	=0.125
PNQ, PNE, PNT	R	[2][2][8]	O	各ガウス積分点における $\frac{\partial N_i}{\partial \xi}, \frac{\partial N_i}{\partial \eta}, \frac{\partial N_i}{\partial \zeta} (i=1\sim 8)$
POS, WEI	R	[2]	O	各ガウス積分点の座標, 重み係数
NCOL1, NCOL2	I	[100]	O	ソート用ワーク配列
SHAPE	R	[2][2][2][8]	O	各ガウス積分点における形状関数 $N_i (i=1\sim 8)$
PNX, PNY, PNZ	R	[2][2][2][8]	O	各ガウス積分点における $\frac{\partial N_i}{\partial x}, \frac{\partial N_i}{\partial y}, \frac{\partial N_i}{\partial z} (i=1\sim 8)$
DETJ	R	[2][2][2]	O	各ガウス積分点におけるヤコビアン行列式
COND, QVOL	R		I	熱伝導率, 体積当たり発熱量係数

マトリクス生成まで

- 一次元の場合は, index, itemに関連した情報を簡単に作ることができた
 - 非ゼロ非対角成分の数は2
 - 番号が自分に対して : +1と-1
- 三次元の場合はもっと複雑
 - 非ゼロ非対角成分の数は7~26 (現在の形状)
 - 実際はもっと複雑
 - 前以て, 非ゼロ非対角成分数はわからない



movie

マトリクス生成まで

- 一次元のときは, index, itemに関連した情報を簡単に作ることができた
 - 非ゼロ非対角成分の数は2
 - 番号が自分に対して : +1と-1
- 三次元の場合はもっと複雑
 - 非ゼロ非対角成分の数は7~26 (現在の形状)
 - 実際はもっと複雑
 - 前以て, 非ゼロ非対角成分数はわからない
- INLU[N], IALU[N][NLU] を使って非ゼロ非対角成分数を予備的に勘定する

全体処理

```

/**
    program heat3D
**/
#include <stdio.h>
#include <stdlib.h>
FILE* fp_log;
#define GLOBAL_VALUE_DEFINE
#include "pfem_util.h"
// #include "solver11.h"
extern void INPUT_CNTL();
extern void INPUT_GRID();
extern void MAT_CON0();
extern void MAT_CON1();
extern void MAT_ASS_MAIN();
extern void MAT_ASS_BC();
extern void SOLVE11();
extern void OUTPUT_UCD();
int main()
{
    INPUT_CNTL();
    INPUT_GRID();

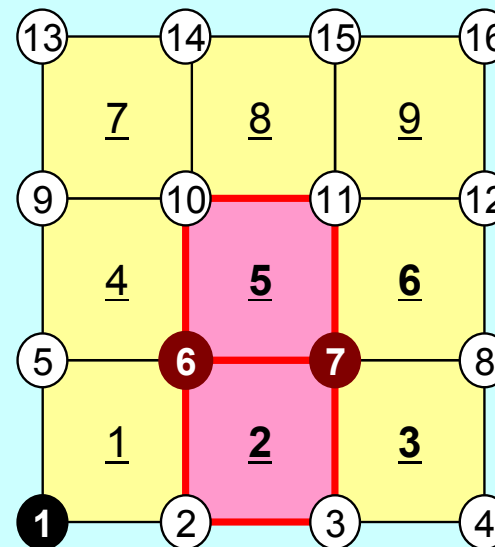
    MAT_CON0();
    MAT_CON1();

    MAT_ASS_MAIN();
    MAT_ASS_BC();

    SOLVE11();

    OUTPUT_UCD();
}

```



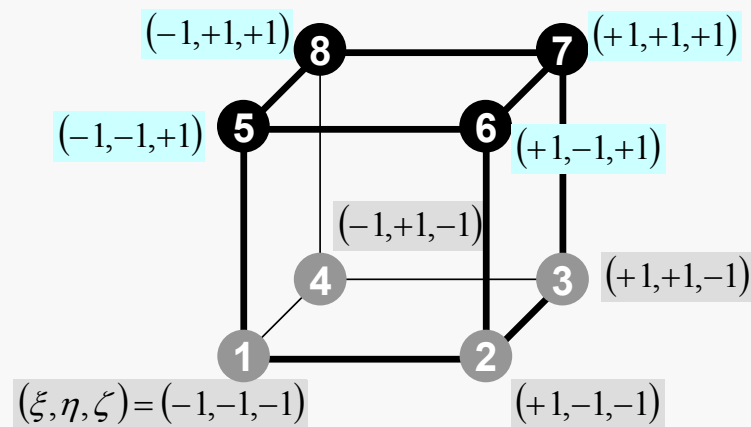
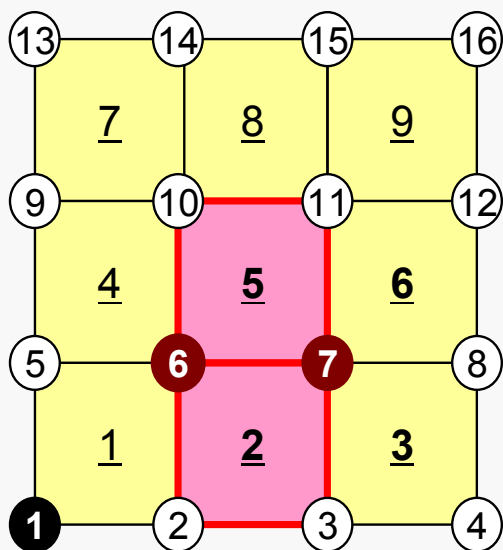
MAT_CON0: INU, IALU生成

MAT_CON1: index, item生成

とりあえず1から始まる節点番号を記憶

MAT_CON0 : 全体構成

```
do icel= 1, ICELTOT
  8節点相互の関係から、
  INLU, IALUを生成
  (FIND_NODE)
enddo
```



行列コネクティビティ生成： MAT_CONO (1/4)

```
/**
** MAT_CONO
**/

#include <stdio.h>
#include "pfem_util.h"
#include "allocate.h"

extern FILE *fp_log;
/** external functions */
extern void mSORT(int*, int*, int);
/** static functions */
static void FIND_TS_NODE (int, int);

void MAT_CONO ()
{
    int i, j, k, icel, in;
    int in1, in2, in3, in4, in5, in6, in7, in8;
    int NN;

    NLU= 26;

    INLU=(KINT* ) allocate_vector (sizeof (KINT), N);
    IALU=(KINT**) allocate_matrix (sizeof (KINT), N, NLU);

    for (i=0; i<N; i++) INLU[i]=0;
    for (i=0; i<N; i++) for (j=0; j<NLU; j++) IALU[i][j]=0;
}
```

NLU:

各節点における
非ゼロ非対角成分
の最大数
(接続する節点数)

今の問題の場合は
わかっているので、
このようにできる

不明の場合の実装：
⇒レポート課題

行列コネクティビティ生成： MAT_CONO (1/4)

```

/**
** MAT_CONO
**/

#include <stdio.h>
#include "pfem_util.h"
#include "allocate.h"

extern FILE *fp_log;
/** external functions */
extern void mSORT(int*, int*, int);
/** static functions */
static void FIND_TS_NODE (int, int);

void MAT_CONO ()
{
    int i, j, k, icel, in;
    int in1, in2, in3, in4, in5, in6, in7, in8;
    int NN;

    NLU= 26;

    INLU=(KINT* ) allocate_vector (sizeof (KINT), N);
    IALU=(KINT**) allocate_matrix (sizeof (KINT), N, NLU);

    for (i=0; i<N; i++) INLU[i]=0;
    for (i=0; i<N; i++) for (j=0; j<NLU; j++) IALU[i][j]=0;

```

変数名	サイズ	内 容
INLU	[N]	各節点の非零非対角成分数
IALU	[N] {[NLU]}	各節点の非零非対角成分 (列番号)

行列コネクティビティ生成： MAT_CON0 (2/4) : 1から始まる番号

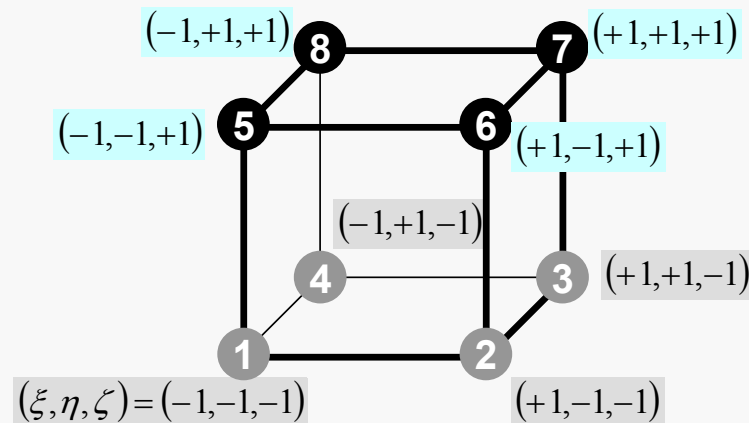
```
for( icel=0; icel< ICELTOT; icel++) {
```

```
  in1=ICELNOD[ icel ][ 0 ];
  in2=ICELNOD[ icel ][ 1 ];
  in3=ICELNOD[ icel ][ 2 ];
  in4=ICELNOD[ icel ][ 3 ];
  in5=ICELNOD[ icel ][ 4 ];
  in6=ICELNOD[ icel ][ 5 ];
  in7=ICELNOD[ icel ][ 6 ];
  in8=ICELNOD[ icel ][ 7 ];
```

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

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

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



節点探索 : FIND_TS_NODE

INLU,IAU探索 : 次元ではこの部分は手動

```

/**
 *** FIND_TS_NODE
 **/

static void FIND_TS_NODE (int ip1, int ip2)
{
  int kk, icou;

  for (kk=1;kk<=INLU[ip1-1];kk++) {
    if(ip2 == IALU[ip1-1][kk-1]) return;
  }

  icou=INLU[ip1-1]+1;
  IALU[ip1-1][icou-1]=ip2;
  INLU[ip1-1]=icou;

  return;
}

```

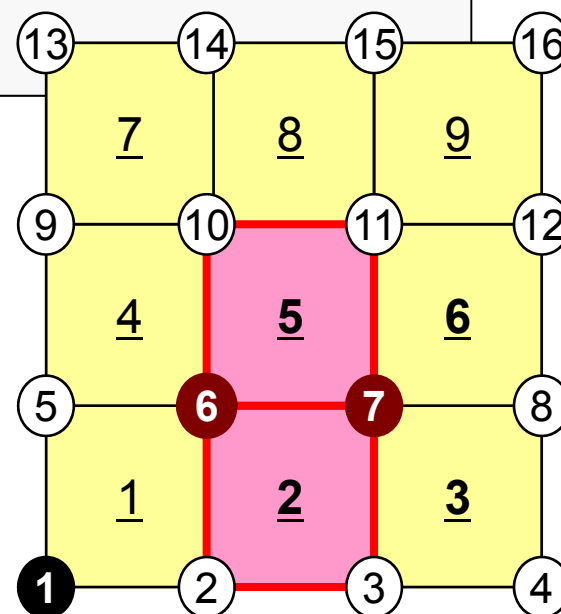
変数名	サイズ	内容
INLU	[N]	各節点の非零非対角成分数
IALU	[N] {[NLU]}	各節点の非零非対角成分 (列番号)

節点探索 : FIND_TS_NODE

一次元ではこの部分は手動

```
/**  
*** FIND_TS_NODE  
**/  
  
static void FIND_TS_NODE (int ip1,int ip2)  
{  
    int kk, icou;  
  
    for (kk=1;kk<=INLU[ip1-1];kk++) {  
        if(ip2 == IALU[ip1-1][kk-1]) return;  
    }  
  
    icou=INLU[ip1-1]+1;  
    IALU[ip1-1][icou-1]=ip2;  
    INLU[ip1-1]=icou;  
  
    return;  
}
```

既にIALUに含まれている
場合は、次のペアへ

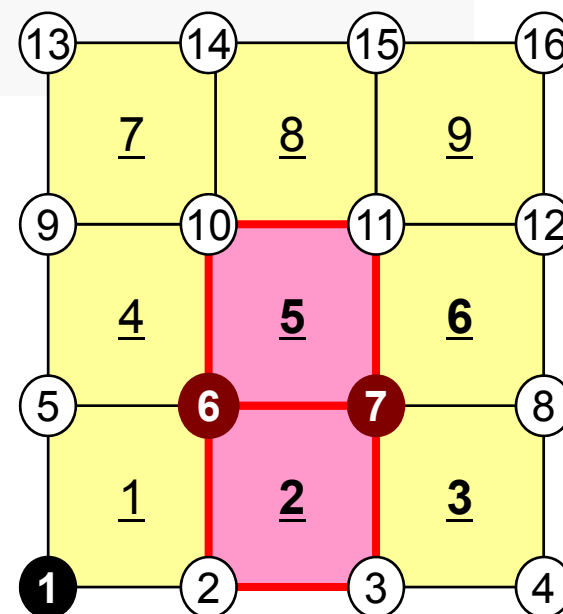


節点探索 : FIND_TS_NODE

一次元ではこの部分は手動

```
/**  
*** FIND_TS_NODE  
**/  
  
static void FIND_TS_NODE (int ip1, int ip2)  
{  
    int kk, icou;  
  
    for (kk=1;kk<=INLU[ip1-1];kk++) {  
        if(ip2 == IALU[ip1-1][kk-1]) return;  
    }  
  
    icou=INLU[ip1-1]+1;  
    IALU[ip1-1][icou-1]=ip2;  
    INLU[ip1-1]=icou;  
  
    return;  
}
```

IALUに含まれていない
場合は, INLUに1を加えて
IALUに格納



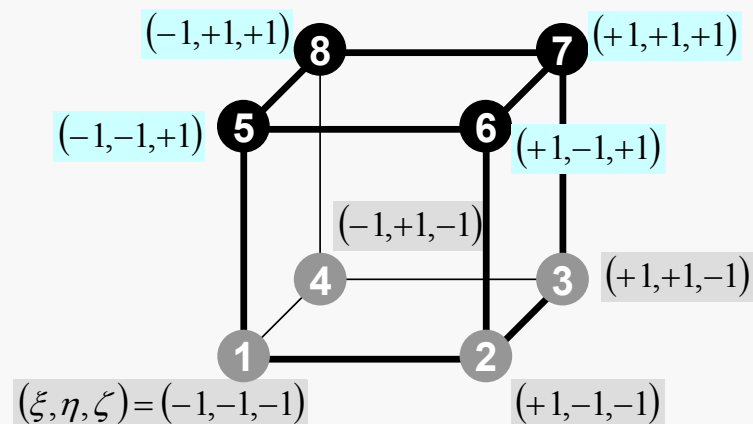
行列コネクティビティ生成： MAT_CON0 (3/4)

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

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

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

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



行列コネクティビティ生成： MAT_CON0 (4/4)

```
FIND_TS_NODE (in8, in1);  
FIND_TS_NODE (in8, in2);  
FIND_TS_NODE (in8, in3);  
FIND_TS_NODE (in8, in4);  
FIND_TS_NODE (in8, in5);  
FIND_TS_NODE (in8, in6);  
FIND_TS_NODE (in8, in7);  
}  
  
for (in=0; in<N; in++) {  
    NN=INLU[in];  
    for (k=0; k<NN; k++) {  
        NCOL1[k]=IALU[in][k];  
    }  
  
    mSORT (NCOL1, NCOL2, NN);  
  
    for (k=NN; k>0; k--) {  
        IALU[in][NN-k]= NCOL1 [NCOL2[k-1]-1];  
    }  
}
```

各節点において、
IALU[i][k]が小さい番号から
大きい番号に並ぶようにソート
(単純なバブルソート)
せいぜい100程度のものをソートする

CRS形式への変換 : MAT_CON1

```

#include <stdio.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE* fp_log;
void MAT_CON1 ()
{
    int i, k, kk;

    indexLU=(KINT*)allocate_vector(sizeof(KINT),N+1);
    for(i=0;i<N+1;i++) indexLU[i]=0;

    for(i=0;i<N;i++){
        indexLU[i+1]=indexLU[i]+INLU[i];
    }

    NPLU=indexLU[N];

    itemLU=(KINT*)allocate_vector(sizeof(KINT),NPLU);

    for(i=0;i<N;i++){
        for(k=0;k<INLU[i];k++){
            kk=k+indexLU[i];
            itemLU[kk]=IALU[i][k]-1;
        }
    }

    deallocate_vector(INLU);
    deallocate_vector(IALU);
}

```

C

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

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

FORTRAN

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

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

CRS形式への変換 : MAT_CON1

```
#include <stdio.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE* fp_log;
void MAT_CON1 ()
{
    int i, k, kk;

    indexLU= (KINT*) allocate_vector (sizeof (KINT), N+1);
    for (i=0; i<N+1; i++) indexLU[i]=0;

    for (i=0; i<N; i++) {
        indexLU[i+1]=indexLU[i]+INLU[i];
    }

    NPLU=indexLU[N];

    itemLU= (KINT*) allocate_vector (sizeof (KINT), NPLU);

    for (i=0; i<N; i++) {
        for (k=0; k<INLU[i]; k++) {
            kk=k+indexLU[i];
            itemLU[kk]=IALU[i][k]-1;
        }
    }

    deallocate_vector (INLU);
    deallocate_vector (IALU);
}

MAT_CON1
```

NPLU=index[N]
itemのサイズ
非ゼロ非対角成分総数

CRS形式への変換 : MAT_CON1

```
#include <stdio.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE* fp_log;
void MAT_CON1 ()
{
    int i, k, kk;

    indexLU= (KINT*) allocate_vector (sizeof (KINT), N+1);
    for (i=0; i<N+1; i++) indexLU[i]=0;

    for (i=0; i<N; i++) {
        indexLU[i+1]=indexLU[i]+INLU[i];
    }

    NPLU=indexLU[N];

    itemLU= (KINT*) allocate_vector (sizeof (KINT), NPLU);

    for (i=0; i<N; i++) {
        for (k=0; k<INLU[i]; k++) {
            kk=k+indexLU[i];
            itemLU[kk]=IALU[i][k]-1;
        }
    }

    deallocate_vector (INLU);
    deallocate_vector (IALU);
}

MAT_CON1
```

itemに「0」から始まる
節点番号を記憶

CRS形式への変換 : MAT_CON1

```
#include <stdio.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE* fp_log;
void MAT_CON1 ()
{
    int i, k, kk;

    indexLU= (KINT*) allocate_vector (sizeof (KINT), N+1);
    for (i=0; i<N+1; i++) indexLU[i]=0;

    for (i=0; i<N; i++) {
        indexLU[i+1]=indexLU[i]+INLU[i];
    }

    NPLU=indexLU[N];

    itemLU= (KINT*) allocate_vector (sizeof (KINT), NPLU);

    for (i=0; i<N; i++) {
        for (k=0; k<INLU[i]; k++) {
            kk=k+indexLU[i];
            itemLU[kk]=IALU[i][k]-1;
        }
    }

    deallocate_vector (INLU);
    deallocate_vector (IALU);
}
MAT_CON1
```

これらはもはや不要

全体処理

```
/**
    program heat3D
**/
#include <stdio.h>
#include <stdlib.h>
FILE* fp_log;
#define GLOBAL_VALUE_DEFINE
#include "pfem_util.h"
// #include "solver11.h"
extern void INPUT_CNTL ();
extern void INPUT_GRID ();
extern void MAT_CONO ();
extern void MAT_CON1 ();
extern void MAT_ASS_MAIN ();
extern void MAT_ASS_BC ();
extern void SOLVE11 ();
extern void OUTPUT_UCD ();
int main()
{
    INPUT_CNTL ();
    INPUT_GRID ();

    MAT_CONO ();
    MAT_CON1 ();

    MAT_ASS_MAIN ();
    MAT_ASS_BC ();

    SOLVE11 ();

    OUTPUT_UCD ();
}
```

MAT_ASS_MAIN : 全体構成

```

do kpn= 1, 2      ガウス積分点番号 (ζ方向)
  do jpn= 1, 2    ガウス積分点番号 (η方向)
    do ipn= 1, 2  ガウス積分点番号 (ξ方向)
      ガウス積分点 (8個) における形状関数,
      およびその「自然座標系」における微分の算出
    enddo
  enddo
enddo

```

```

do icel= 1, ICELTOT  要素ループ
  8節点の座標から, ガウス積分点における, 形状関数の「全体座標系」における微分,
  およびヤコビアンを算出 (JACOBI)

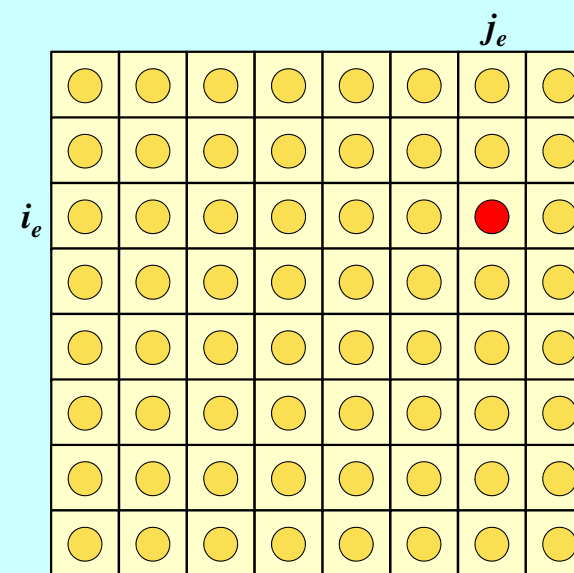
```

```

do ie= 1, 8      局所節点番号
  do je= 1, 8    局所節点番号
    全体節点番号 : ip, jp
    Aip, jp の itemLUI におけるアドレス : kk

    do kpn= 1, 2  ガウス積分点番号 (ζ方向)
      do jpn= 1, 2  ガウス積分点番号 (η方向)
        do ipn= 1, 2  ガウス積分点番号 (ξ方向)
          要素積分⇒要素行列成分計算, 全体行列への足しこみ
        enddo
      enddo
    enddo
  enddo
enddo
enddo

```



係数行列 : MAT_ASS_MAIN (1/6)

```

#include <stdio.h>
#include <math.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE *fp_log;
extern void JACOBI();
void MAT_ASS_MAIN()
{
    int i, k, kk;
    int ip, jp, kp;
    int ipn, jpn, kpn;
    int icel;
    int ie, je;
    int iiS, iiE;
    int in1, in2, in3, in4, in5, in6, in7, in8;
    double SHi;
    double QP1, QM1, EP1, EM1, TP1, TM1;
    double X1, X2, X3, X4, X5, X6, X7, X8;
    double Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8;
    double Z1, Z2, Z3, Z4, Z5, Z6, Z7, Z8;
    double PNXi, PNYi, PNZi, PNXj, PNYj, PNZj;
    double CONDO, QVO, QVC, COEFij;
    double coef;

```

```
KINT nodLOCAL[8];
```

```

AMAT=(KREAL*) allocate_vector(sizeof(KREAL), NPLU);
B=(KREAL*) allocate_vector(sizeof(KREAL), N);
D=(KREAL*) allocate_vector(sizeof(KREAL), N);
X=(KREAL*) allocate_vector(sizeof(KREAL), N);

```

係数行列 (非零非対角成分)
 右辺ベクトル
 解ベクトル
 係数行列 (対角成分)

```

for (i=0; i<NPLU; i++) AMAT[i]=0.0;
for (i=0; i<N; i++) B[i]=0.0;
for (i=0; i<N; i++) D[i]=0.0;
for (i=0; i<N; i++) X[i]=0.0;

```

```

WEI[0]= 1.0000000000e0;
WEI[1]= 1.0000000000e0;
POS[0]= -0.5773502692e0;
POS[1]= 0.5773502692e0;

```

係数行列 : MAT_ASS_MAIN (1/6)

```

#include <stdio.h>
#include <math.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE *fp_log;
extern void JACOBI();
void MAT_ASS_MAIN()
{
    int i, k, kk;
    int ip, jp, kp;
    int ipn, jpn, kpn;
    int icel;
    int ie, je;
    int iiS, iiE;
    int in1, in2, in3, in4, in5, in6, in7, in8;
    double SHi;
    double QP1, QM1, EP1, EM1, TP1, TM1;
    double X1, X2, X3, X4, X5, X6, X7, X8;
    double Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8;
    double Z1, Z2, Z3, Z4, Z5, Z6, Z7, Z8;
    double PNXi, PNYi, PNZi, PNXj, PNYj, PNZj;
    double CONDO, QVO, QVC, COEFij;
    double coef;

    KINT nodLOCAL[8];

    AMAT=(KREAL*) allocate_vector(sizeof(KREAL), NPLU);
    B=(KREAL*) allocate_vector(sizeof(KREAL), N);
    D=(KREAL*) allocate_vector(sizeof(KREAL), N);
    X=(KREAL*) allocate_vector(sizeof(KREAL), N);

    for(i=0; i<NPLU; i++) AMAT[i]=0.0;
    for(i=0; i<N; i++) B[i]=0.0;
    for(i=0; i<N; i++) D[i]=0.0;
    for(i=0; i<N; i++) X[i]=0.0;

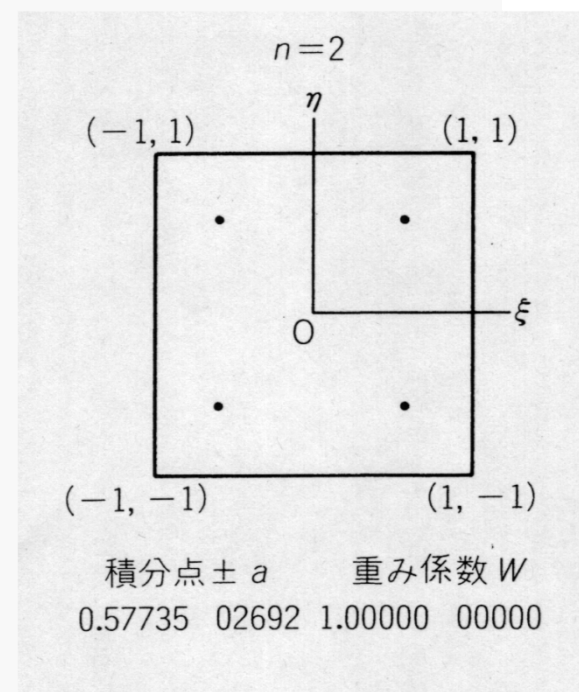
```

```

WEI[0]= 1.000000000e0;
WEI[1]= 1.000000000e0;
POS[0]= -0.5773502692e0;
POS[1]= 0.5773502692e0;

```

POS: 積分点座標
WEI: 重み係数



系数行列：MAT_ASS_MAIN (2/6)

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

for(ip=0; ip<2; ip++) {
  for(jp=0; jp<2; jp++) {
    for(kp=0; kp<2; kp++) {

      QP1= 1. e0 + POS[ip];
      QM1= 1. e0 - POS[ip];
      EP1= 1. e0 + POS[jp];
      EM1= 1. e0 - POS[jp];
      TP1= 1. e0 + POS[kp];
      TM1= 1. e0 - POS[kp];

      SHAPE[ip][jp][kp][0]= 08th * QM1 * EM1 * TM1;
      SHAPE[ip][jp][kp][1]= 08th * QP1 * EM1 * TM1;
      SHAPE[ip][jp][kp][2]= 08th * QP1 * EP1 * TM1;
      SHAPE[ip][jp][kp][3]= 08th * QM1 * EP1 * TM1;
      SHAPE[ip][jp][kp][4]= 08th * QM1 * EM1 * TP1;
      SHAPE[ip][jp][kp][5]= 08th * QP1 * EM1 * TP1;
      SHAPE[ip][jp][kp][6]= 08th * QP1 * EP1 * TP1;
      SHAPE[ip][jp][kp][7]= 08th * QM1 * EP1 * TP1;
```


系数行列：MAT_ASS_MAIN (2/6)

```

/**
  INIT.
  PNQ - 1st-order derivative of shape function by QSI
  PNE - 1st-order derivative of shape function by ETA
  PNT - 1st-order derivative of shape function by ZET
***/

```

```

for(ip=0; ip<2; ip++) {
  for(jp=0; jp<2; jp++) {
    for(kp=0; kp<2; kp++) {

```

```

      QP1= 1. e0 + POS[ip];
      QM1= 1. e0 - POS[ip];
      EP1= 1. e0 + POS[jp];
      EM1= 1. e0 - POS[jp];
      TP1= 1. e0 + POS[kp];
      TM1= 1. e0 - POS[kp];

```

```

      SHAPE[ip][jp][kp][0]= 08th * QM1 * EM1 * TM1;
      SHAPE[ip][jp][kp][1]= 08th * QP1 * EM1 * TM1;
      SHAPE[ip][jp][kp][2]= 08th * QP1 * EP1 * TM1;
      SHAPE[ip][jp][kp][3]= 08th * QM1 * EP1 * TM1;
      SHAPE[ip][jp][kp][4]= 08th * QM1 * EM1 * TP1;
      SHAPE[ip][jp][kp][5]= 08th * QP1 * EM1 * TP1;
      SHAPE[ip][jp][kp][6]= 08th * QP1 * EP1 * TP1;
      SHAPE[ip][jp][kp][7]= 08th * QM1 * EP1 * TP1;

```

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

$$EP1(j) = (1 + \eta_j), \quad EM1(j) = (1 - \eta_j)$$

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

系数行列：MAT_ASS_MAIN (2/6)

```

/**
  INIT.
  PNQ - 1st-order derivative of shape function by QSI
  PNE - 1st-order derivative of shape function by ETA
  PNT - 1st-order derivative of shape function by ZET
***/

```

```

for (ip=0; ip<2; ip++) {
  for (jp=0; jp<2; jp++) {
    for (kp=0; kp<2; kp++) {

```

```

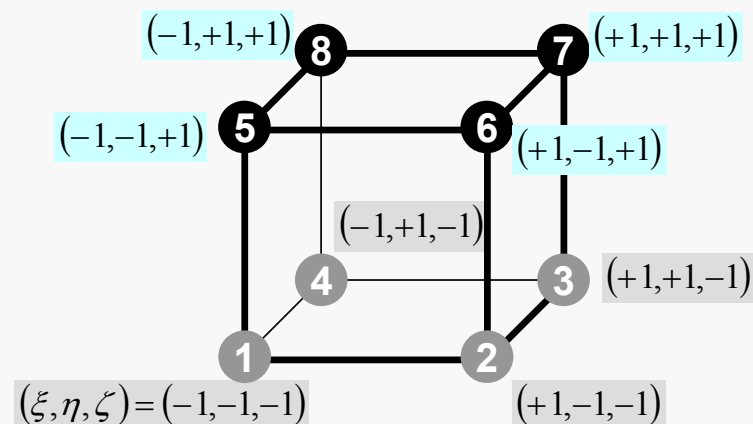
      QP1= 1. e0 + POS[ip]:
      QM1= 1. e0 - POS[ip]:
      EP1= 1. e0 + POS[jp]:
      EM1= 1. e0 - POS[jp]:
      TP1= 1. e0 + POS[kp]:
      TM1= 1. e0 - POS[kp]:

```

```

      SHAPE[ip][jp][kp][0]= 08th * QM1 * EM1 * TM1;
      SHAPE[ip][jp][kp][1]= 08th * QP1 * EM1 * TM1;
      SHAPE[ip][jp][kp][2]= 08th * QP1 * EP1 * TM1;
      SHAPE[ip][jp][kp][3]= 08th * QM1 * EP1 * TM1;
      SHAPE[ip][jp][kp][4]= 08th * QM1 * EM1 * TP1;
      SHAPE[ip][jp][kp][5]= 08th * QP1 * EM1 * TP1;
      SHAPE[ip][jp][kp][6]= 08th * QP1 * EP1 * TP1;
      SHAPE[ip][jp][kp][7]= 08th * QM1 * EP1 * TP1;

```



系数行列：MAT_ASS_MAIN (2/6)

```

/**
  INIT.
  PNQ - 1st-order derivative of shape function by QSI
  PNE - 1st-order derivative of shape function by ETA
  PNT - 1st-order derivative of shape function by ZET
***/

for (ip=0; ip<2; ip++) {
  for (jp=0; jp<2; jp++) {
    for (kp=0; kp<2; kp++) {

      QP1= 1. e0 + POS[ip]:
      QM1= 1. e0 - POS[ip]:
      EP1= 1. e0 + POS[jp]:
      EM1= 1. e0 - POS[jp]:
      TP1= 1. e0 + POS[kp]:
      TM1= 1. e0 - POS[kp]:

      SHAPE[ip][jp][kp][0]= 08th * QM1 * EM1 * TM1;
      SHAPE[ip][jp][kp][1]= 08th * QP1 * EM1 * TM1;
      SHAPE[ip][jp][kp][2]= 08th * QP1 * EP1 * TM1;
      SHAPE[ip][jp][kp][3]= 08th * QM1 * EP1 * TM1;
      SHAPE[ip][jp][kp][4]= 08th * QM1 * EM1 * TP1;
      SHAPE[ip][jp][kp][5]= 08th * QP1 * EM1 * TP1;
      SHAPE[ip][jp][kp][6]= 08th * QP1 * EP1 * TP1;
      SHAPE[ip][jp][kp][7]= 08th * QM1 * EP1 * TP1;
    }
  }
}

```

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

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

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

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

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

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

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

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

係数行列 : MAT_ASS_MAIN (3/6)

```

PNQ [jp] [kp] [0] = - 08th * EM1 * TM1 ;
PNQ [jp] [kp] [1] = + 08th * EM1 * TM1 ;
PNQ [jp] [kp] [2] = + 08th * EP1 * TM1 ;
PNQ [jp] [kp] [3] = - 08th * EP1 * TM1 ;
PNQ [jp] [kp] [4] = - 08th * EM1 * TP1 ;
PNQ [jp] [kp] [5] = + 08th * EM1 * TP1 ;
PNQ [jp] [kp] [6] = + 08th * EP1 * TP1 ;
PNQ [jp] [kp] [7] = - 08th * EP1 * TP1 ;

```

```

PNE [ip] [kp] [0] = - 08th * QM1 * TM1 ;
PNE [ip] [kp] [1] = - 08th * QP1 * TM1 ;
PNE [ip] [kp] [2] = + 08th * QP1 * TM1 ;
PNE [ip] [kp] [3] = + 08th * QM1 * TM1 ;
PNE [ip] [kp] [4] = - 08th * QM1 * TP1 ;
PNE [ip] [kp] [5] = - 08th * QP1 * TP1 ;
PNE [ip] [kp] [6] = + 08th * QP1 * TP1 ;
PNE [ip] [kp] [7] = + 08th * QM1 * TP1 ;

```

```

PNT [ip] [jp] [0] = - 08th * QM1 * EM1 ;
PNT [ip] [jp] [1] = - 08th * QP1 * EM1 ;
PNT [ip] [jp] [2] = - 08th * QP1 * EP1 ;
PNT [ip] [jp] [3] = - 08th * QM1 * EP1 ;
PNT [ip] [jp] [4] = + 08th * QM1 * EM1 ;
PNT [ip] [jp] [5] = + 08th * QP1 * EM1 ;
PNT [ip] [jp] [6] = + 08th * QP1 * EP1 ;
PNT [ip] [jp] [7] = + 08th * QM1 * EP1 ;

```

```

}
}
for ( icel=0; icel < ICELTOT; icel++) {
  CONDO= COND;

```

```

in1=ICELNOD [ icel ] [ 0 ] ;
in2=ICELNOD [ icel ] [ 1 ] ;
in3=ICELNOD [ icel ] [ 2 ] ;
in4=ICELNOD [ icel ] [ 3 ] ;
in5=ICELNOD [ icel ] [ 4 ] ;
in6=ICELNOD [ icel ] [ 5 ] ;
in7=ICELNOD [ icel ] [ 6 ] ;
in8=ICELNOD [ icel ] [ 7 ] ;

```

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

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

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

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

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

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

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

(ξ_i, η_j, ζ_k) における形状関数の一階微分

系数行列：MAT_ASS_MAIN (3/6)

```

PNQ [jp] [kp] [0] = - 08th * EM1 * TM1 ;
PNQ [jp] [kp] [1] = + 08th * EM1 * TM1 ;
PNQ [jp] [kp] [2] = + 08th * EP1 * TM1 ;
PNQ [jp] [kp] [3] = - 08th * EP1 * TM1 ;
PNQ [jp] [kp] [4] = - 08th * EM1 * TP1 ;
PNQ [jp] [kp] [5] = + 08th * EM1 * TP1 ;
PNQ [jp] [kp] [6] = + 08th * EP1 * TP1 ;
PNQ [jp] [kp] [7] = - 08th * EP1 * TP1 ;

```

```

PNE [ip] [kp] [0] = - 08th * QM1 * TM1 ;
PNE [ip] [kp] [1] = - 08th * QP1 * TM1 ;
PNE [ip] [kp] [2] = + 08th * QP1 * TM1 ;
PNE [ip] [kp] [3] = + 08th * QM1 * TM1 ;
PNE [ip] [kp] [4] = - 08th * QM1 * TP1 ;
PNE [ip] [kp] [5] = - 08th * QP1 * TP1 ;
PNE [ip] [kp] [6] = + 08th * QP1 * TP1 ;
PNE [ip] [kp] [7] = + 08th * QM1 * TP1 ;

```

```

PNT [ip] [jp] [0] = - 08th * QM1 * EM1 ;
PNT [ip] [jp] [1] = - 08th * QP1 * EM1 ;
PNT [ip] [jp] [2] = - 08th * QP1 * EP1 ;
PNT [ip] [jp] [3] = - 08th * QM1 * EP1 ;
PNT [ip] [jp] [4] = + 08th * QM1 * EM1 ;
PNT [ip] [jp] [5] = + 08th * QP1 * EM1 ;
PNT [ip] [jp] [6] = + 08th * QP1 * EP1 ;
PNT [ip] [jp] [7] = + 08th * QM1 * EP1 ;

```

```

}
}
}

```

```

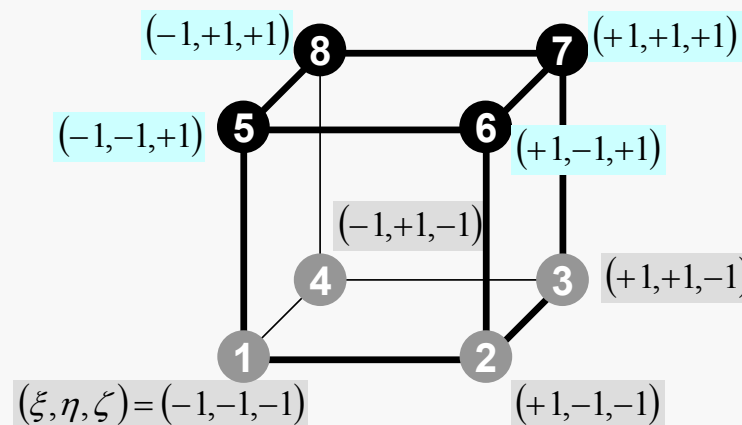
for ( icel=0; icel < ICELTOT; icel++) {
  CONDO= COND;

```

```

  in1=ICELNOD [ icel ] [ 0 ] ;
  in2=ICELNOD [ icel ] [ 1 ] ;
  in3=ICELNOD [ icel ] [ 2 ] ;
  in4=ICELNOD [ icel ] [ 3 ] ;
  in5=ICELNOD [ icel ] [ 4 ] ;
  in6=ICELNOD [ icel ] [ 5 ] ;
  in7=ICELNOD [ icel ] [ 6 ] ;
  in8=ICELNOD [ icel ] [ 7 ] ;

```



係数行列 : MAT_ASS_MAIN (4/6)

```

nodLOCAL [0] = in1 ;
nodLOCAL [1] = in2 ;
nodLOCAL [2] = in3 ;
nodLOCAL [3] = in4 ;
nodLOCAL [4] = in5 ;
nodLOCAL [5] = in6 ;
nodLOCAL [6] = in7 ;
nodLOCAL [7] = in8 ;

```

```

X1=XYZ[in1-1][0] ;
X2=XYZ[in2-1][0] ;
X3=XYZ[in3-1][0] ;
X4=XYZ[in4-1][0] ;
X5=XYZ[in5-1][0] ;
X6=XYZ[in6-1][0] ;
X7=XYZ[in7-1][0] ;
X8=XYZ[in8-1][0] ;

```

```

Y1=XYZ[in1-1][1] ;
Y2=XYZ[in2-1][1] ;
Y3=XYZ[in3-1][1] ;
Y4=XYZ[in4-1][1] ;
Y5=XYZ[in5-1][1] ;
Y6=XYZ[in6-1][1] ;
Y7=XYZ[in7-1][1] ;
Y8=XYZ[in8-1][1] ;

```

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

```

Z1=XYZ[in1-1][2] ;
Z2=XYZ[in2-1][2] ;
Z3=XYZ[in3-1][2] ;
Z4=XYZ[in4-1][2] ;
Z5=XYZ[in5-1][2] ;
Z6=XYZ[in6-1][2] ;
Z7=XYZ[in7-1][2] ;
Z8=XYZ[in8-1][2] ;

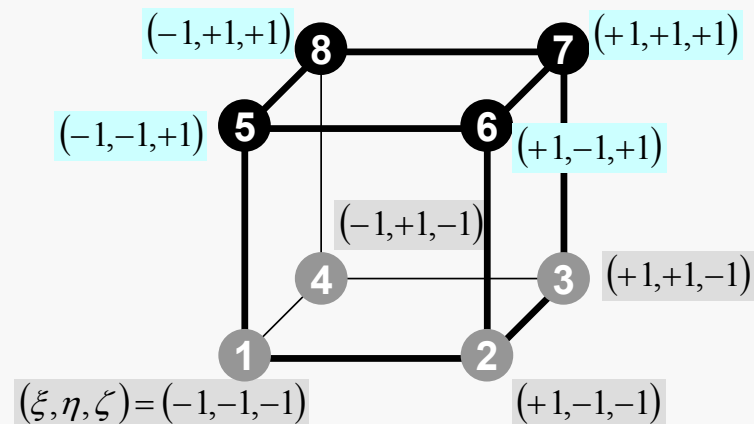
```

```

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) ;

```

8節点の節点番号



係数行列 : MAT_ASS_MAIN (4/6)

```
nodLOCAL [0] = in1 ;
nodLOCAL [1] = in2 ;
nodLOCAL [2] = in3 ;
nodLOCAL [3] = in4 ;
nodLOCAL [4] = in5 ;
nodLOCAL [5] = in6 ;
nodLOCAL [6] = in7 ;
nodLOCAL [7] = in8 ;
```

```
X1=XYZ[in1-1][0] ;
X2=XYZ[in2-1][0] ;
X3=XYZ[in3-1][0] ;
X4=XYZ[in4-1][0] ;
X5=XYZ[in5-1][0] ;
X6=XYZ[in6-1][0] ;
X7=XYZ[in7-1][0] ;
X8=XYZ[in8-1][0] ;
```

8節点のX座標

```
Y1=XYZ[in1-1][1] ;
Y2=XYZ[in2-1][1] ;
Y3=XYZ[in3-1][1] ;
Y4=XYZ[in4-1][1] ;
Y5=XYZ[in5-1][1] ;
Y6=XYZ[in6-1][1] ;
Y7=XYZ[in7-1][1] ;
Y8=XYZ[in8-1][1] ;
```

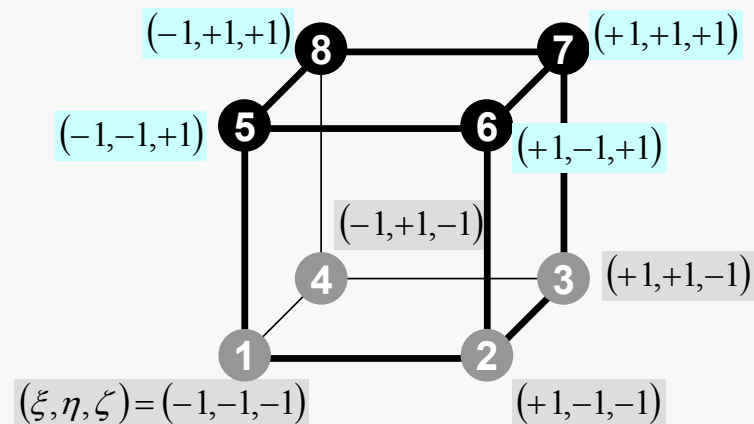
8節点のY座標

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

```
Z1=XYZ[in1-1][2] ;
Z2=XYZ[in2-1][2] ;
Z3=XYZ[in3-1][2] ;
Z4=XYZ[in4-1][2] ;
Z5=XYZ[in5-1][2] ;
Z6=XYZ[in6-1][2] ;
Z7=XYZ[in7-1][2] ;
Z8=XYZ[in8-1][2] ;
```

8節点のZ座標

```
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) ;
```



座標値：
節点番号から1引く

係数行列 : MAT_ASS_MAIN (4/6)

```
nodLOCAL [0] = in1 ;
nodLOCAL [1] = in2 ;
nodLOCAL [2] = in3 ;
nodLOCAL [3] = in4 ;
nodLOCAL [4] = in5 ;
nodLOCAL [5] = in6 ;
nodLOCAL [6] = in7 ;
nodLOCAL [7] = in8 ;
```

```
X1=XYZ[in1-1][0] ;
X2=XYZ[in2-1][0] ;
X3=XYZ[in3-1][0] ;
X4=XYZ[in4-1][0] ;
X5=XYZ[in5-1][0] ;
X6=XYZ[in6-1][0] ;
X7=XYZ[in7-1][0] ;
X8=XYZ[in8-1][0] ;
```

```
Y1=XYZ[in1-1][1] ;
Y2=XYZ[in2-1][1] ;
Y3=XYZ[in3-1][1] ;
Y4=XYZ[in4-1][1] ;
Y5=XYZ[in5-1][1] ;
Y6=XYZ[in6-1][1] ;
Y7=XYZ[in7-1][1] ;
Y8=XYZ[in8-1][1] ;
```

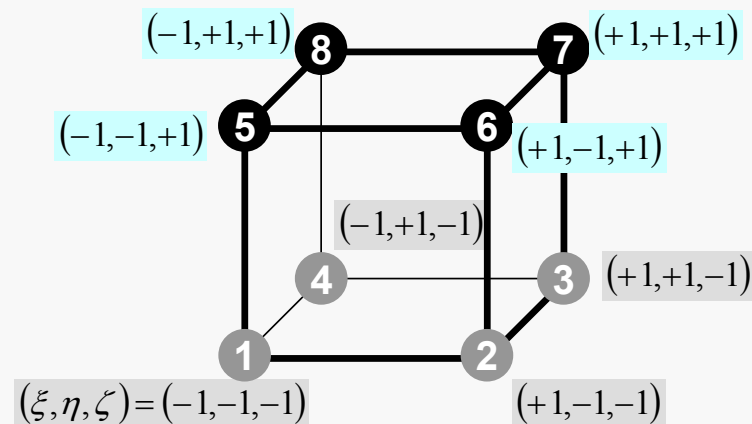
QVC= 08th*(X1+X2+X3+X4+X5+X6+X7+X8+Y1+Y2+Y3+Y4+Y5+Y6+Y7+Y8) ;

```
Z1=XYZ[in1-1][2] ;
Z2=XYZ[in2-1][2] ;
Z3=XYZ[in3-1][2] ;
Z4=XYZ[in4-1][2] ;
Z5=XYZ[in5-1][2] ;
Z6=XYZ[in6-1][2] ;
Z7=XYZ[in7-1][2] ;
Z8=XYZ[in8-1][2] ;
```

```
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) ;
```

8節点のX座標

8節点のY座標



座標値：
節点番号から1引く

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

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

体積当たり発熱量は位置 (メッシュの中心の座標 x_c, y_c) に依存

係数行列 : MAT_ASS_MAIN (4/6)

```
nodLOCAL [0] = in1 ;
nodLOCAL [1] = in2 ;
nodLOCAL [2] = in3 ;
nodLOCAL [3] = in4 ;
nodLOCAL [4] = in5 ;
nodLOCAL [5] = in6 ;
nodLOCAL [6] = in7 ;
nodLOCAL [7] = in8 ;
```

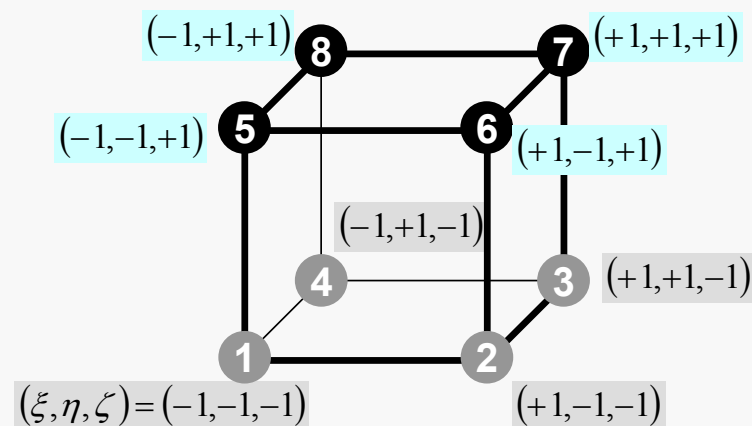
```
X1=XYZ[in1-1][0] ;
X2=XYZ[in2-1][0] ;
X3=XYZ[in3-1][0] ;
X4=XYZ[in4-1][0] ;
X5=XYZ[in5-1][0] ;
X6=XYZ[in6-1][0] ;
X7=XYZ[in7-1][0] ;
X8=XYZ[in8-1][0] ;
```

```
Y1=XYZ[in1-1][1] ;
Y2=XYZ[in2-1][1] ;
Y3=XYZ[in3-1][1] ;
Y4=XYZ[in4-1][1] ;
Y5=XYZ[in5-1][1] ;
Y6=XYZ[in6-1][1] ;
Y7=XYZ[in7-1][1] ;
Y8=XYZ[in8-1][1] ;
```

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

```
Z1=XYZ[in1-1][2] ;
Z2=XYZ[in2-1][2] ;
Z3=XYZ[in3-1][2] ;
Z4=XYZ[in4-1][2] ;
Z5=XYZ[in5-1][2] ;
Z6=XYZ[in6-1][2] ;
Z7=XYZ[in7-1][2] ;
Z8=XYZ[in8-1][2] ;
```

```
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) ;
```



座標値：
節点番号から1引く

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

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

$$QVC = |x_c + y_c|$$

系数行列：MAT_ASS_MAIN (4/6)

```
nodLOCAL [0]= in1 ;  
nodLOCAL [1]= in2 ;  
nodLOCAL [2]= in3 ;  
nodLOCAL [3]= in4 ;  
nodLOCAL [4]= in5 ;  
nodLOCAL [5]= in6 ;  
nodLOCAL [6]= in7 ;  
nodLOCAL [7]= in8 ;
```

```
X1=XYZ[in1-1][0] ;  
X2=XYZ[in2-1][0] ;  
X3=XYZ[in3-1][0] ;  
X4=XYZ[in4-1][0] ;  
X5=XYZ[in5-1][0] ;  
X6=XYZ[in6-1][0] ;  
X7=XYZ[in7-1][0] ;  
X8=XYZ[in8-1][0] ;
```

```
Y1=XYZ[in1-1][1] ;  
Y2=XYZ[in2-1][1] ;  
Y3=XYZ[in3-1][1] ;  
Y4=XYZ[in4-1][1] ;  
Y5=XYZ[in5-1][1] ;  
Y6=XYZ[in6-1][1] ;  
Y7=XYZ[in7-1][1] ;  
Y8=XYZ[in8-1][1] ;
```

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

```
Z1=XYZ[in1-1][2] ;  
Z2=XYZ[in2-1][2] ;  
Z3=XYZ[in3-1][2] ;  
Z4=XYZ[in4-1][2] ;  
Z5=XYZ[in5-1][2] ;  
Z6=XYZ[in6-1][2] ;  
Z7=XYZ[in7-1][2] ;  
Z8=XYZ[in8-1][2] ;
```

```
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)

```

#include <stdio.h>
#include <math.h>
#include "precision.h"
#include "allocate.h"
/**
 *** JACOBI
 ***/
void JACOBI (
    KREAL DETJ[2][2][2],
    KREAL PNQ[2][2][8], KREAL PNE[2][2][8], KREAL PNT[2][2][8],
    KREAL PNX[2][2][8], KREAL PNY[2][2][8], KREAL PNZ[2][2][8],
    KREAL X1, KREAL X2, KREAL X3, KREAL X4, KREAL X5, KREAL X6, KREAL X7, KREAL X8,
    KREAL Y1, KREAL Y2, KREAL Y3, KREAL Y4, KREAL Y5, KREAL Y6, KREAL Y7, KREAL Y8,
    KREAL Z1, KREAL Z2, KREAL Z3, KREAL Z4, KREAL Z5, KREAL Z6, KREAL Z7, KREAL Z8)
{
/**
    calculates JACOBIAN & INVERSE JACOBIAN
        dNi/dx, dNi/dy & dNi/dz
**/

    int ip, jp, kp;
    double dXdQ, dYdQ, dZdQ, dXdE, dYdE, dZdE, dXdT, dYdT, dZdT;
    double coef;
    double a11, a12, a13, a21, a22, a23, a31, a32, a33;

    for (ip=0; ip<2; ip++) {
        for (jp=0; jp<2; jp++) {
            for (kp=0; kp<2; kp++) {
                PNQ[ip][jp][kp][0]=0.0;
                PNQ[ip][jp][kp][1]=0.0;
                PNQ[ip][jp][kp][2]=0.0;
                PNQ[ip][jp][kp][3]=0.0;
                PNQ[ip][jp][kp][4]=0.0;
                PNQ[ip][jp][kp][5]=0.0;
                PNQ[ip][jp][kp][6]=0.0;
                PNQ[ip][jp][kp][7]=0.0;
            }
        }
    }
}

```

入力

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

出力

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

各ガウス積分点[ip][jp][kp]における値

自然座標系における偏微分 (1/4)

- 偏微分の公式より以下のようなになる :

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

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

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

$\left[\frac{\partial N_i}{\partial \xi}, \frac{\partial N_i}{\partial \eta}, \frac{\partial N_i}{\partial \zeta} \right]$ は定義より簡単に求められるが

$\left[\frac{\partial N_i}{\partial x}, \frac{\partial N_i}{\partial y}, \frac{\partial N_i}{\partial z} \right]$ を実際の計算で使用する

自然座標系における偏微分 (2/4)

- マトリックス表示すると：

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

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

$[J]$: ヤコビのマトリクス
(Jacobi matrix
Jacobian)

自然座標系における偏微分 (3/4)

- N_i の定義より簡単に求められる

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

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

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

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

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

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

JACOBI (2/4)

```

PNY[ip][jp][kp][0]=0.0;
PNY[ip][jp][kp][1]=0.0;
PNY[ip][jp][kp][2]=0.0;
PNY[ip][jp][kp][3]=0.0;
PNY[ip][jp][kp][4]=0.0;
PNY[ip][jp][kp][5]=0.0;
PNY[ip][jp][kp][6]=0.0;
PNY[ip][jp][kp][7]=0.0;

```

```

PNZ[ip][jp][kp][0]=0.0;
PNZ[ip][jp][kp][1]=0.0;
PNZ[ip][jp][kp][2]=0.0;
PNZ[ip][jp][kp][3]=0.0;
PNZ[ip][jp][kp][4]=0.0;
PNZ[ip][jp][kp][5]=0.0;
PNZ[ip][jp][kp][6]=0.0;
PNZ[ip][jp][kp][7]=0.0;

```

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

/**

DETERMINANT of the JACOBIAN

**/

```

dXdQ = PNQ[jp][kp][0]*X1 + PNQ[jp][kp][1]*X2
      + PNQ[jp][kp][2]*X3 + PNQ[jp][kp][3]*X4
      + PNQ[jp][kp][4]*X5 + PNQ[jp][kp][5]*X6
      + PNQ[jp][kp][6]*X7 + PNQ[jp][kp][7]*X8;
dYdQ = PNQ[jp][kp][0]*Y1 + PNQ[jp][kp][1]*Y2
      + PNQ[jp][kp][2]*Y3 + PNQ[jp][kp][3]*Y4
      + PNQ[jp][kp][4]*Y5 + PNQ[jp][kp][5]*Y6
      + PNQ[jp][kp][6]*Y7 + PNQ[jp][kp][7]*Y8;
dZdQ = PNQ[jp][kp][0]*Z1 + PNQ[jp][kp][1]*Z2
      + PNQ[jp][kp][2]*Z3 + PNQ[jp][kp][3]*Z4
      + PNQ[jp][kp][4]*Z5 + PNQ[jp][kp][5]*Z6
      + PNQ[jp][kp][6]*Z7 + PNQ[jp][kp][7]*Z8;
dXdE = PNE[ip][kp][0]*X1 + PNE[ip][kp][1]*X2
      + PNE[ip][kp][2]*X3 + PNE[ip][kp][3]*X4
      + PNE[ip][kp][4]*X5 + PNE[ip][kp][5]*X6
      + PNE[ip][kp][6]*X7 + PNE[ip][kp][7]*X8;

```

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

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

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

JACOBI (3/4)

```

dYdE = PNE[ip][kp][0]*Y1 + PNE[ip][kp][1]*Y2
        + PNE[ip][kp][2]*Y3 + PNE[ip][kp][3]*Y4
        + PNE[ip][kp][4]*Y5 + PNE[ip][kp][5]*Y6
        + PNE[ip][kp][6]*Y7 + PNE[ip][kp][7]*Y8;
dZdE = PNE[ip][kp][0]*Z1 + PNE[ip][kp][1]*Z2
        + PNE[ip][kp][2]*Z3 + PNE[ip][kp][3]*Z4
        + PNE[ip][kp][4]*Z5 + PNE[ip][kp][5]*Z6
        + PNE[ip][kp][6]*Z7 + PNE[ip][kp][7]*Z8;
dXdT = PNT[ip][jp][0]*X1 + PNT[ip][jp][1]*X2
        + PNT[ip][jp][2]*X3 + PNT[ip][jp][3]*X4
        + PNT[ip][jp][4]*X5 + PNT[ip][jp][5]*X6
        + PNT[ip][jp][6]*X7 + PNT[ip][jp][7]*X8;
dYdT = PNT[ip][jp][0]*Y1 + PNT[ip][jp][1]*Y2
        + PNT[ip][jp][2]*Y3 + PNT[ip][jp][3]*Y4
        + PNT[ip][jp][4]*Y5 + PNT[ip][jp][5]*Y6
        + PNT[ip][jp][6]*Y7 + PNT[ip][jp][7]*Y8;
dZdT = PNT[ip][jp][0]*Z1 + PNT[ip][jp][1]*Z2
        + PNT[ip][jp][2]*Z3 + PNT[ip][jp][3]*Z4
        + PNT[ip][jp][4]*Z5 + PNT[ip][jp][5]*Z6
        + PNT[ip][jp][6]*Z7 + PNT[ip][jp][7]*Z8;

```

```

DETJ[ip][jp][kp]= dXdQ*(dYdE*dZdT-dZdE*dYdT) +
                  dYdQ*(dZdE*dXdT-dXdE*dZdT) +
                  dZdQ*(dXdE*dYdT-dYdE*dXdT);

```

```

/**
INVERSE JACOBIAN
**/

```

```

coef=1.0 / 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]=fabs(DETJ[ip][jp][kp]);

```

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

自然座標系における偏微分 (4/4)

- 従って下記のように偏微分を計算できる
 - ヤコビアン (3×3行列) の逆行列を求める

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

JACOBI (3/4)

```

dYdE = PNE[ip][kp][0]*Y1 + PNE[ip][kp][1]*Y2
        + PNE[ip][kp][2]*Y3 + PNE[ip][kp][3]*Y4
        + PNE[ip][kp][4]*Y5 + PNE[ip][kp][5]*Y6
        + PNE[ip][kp][6]*Y7 + PNE[ip][kp][7]*Y8;
dZdE = PNE[ip][kp][0]*Z1 + PNE[ip][kp][1]*Z2
        + PNE[ip][kp][2]*Z3 + PNE[ip][kp][3]*Z4
        + PNE[ip][kp][4]*Z5 + PNE[ip][kp][5]*Z6
        + PNE[ip][kp][6]*Z7 + PNE[ip][kp][7]*Z8;
dXdT = PNT[ip][jp][0]*X1 + PNT[ip][jp][1]*X2
        + PNT[ip][jp][2]*X3 + PNT[ip][jp][3]*X4
        + PNT[ip][jp][4]*X5 + PNT[ip][jp][5]*X6
        + PNT[ip][jp][6]*X7 + PNT[ip][jp][7]*X8;
dYdT = PNT[ip][jp][0]*Y1 + PNT[ip][jp][1]*Y2
        + PNT[ip][jp][2]*Y3 + PNT[ip][jp][3]*Y4
        + PNT[ip][jp][4]*Y5 + PNT[ip][jp][5]*Y6
        + PNT[ip][jp][6]*Y7 + PNT[ip][jp][7]*Y8;
dZdT = PNT[ip][jp][0]*Z1 + PNT[ip][jp][1]*Z2
        + PNT[ip][jp][2]*Z3 + PNT[ip][jp][3]*Z4
        + PNT[ip][jp][4]*Z5 + PNT[ip][jp][5]*Z6
        + PNT[ip][jp][6]*Z7 + PNT[ip][jp][7]*Z8;
DETJ[ip][jp][kp]= dXdQ*(dYdE*dZdT-dZdE*dYdT) +
                  dYdQ*(dZdE*dXdT-dXdE*dZdT) +
                  dZdQ*(dXdE*dYdT-dYdE*dXdT);

```

```

/**
INVERSE JACOBIAN
**/

```

```
coef=1.0 / 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]=fabs(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)

```

/**
 set the dNi/dX, dNi/dY & dNi/dZ components
**/
PNX[ip][jp][kp][0]=a11*PNQ[jp][kp][0]+a12*PNE[ip][kp][0]+a13*PNT[ip][jp][0];
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];
PNY[ip][jp][kp][0]=a21*PNQ[jp][kp][0]+a22*PNE[ip][kp][0]+a23*PNT[ip][jp][0];
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];
PNZ[ip][jp][kp][0]=a31*PNQ[jp][kp][0]+a32*PNE[ip][kp][0]+a33*PNT[ip][jp][0];
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];
    }
}
}

```

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

係数行列 : MAT_ASS_MAIN (5/6)

```

/**
CONSTRUCT the GLOBAL MATRIX
**/

for (ie=0; ie<8; ie++) {
  ip=nodLOCAL[ie];

  for (je=0; je<8; je++) {
    jp=nodLOCAL[je];

    kk=-1;
    if ( jp != ip ) {
      iiS=indexLU[ip-1];
      iiE=indexLU[ip ];
      for ( k=iiS; k<iiE; k++) {
        if ( itemLU[k] == jp-1 ) {
          kk=k;
          break;
        }
      }
    }
  }
}

```

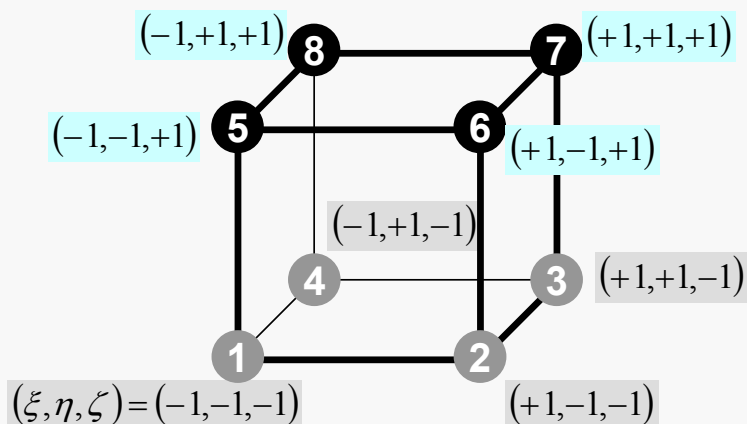
全体行列の非対角成分

$$A_{ip, jp}$$

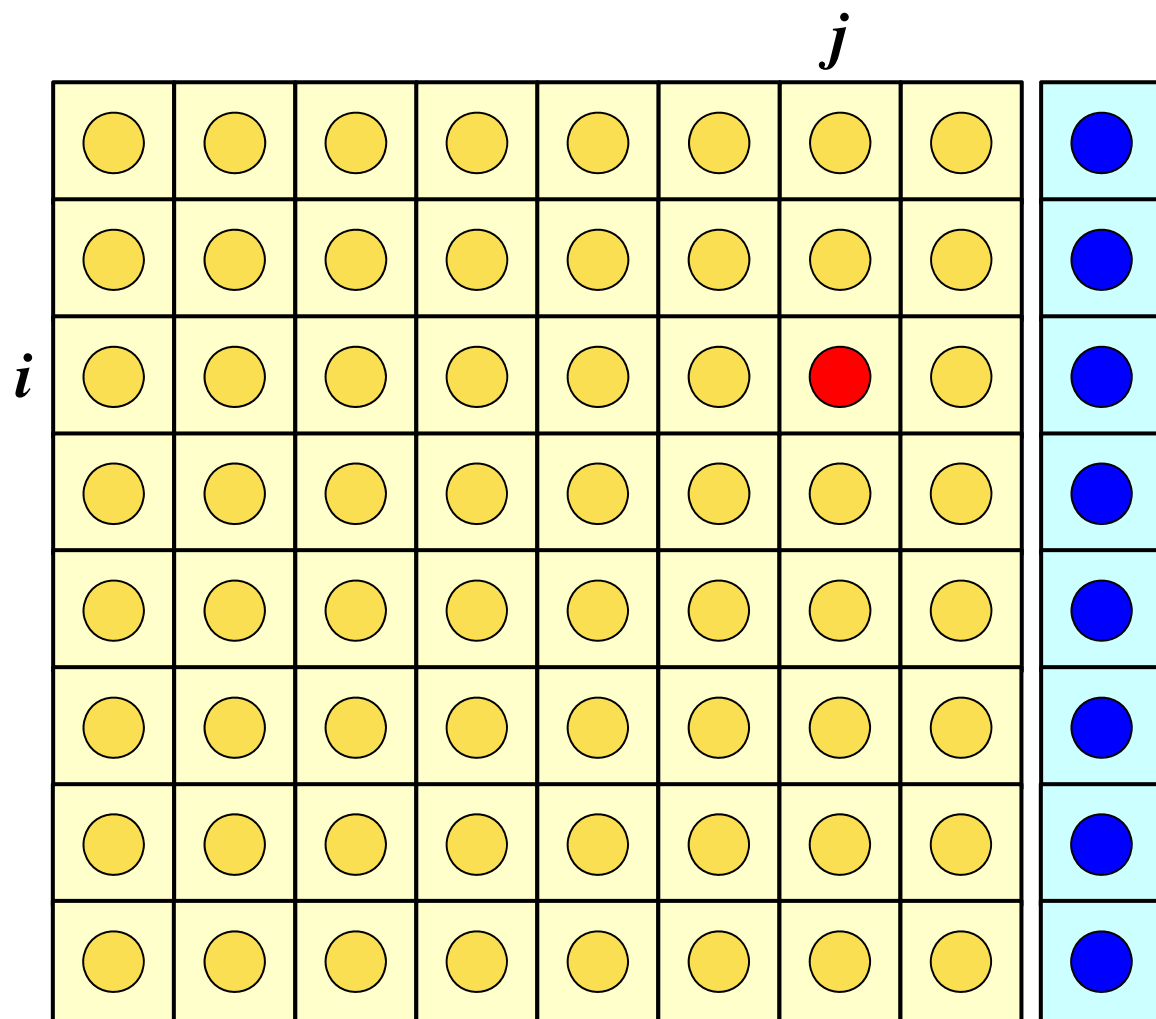
kk: itemにおけるアドレス

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

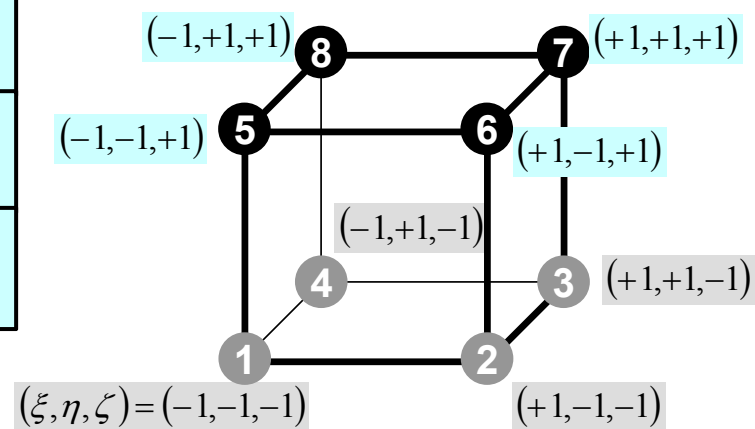
1から始まる節点番号



要素マトリクス : 8×8 行列



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



係数行列 : MAT_ASS_MAIN (5/6)

```

/**
CONSTRUCT the GLOBAL MATRIX
**/

for (ie=0; ie<8; ie++) {
  ip=nodLOCAL[ie];

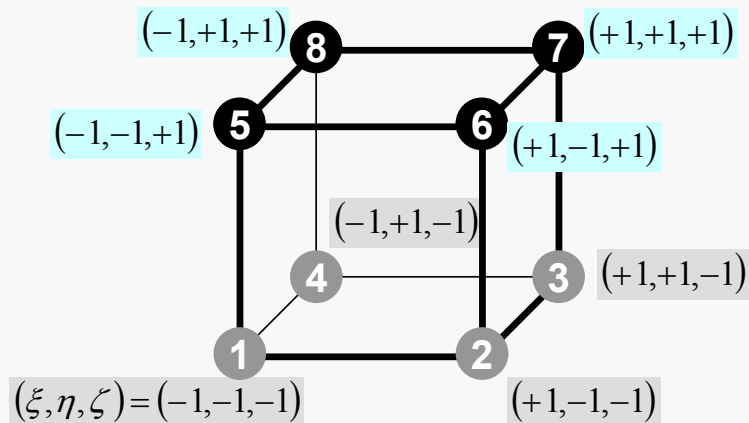
  for (je=0; je<8; je++) {
    jp=nodLOCAL[je];

    kk=-1;
    if ( jp != ip ) {
      iiS=indexLU[ip-1];
      iiE=indexLU[ip ];
      for ( k=iiS; k<iiE; k++) {
        if ( itemLU[k] == jp-1 ) {
          kk=k;
          break;
        }
      }
    }
  }
}

```

要素マトリクス ($i_e \sim j_e$)
 全体マトリクス ($i_p \sim j_p$) の関係

kk: itemLUにおけるアドレス



系数行列：MAT_ASS_MAIN (6/6)

```

QV0= 0. e0;
COEFij= 0. e0;

for (kpn=0;kpn<2;kpn++) {
  for (jpn=0;jpn<2;jpn++) {
    for (ipn=0;ipn<2;ipn++) {
      coef= fabs (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+= coef*CONDO*(PNXi*PNXj+PNYi*PNYj+PNZi*PNZj);

      SHi= SHAPE[ipn][jpn][kpn][ie];
      QV0+= SHi * QVOL * coef;
    }
  }
}

if (jp==ip) {
  D[ip-1]+= COEFij;
  B[ip-1]+= QV0*QVC;
}
if (jp != ip) {
  AMAT[kk]+= COEFij;
}
}
}
}
}

```

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

系数行列：MAT_ASS_MAIN (6/6)

```

QV0= 0. e0;
COEFij= 0. e0;

for (kpn=0;kpn<2;kpn++) {
  for (jpn=0;jpn<2;jpn++) {
    for (ipn=0;ipn<2;ipn++) {
      coef= fabs(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+= coef*CONDO*(PNXi*PNXj+PNYi*PNYj+PNZi*PNZj);

```

```

SHi= SHAPE[ipn][jpn][kpn][ie];
QV0+= SHi * QVOL * coef;

```

```

    }
  }
}

if (jp==ip) {
  D[ip-1]+= COEFij;
  B[ip-1]+= QV0*QVC;
}
if (jp != ip) {
  AMAT[kk]+= COEFij;
}
}
}
}

```

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

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

系数行列：MAT_ASS_MAIN (6/6)

```

QV0= 0. e0;
COEFij= 0. e0;

for (kpn=0; kpn<2; kpn++) {
  for (jpn=0; jpn<2; jpn++) {
    for (ipn=0; ipn<2; ipn++) {
      coef= fabs(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];

```

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

```

PNXj= PNx[ipn][jpn][kpn][je];
PNYj= PNY[ipn][jpn][kpn][je];
PNZj= PNz[ipn][jpn][kpn][je];

```

```

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

```

```

SHi= SHAPE[ipn][jpn][kpn][ie];
QV0+= SHi * QVOL * coef;

```

```

}
}
}
if (jp==ip) {
  D[ip-1]+= COEFij;
  B[ip-1]+= QV0*QVC;
}
if (jp != ip) {
  AMAT[kk]+= COEFij;
}
}
}
}

```

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

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

系数行列：MAT_ASS_MAIN (6/6)

```

QV0= 0. e0;
COEFij= 0. e0;

for (kpn=0; kpn<2; kpn++) {
  for (jpn=0; jpn<2; jpn++) {
    for (ipn=0; ipn<2; ipn++) {
      coef= fabs(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+= coef*CONDO*(PNXi*PNXj+PNYi*PNYj+PNZi*PNZj);

      SHi= SHAPE[ipn][jpn][kpn][ie];
      QV0+= SHi * QVOL * coef;
    }
  }
}

if (jp==ip) {
  D[ip-1]+= COEFij;
  B[ip-1]+= QV0*QVC;
}
if (jp != ip) {
  AMAT[kk]+= COEFij;
}
}
}
}

```

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

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

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

$$QVC = |x_c + y_c|$$

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

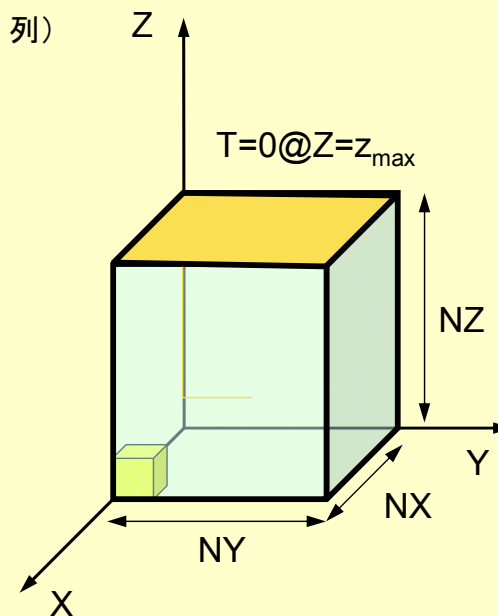
$$\{f\}^{(e)} = QV0 \cdot QVC$$

MAT_ASS_BC : 全体構成

```
do i= 1, N    節点ループ
  (ディリクレ) 境界条件を設定する節点をマーク (IWKX)
enddo
```

```
do i= 1, N  節点ループ
  if (IWKX(i,1).eq.1) then  マークされた節点だったら
    対応する右辺ベクトル (B) の成分, 対角成分 (D) の成分の修正 (行・列)
    do k= index(i-1)+1, index(i)
      対応する非零非対角成分 (AMAT) の成分の修正 (行)
    enddo
  endif
enddo
```

```
do i= 1, N  節点ループ
  do k= indexLU (i-1)+1, index (i)
    if (IWKX(item (k),1).eq.1) then  対応する非零非対角成分の
      節点がマークされていたら
      対応する右辺ベクトル, 非零非対角成分 (AMAT) の成分の修正 (列)
    endif
  enddo
enddo
```



境界条件 : MAT_ASS_BC (1/2)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE *fp_log;
void MAT_ASS_BC()
{
    int i, j, k, in, ib, ib0, icel;
    int in1, in2, in3, in4, in5, in6, in7, in8;
    int iq1, iq2, iq3, iq4, iq5, iq6, iq7, iq8;
    int iS, iE;
    double STRESS, VAL;

    IWKX=(KINT**) allocate_matrix(sizeof(KINT), N, 2);
    for(i=0; i<N; i++) for(j=0; j<2; j++) IWKX[i][j]=0;

    /**
     * Z=Zmax
     */

    for(in=0; in<N; in++) IWKX[in][0]=0;

    ib0=-1;

    for( ib0=0; ib0<NODGRPtot; ib0++) {
        if( strcmp(NODGRP_NAME[ib0].name, "Zmax") == 0 ) break;
    }

    for( ib=NODGRP_INDEX[ib0]; ib<NODGRP_INDEX[ib0+1]; ib++) {
        in=NODGRP_ITEM[ib];
        IWKX[in-1][0]=1;
    }
}
```

節点グループ名が「Zmax」である
節点in(1から始まる)において:

$$IWKX[in-1][0] = 1$$

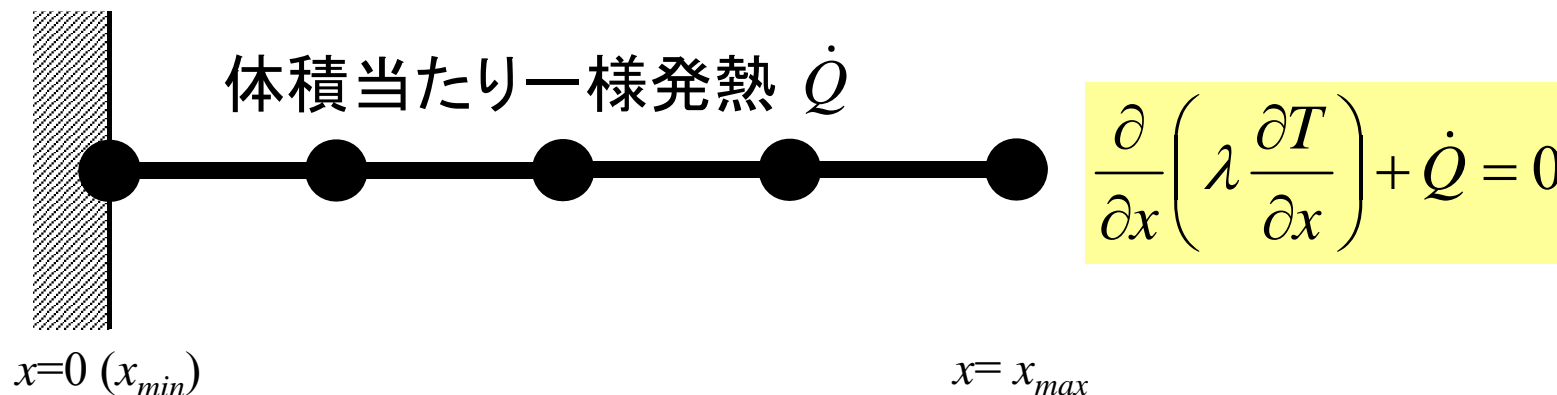
とする

境界条件 : MAT_ASS_BC (2/2)

```
for(in=0; in<N; in++){
  if( IWKX[in][0] == 1 ){
    B[in]= 0. e0;
    D[in]= 1. e0;
    for(k=indexLU[in]; k<indexLU[in+1]; k++){
      AMAT[k]= 0. e0;
    }
  }
}

for(in=0; in<N; in++){
  for(k=indexLU[in]; k<indexLU[in+1]; k++){
    if (IWKX[itemLU[k]][0] == 1 ) {
      AMAT[k]= 0. e0;
    }
  }
}
}
```

対象とする問題：一次元熱伝導問題

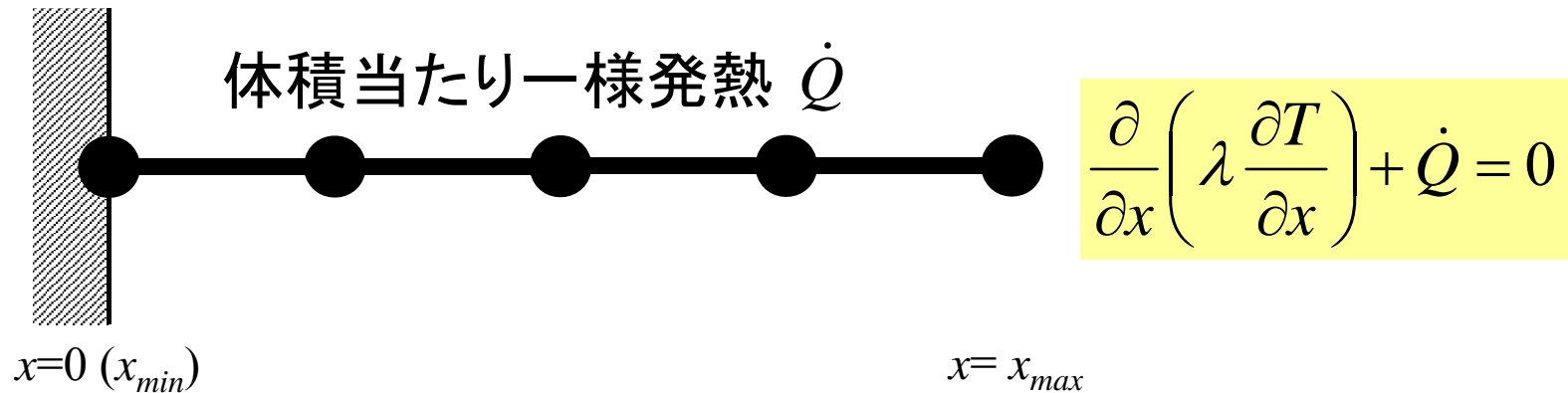


- 一様な：断面積 A ，熱伝導率 λ
- 体積当たり一様発熱（時間当たり） $[QL^{-3}T^{-1}]$ \dot{Q}
- 境界条件
 - $x=0$: $T=0$ （固定）
 -
 - $x=x_{max}$: $\frac{\partial T}{\partial x} = 0$ （断熱）

復習：一次元問題

x=0で成立する方程式

T₀=0



- 一様な：断面積 A ，熱伝導率 λ
- 体積当たり一様発熱（時間当たり） $[QL^{-3}T^{-1}]$ \dot{Q}
- 境界条件
 - $x=0$: $T=0$ （固定）
 -
 - $x=x_{max}$: $\frac{\partial T}{\partial x} = 0$ （断熱）

復習：一次元問題

プログラム: 1d.c (6/6)

第一種境界条件 @x=0

```

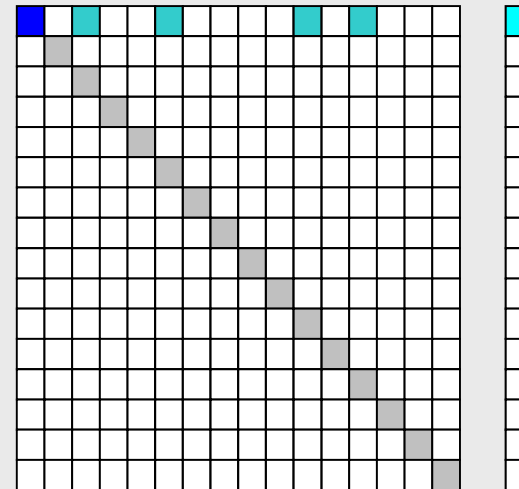
/*
// +-----+
// | BOUNDARY conditions |
// +-----+
*/

/* X=Xmin */
  i=0;
  jS= Index[i];
  AMat[jS]= 0.0;
  Diag[i ]= 1.0;
  Rhs [i ]= 0.0;

  for (k=0;k<NPLU;k++) {
    if (Item[k]==0) {AMat[k]=0.0;
    }}

```

$T_0=0$
 対角成分=1, 右辺=0, 非対角成分=0



復習: 一次元問題

プログラム: 1d.c (6/6)

第一種境界条件 @x=0

```

/*
// +-----+
// | BOUNDARY conditions |
// +-----+
*/

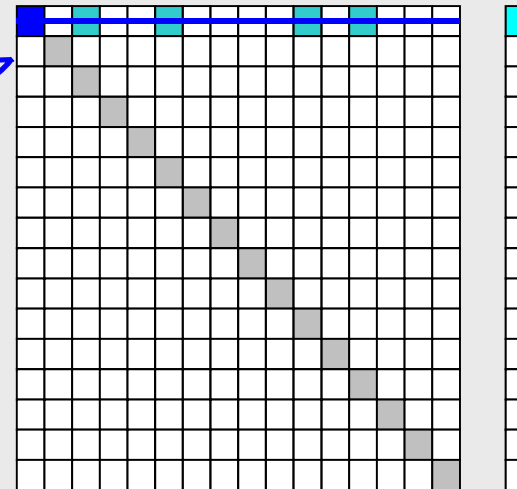
/* X=Xmin */
i=0;
jS= Index[i];
AMat[jS]= 0.0;
Diag[i ]= 1.0;
Rhs [i ]= 0.0;

for (k=0;k<NPLU;k++) {
  if (Item[k]==0) {AMat[k]=0.0;
  }}

```

$T_0=0$
 対角成分=1, 右辺=0, 非対角成分=0

ゼロクリア



復習: 一次元問題

プログラム: 1d.c (6/6)

第一種境界条件 @x=0

```

/*
// +-----+
// | BOUNDARY conditions |
// +-----+
*/

/* X=Xmin */
i=0;
jS= Index[i];
AMat[jS]= 0.0;
Diag[i ]= 1.0;
Rhs [i ]= 0.0;

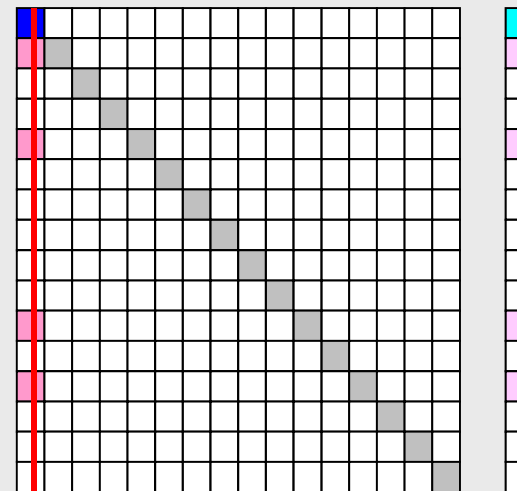
for (k=0;k<NPLU;k++) {
  if (Item[k]==0) {AMat[k]=0.0;
  }}

```

行列の対称性を保つため、第一種境界条件を適用している節点に対応する「列」を、右辺に移項して消去する(今の場合は非対角成分を0にするだけで良い)

$T_0=0$
対角成分=1, 右辺=0, 非対角成分=0

消去, ゼロクリア



復習: 一次元問題

第一種境界条件が $T \neq 0$ の場合

```

/*
// +-----+
// | BOUNDARY conditions |
// +-----+
*/

```

行列の対称性を保つため、第一種境界条件を適用している節点に対応する「列」を、右辺に移項して消去する

```

/* X=Xmin */

```

```

    i=0;
    jS= Index[i];
    AMat[jS]= 0.0;
    Diag[i ]= 1.0;
    Rhs [i ]= PHImin;

```

```

    for (j=1; i<N; i++) {
        for (k=Index[j]; k<Index[j+1]; k++) {
            if (Item[k]==0) {
                Rhs [j]= Rhs[j] - AMat[k]*PHImin;
                AMat[k]= 0.0;
            }
        }
    }

```

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

第一種境界条件が $T \neq 0$ の場合

```

/*
// +-----+
// | BOUNDARY conditions |
// +-----+
*/

```

行列の対称性を保つため、第一種境界条件を適用している節点に対応する「列」を、右辺に移項して消去する

```

/* X=Xmin */
  i=0;
  jS= Index[i];
  AMat[jS]= 0.0;
  Diag[i ]= 1.0;
  Rhs [i ]= PHImin;

```

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

```

for (j=1; i<N; i++) {
  for (k=Index[j]; k<Index[j+1]; k++) {
    if (Item[k]==0) {
      Rhs [j]= Rhs[j] - Amat[k]*PHImin;
      AMat[k]= 0.0;
    }
  }
}

```

境界条件 : MAT_ASS_BC (2/2)

```

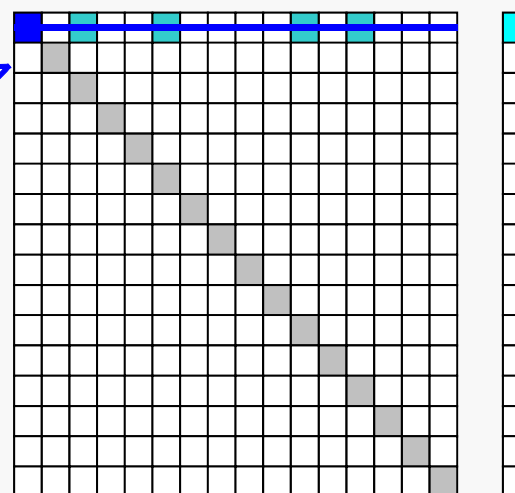
for (in=0; in<N; in++) {
  if ( IWKX[in][0] == 1 ) {
    B[in]= 0. e0;
    D[in]= 1. e0;
    for (k=indexLU[in]; k<indexLU[in+1]; k++) {
      AMAT[k]= 0. e0;
    }
  }
}

for (in=0; in<N; in++) {
  for (k=indexLU[in]; k<indexLU[in+1]; k++) {
    if ( IWKX[itemLU[k]][0] == 1 ) {
      AMAT[k]= 0. e0;
    }
  }
}

```

IWKX[in-1][0]=1となる節点に対して
対角成分=1, 右辺=0, 非零対角成分=0

ゼロクリア



ここでやっていることも一次元の時と
全く変わらない

境界条件 : MAT_ASS_BC (2/2)

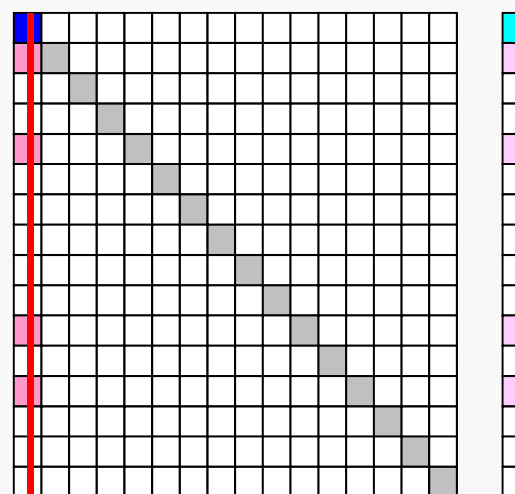
```

for(in=0; in<N; in++) {
  if( IWKX[in][0] == 1 ) {
    B[in]= 0. e0;
    D[in]= 1. e0;
    for(k=indexLU[in]; k<indexLU[in+1]; k++) {
      AMAT[k]= 0. e0;
    }
  }
}

for(in=0; in<N; in++) {
  for(k=indexLU[in]; k<indexLU[in+1]; k++) {
    if( IWKX[itemLU[k]][0] == 1 ) {
      AMAT[k]= 0. e0;
    }
  }
}

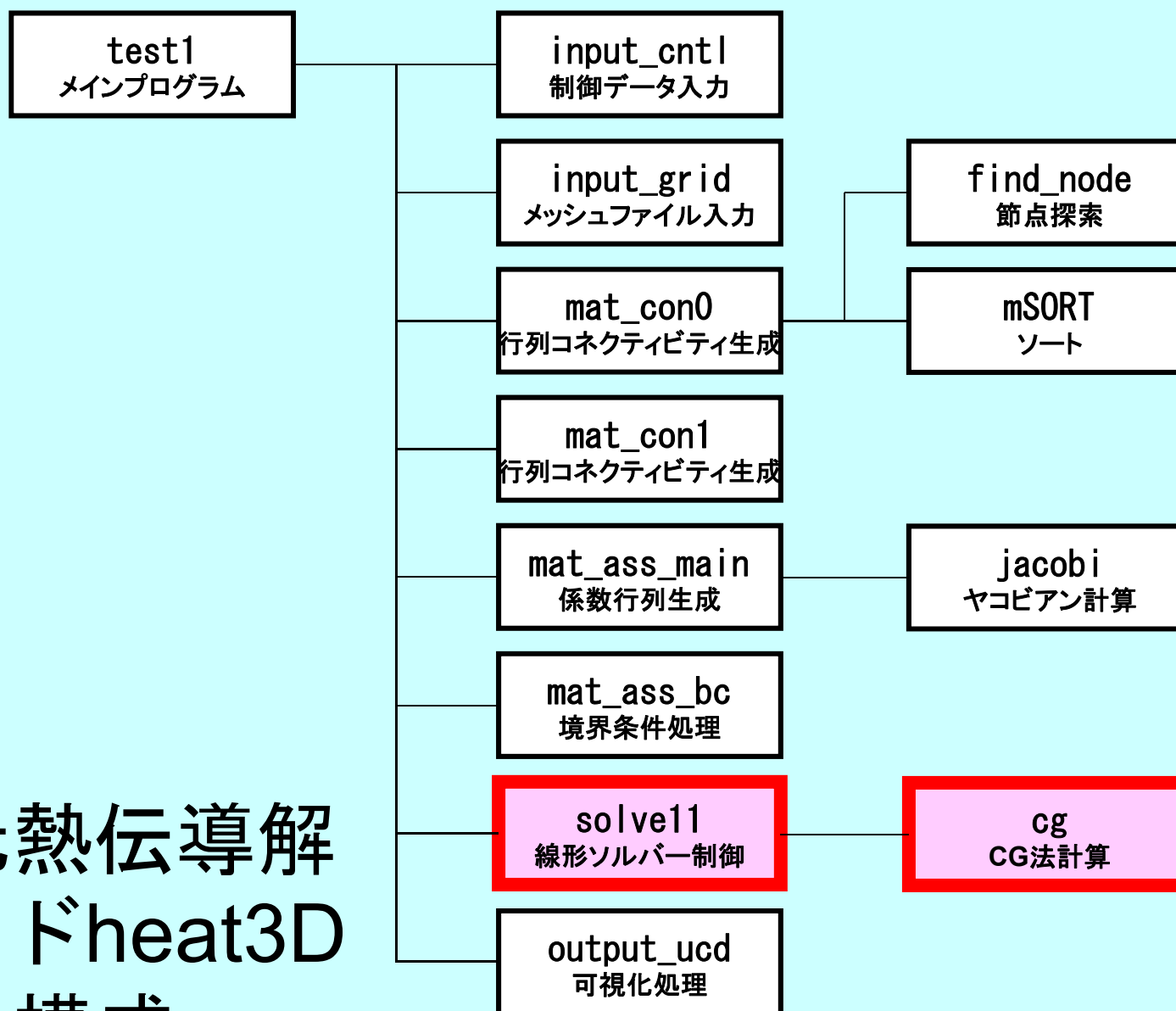
```

IWKX[in-1][0]=1となる節点を非零非対角成分として有している節点に対して、右辺へ移項，当該非零非対角成分=0



消去，ゼロクリア

ここでやっていることも一次元の時と全く変わらない



三次元熱伝導解析コードheat3Dの構成

全体処理

```
/**
    program heat3D
**/
#include <stdio.h>
#include <stdlib.h>
FILE* fp_log;
#define GLOBAL_VALUE_DEFINE
#include "pfem_util.h"
// #include "solver11.h"
extern void INPUT_CNTL ();
extern void INPUT_GRID ();
extern void MAT_CONO ();
extern void MAT_CON1 ();
extern void MAT_ASS_MAIN ();
extern void MAT_ASS_BC ();
extern void SOLVE11 ();
extern void OUTPUT_UCD ();
int main()
{
    INPUT_CNTL ();
    INPUT_GRID ();

    MAT_CONO ();
    MAT_CON1 ();

    MAT_ASS_MAIN ();
    MAT_ASS_BC ();

    SOLVE11 ();

    OUTPUT_UCD ();
}
```

SOLVE11

```

#include <stdio.h>
#include <string.h>
#include <math.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE *fp_log;
extern void CG();
void SOLVE11()
{
    int i, j, k, ii, L;

    int ERROR, ICFLAG=0;
    CHAR_LENGTH BUF;

    /**
     +-----+
     | PARAMETERS |
     +-----+
    **/
    ITER      = pfemIarray[0];      CG法の最大反復回数
    RESID     = pfemRarray[0];      CG法の収束判定値

    /**
     +-----+
     | ITERATIVE solver |
     +-----+
    **/
    CG (N, NPLU, D, AMAT, indexLU, itemLU, B, X, RESID, ITER, &ERROR);
    ITERactual= ITER;
}

```

前処理付き共役勾配法

Preconditioned Conjugate Gradient Method (CG)

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

前処理: 対角スケーリング

対角スケーリング, 点ヤコビ前処理

- 前処理行列として, もとの行列の対角成分のみを取り出した行列を前処理行列 $[M]$ とする。
 - 対角スケーリング, 点ヤコビ (point-Jacobi) 前処理

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

- **solve $[M]z^{(i-1)} = r^{(i-1)}$** という場合に逆行列を簡単に求めることができる。

CGソルバー(1/6)

```
#include <stdio.h>
#include <math.h>
#include "precision.h"
#include "allocate.h"
extern FILE *fp_log;

void CG (
    KINT N, KINT NPLU, KREAL D[],
    KREAL AMAT[], KINT indexLU[], KINT itemLU[],
    KREAL B[], KREAL X[], KREAL RESID, KINT ITER, KINT *ERROR)
{
    int i, j, k;
    int ieL, isL, ieU, isU;
    double WVAL;
    double BNRM20, BNRM2, DNRM20, DNRM2;
    double S1_TIME, E1_TIME;
    double ALPHA, BETA;
    double C1, C10, RH0, RH00, RH01;
    int iterPRE;

    KREAL **WW;

    KINT R=0, Z=1, Q=1, P=2, DD=3;
    KINT MAXIT;
    KREAL TOL;
```

CGソルバー(1/6)

```

#include <stdio.h>
#include <math.h>
# WW[i][0]= WW[i][R] => {r}
# WW[i][1]= WW[i][Z] => {z}
e WW[i][1]= WW[i][Q] => {q}
v WW[i][2]= WW[i][P] => {p}
WW[i][3]= WW[i][DD] => 1/{D}
{
  int i, j, k;
  int ieL, isL, ieU, isU;
  double WVAL;
  double BNRM20, BNRM2, DNRM20, DNRM2;
  double S1_TIME, E1_TIME;
  double ALPHA, BETA;
  double C1, C10, RH0, RH00, RH01;
  int iterPRE;

  KREAL **WW;

  KINT R=0, Z=1, Q=1, P=2, DD=3;
  KINT MAXIT;
  KREAL TOL;

```

```

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ソルバー (2/6)

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

```
MAXIT = ITER;
```

```
TOL = RESID;
```

```
for (i=0; i<N; i++) {
    X[i]=0.0;
}
```

```
for (i=0; i<N; i++) for (j=0; j<4; j++) WW[j][i]=0.0;
```

```
/**
```

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

```
**/
```

```
for (j=0; j<N; j++) {
```

```
    WW[DD][j]= 1.0/D[j];
```

```
    WVAL= B[j] - D[j]*X[j];
```

```
    for ( k=indexLU[j]; k<indexLU[j+1]; k++) {
```

```
        i= itemLU[k];
```

```
        WVAL+= -AMAT[k]*X[i];
```

```
    }
```

```
    WW[R][j]= WVAL;
```

```
}
```

$$WW[i][0] = WW[i][R] \Rightarrow \{r\}$$

$$WW[i][1] = WW[i][Z] \Rightarrow \{z\}$$

$$WW[i][1] = WW[i][Q] \Rightarrow \{q\}$$

$$WW[i][2] = WW[i][P] \Rightarrow \{p\}$$

$$WW[i][3] = WW[i][DD] \Rightarrow 1/\{D\}$$

対角成分の逆数(前処理用)

その都度, 除算をすると効率が
悪いため, 予め配列に格納

CGソルバー (2/6)

```

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

MAXIT = ITER;
TOL   = RESID;

for (i=0; i<N; i++) {
    X[i]=0.0;
}
for (i=0; i<N; i++) for (j=0; j<4; j++) WW[j][i]=0
/**
+-----+
| {r0} = {b} - [A]{xini} |
+-----+
**/
for (j=0; j<N; j++) {
    WW[DD][j] = 1.0/D[j];
    WVAL = B[j] - D[j]*X[j];

    for (k=indexLU[j]; k<indexLU[j+1]; k++) {
        i = itemLU[k];
        WVAL += -AMAT[k]*X[i];
    }
    WW[R][j] = WVAL;
}

```

```

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ソルバー (3/6)

```
BNRM2= 0. e0;  
for (i=0; i<N; i++) {  
    BNRM2+= B[i]*B[i];  
}
```

```
BNRM2= BNRM2;
```

```
if (BNRM2 == 0. e0) BNRM2= 1. e0;
```

```
ITER = 0;
```

```
for ( ITER=1; ITER<= MAXIT; ITER++) {
```

```
/**
```

```
***** Conjugate Gradient Iteration
```

```
**/
```

```
/**
```

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

```
**/
```

```
for (i=0; i<N; i++) {  
    WW[Z][i] = WW[DD][i]*WW[R][i];  
}
```

$BNRM2 = |b|^2$
あとで収束判定に使用

CGソルバー (3/6)

```

    BNRM20= 0. e0;
  for (i=0; i<N; i++) {
    BNRM20+= B[i]*B[i];
  }

  BNRM2= BNRM20;

  if (BNRM2 == 0. e0) BNRM2= 1. e0;

  ITER = 0;

  for ( ITER=1; ITER<= MAXIT; ITER++) {
/**
*****
**/

/**
+-----+
| {z}= [Minv] {r} |
+-----+
**/
    for (i=0; i<N; i++) {
      WW[Z][i]= WW[DD][i]*WW[R][i];
    }

```

```

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

```

CGソルバー(4/6)

```

/**
+-----+
| {RHO}= {r} {z} |
+-----+
**/
RH00= 0. e0;

for(i=0;i<N;i++){
  RH00+= WW[R][i]*WW[Z][i];
}
RHO= RH00;
/**
+-----+
| {p} = {z} if      ITER=1 |
| BETA= RHO / RH01 otherwise |
+-----+
**/
if( ITER == 1 ){
  for(i=0;i<N;i++){
    WW[P][i]=WW[Z][i];
  }
}else{
  BETA= RHO / RH01;
  for(i=0;i<N;i++){
    WW[P][i]=WW[Z][i] + BETA*WW[P][i];
  }
}

```

```

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

```

CGソルバー (5/6)

```

/**
+-----+
| {q} = [A] {p} |
+-----+
**/
for ( j=0; j<N; j++) {
    WVAL= D[j] * WW[P][j];
    for (k=indexLU[j]; k<indexLU[j+1]; k++) {
        i=itemLU[k];
        WVAL+= AMAT[k] * WW[P][i];
    }
    WW[Q][j]=WVAL;
}

/**
+-----+
| ALPHA= RHO / {p} {q} |
+-----+
**/
C10= 0. e0;
for (i=0; i<N; i++) {
    C10+=WW[P][i]*WW[Q][i];
}
C1=C10;

ALPHA= RHO / C1;

```

```

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

```

CGソルバー (6/6)

```

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

DNRM2= 0. e0;
for (i=0; i<N; i++) {
    DNRM2+=WW[R][i]*WW[R][i];
}
DNRM2= DNRM2;
RESID= sqrt(DNRM2/BNRM2);

if ( RESID <= TOL ) break;
if ( ITER == MAXIT ) *ERROR= -300;

RH01 = RHO ;
}

```

```

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

```

CGソルバー (6/6)

```

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

DNRM2= 0. e0;
for (i=0; i<N; i++) {
  DNRM2+=WW[R][i]*WW[R][i];
}
DNRM2= DNRM2;
RESID= sqrt(DNRM2/BNRM2);

if ( RESID <= TOL ) break;
if ( ITER == MAXIT ) *ERROR= -300;

RH01 = RHO ;
}

```

```

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

```

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