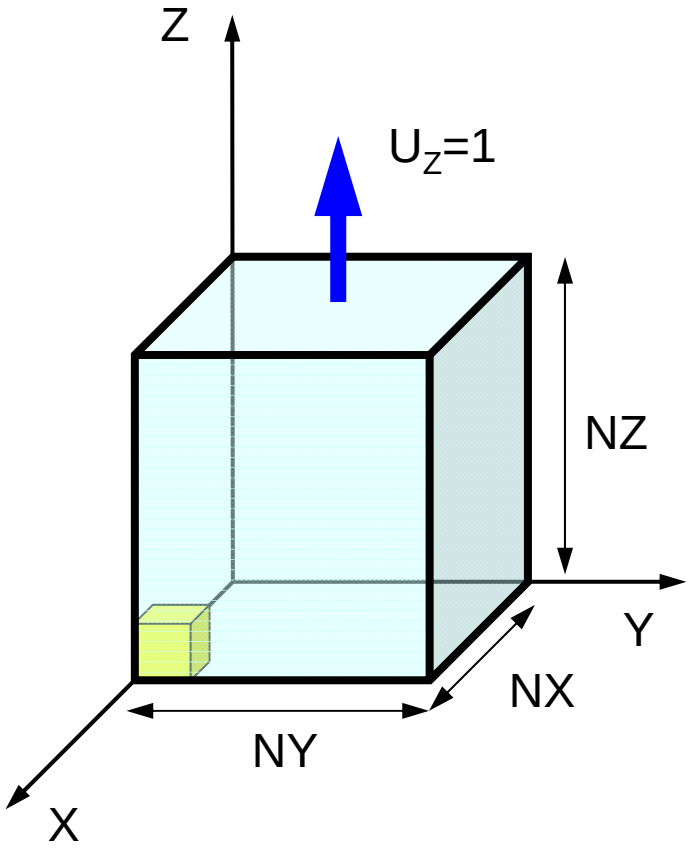
三次元弾性解析コード（1/3）
プログラムの概要

2012年夏学期

中島 研吾

科学技術計算Ｉ（4820-1027）・コンピュータ科学特別講義Ｉ（4810-1204）

対象とする問題



- 弾性体
  - ヤング率　　E
  - ポアソン比　　ν
- 直方体
  - 一辺長さ1の立方体（六面体）要素
  - 各方向にNX・NY・NZ個
- 境界条件
  - 対称条件
    - U_X=0@X=0
    - U_Y=0@Y=0
    - U_Z=0@Z=0
  - 強制変位
    - U_Z=1@Z=Z_{max}

movie

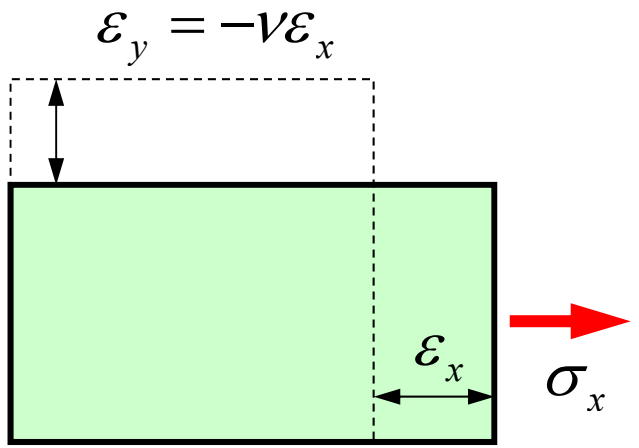
構成式：応力～ひずみ関係

- ヤング率 E
  - 応力とひずみは比例
  - 比例定数をヤング率Eとする（各物質に固有の値）

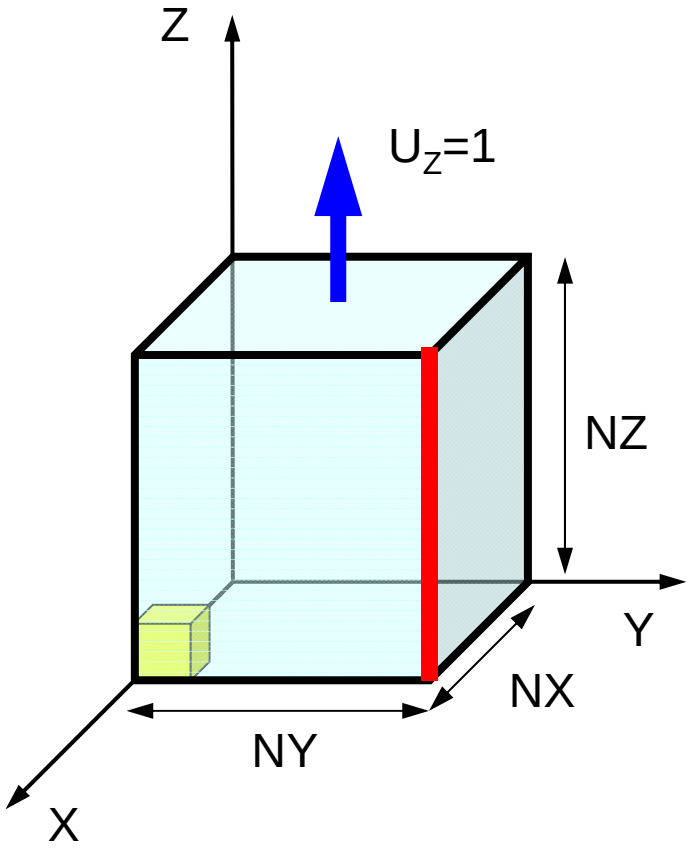
$$\sigma_x = E\varepsilon_x, \quad \varepsilon_x = \frac{\sigma_x}{E}$$

- ポアソン比 ν
  - X方向に荷重をかけると，横方向（Y,Z）にも変形
  - 縮み割合をポアソン比νとする
    - 各物質に固有の値
      - 金属では0.30程度
      - 水：0.50，ゴム：ほぼ0.50⇒非圧縮

$$\varepsilon_y = -\nu\varepsilon_x = -\nu\frac{\sigma_x}{E}$$



NX=NY=NZ=10, ν=0.30とすると



$$\varepsilon_z = \frac{1}{10} = 0.10$$

$$\varepsilon_x = \varepsilon_y = -\nu\varepsilon_z = -0.03$$

$$\therefore u_x\big|_{X=10,Y=10} = u_y\big|_{X=10,Y=10} = 10 \times \varepsilon_x = -0.30$$

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

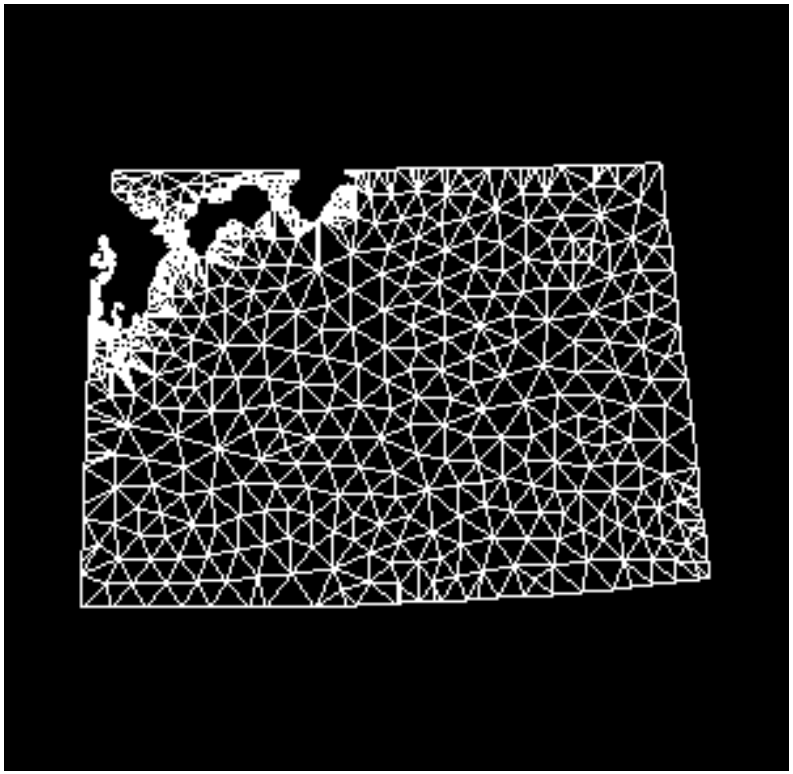
- 初期化
  - 制御変数読み込み
  - 座標読み込み⇒要素生成（N:節点数，ICELTOT：要素数）
  - 配列初期化（全体マトリクス，要素マトリクス）
  - 要素⇒全体マトリクスマッピング（Index，Item）
- マトリクス生成
  - 要素単位の処理（do icel= 1, ICELTOT）
    - 要素マトリクス計算
    - 全体マトリクスへの重ね合わせ
  - 境界条件の処理
- 連立一次方程式
  - 共役勾配法（CG）
- 応力計算

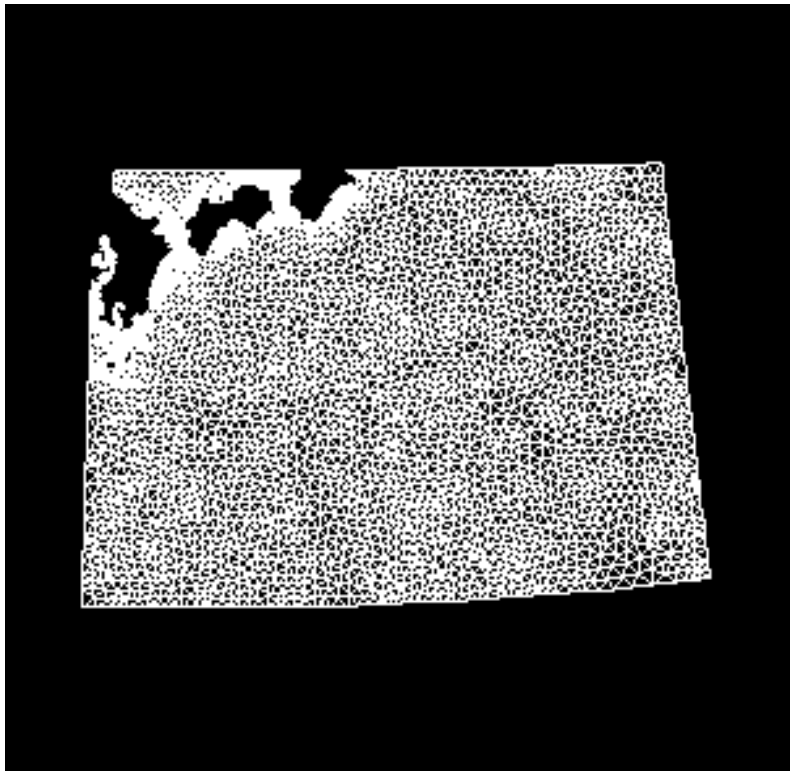
- 三次元要素の定式化
- 三次元弾性力学方程式
  - ガラーキン法
  - 要素マトリクス生成
- 宿題

- プログラムの実行
- データ構造
- プログラムの構成
- 計算効率について
- ParaViewによる可視化

二次元への拡張：三角形要素

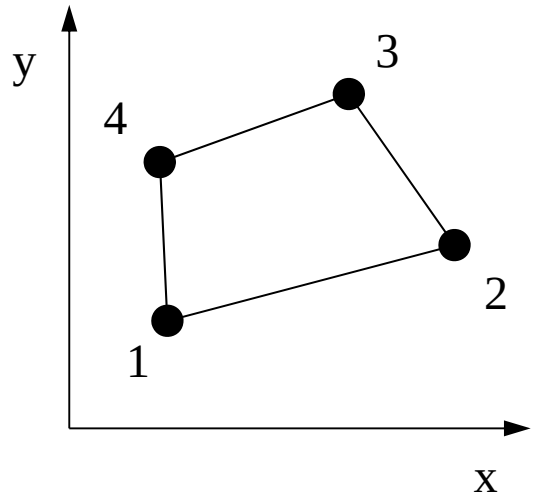
- 任意の形状を扱うことができる。
- 特に一次要素は精度が悪く，一部の問題を除いてあまり使用されない。





二次元への拡張：四角形要素

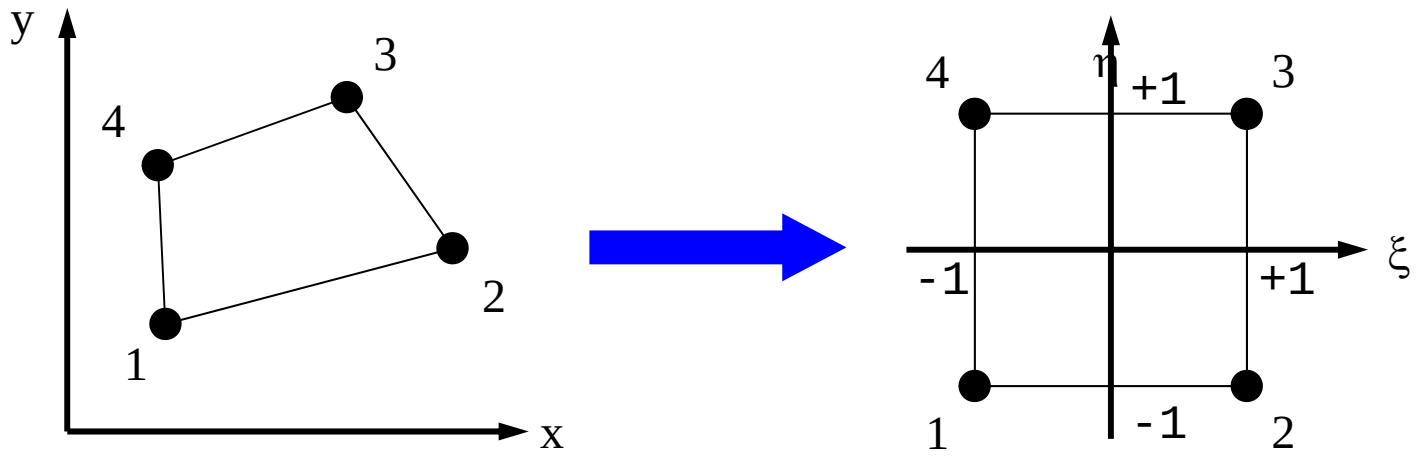
- 一次元要素と同じ形状関数をx,y軸に適用することによって，四角形要素の定式化は可能である。
  - 三角形と比較して特に低次要素の精度はよい
- しかしながら，各辺が座標軸に平行な長方形でなければならない
  - 差分法と変わらない



- このような形状を扱うことができない。

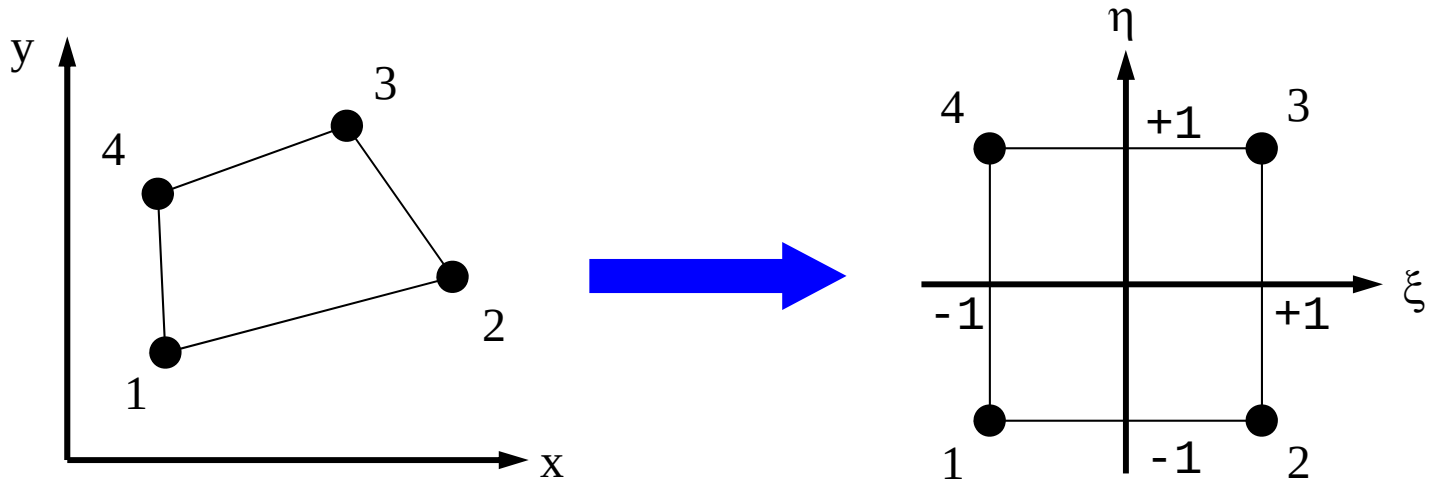
アイソパラメトリック要素（1/3）

- 各要素を，自然座標系（ξ,η）の正方形要素[±1, ±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）

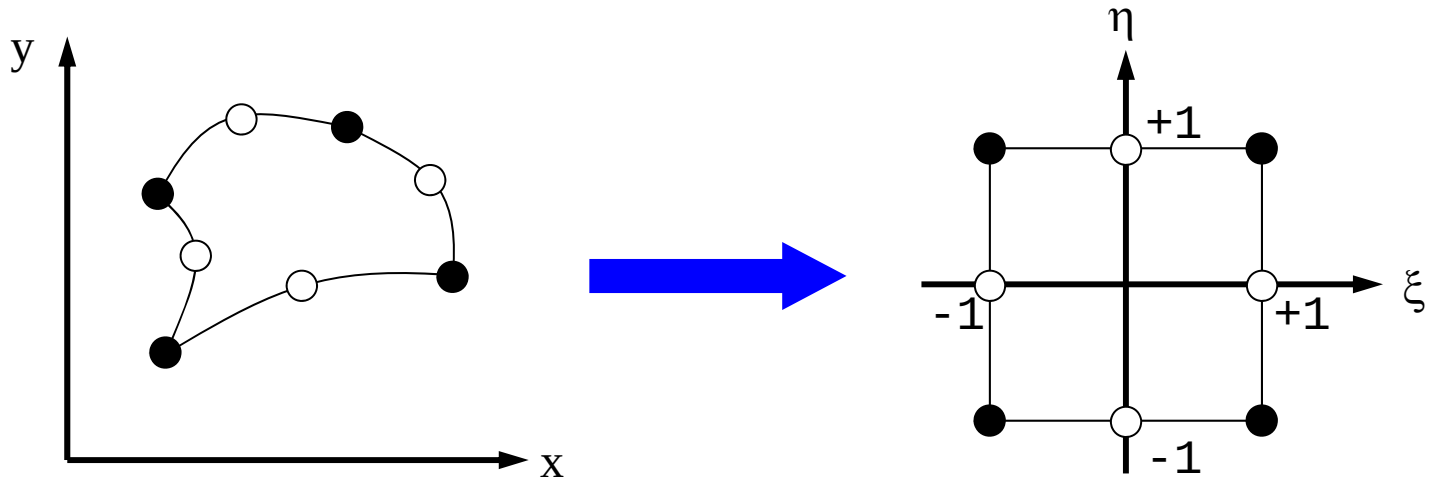


- 各節点の座標：(x_1,y_1), (x_2,y_2), (x_3,y_3), (x_4,y_4)
- 各節点におけるX方向変位：u_1, u_2, u_3, u_4

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

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

アイソパラメトリック要素（3/3）

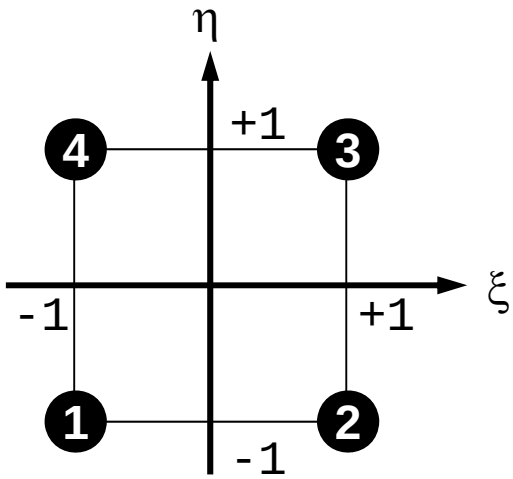


- 高次の補間関数を使えば，曲線，曲面も扱うことが可能となる。
- そういう意味で「自然座標系」と呼んでいる。

2D自然座標系の形状関数（1/2）

- 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：

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



- 各節点での条件より：

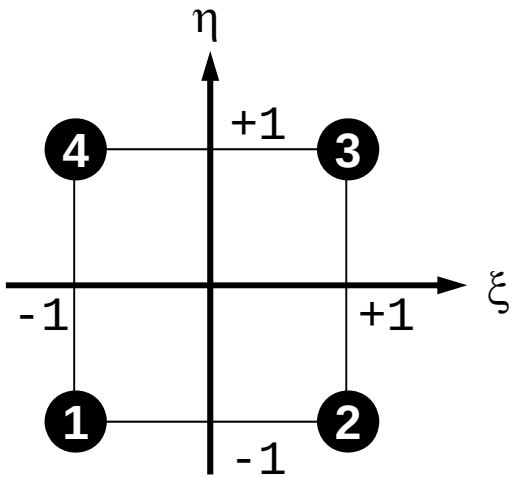
$$\alpha_1 = \frac{u_1 + u_2 + u_3 + u_4}{4}, \quad \alpha_2 = \frac{-u_1 + u_2 + u_3 - u_4}{4},$$

$$\alpha_3 = \frac{-u_1 - u_2 + u_3 + u_4}{4}, \quad \alpha_4 = \frac{u_1 - u_2 + u_3 - u_4}{4}$$

2D自然座標系の形状関数（2/2）

- 元の式に代入して，u_iについて整理すると以下のようになる：

$$u = N_1 u_1 + N_2 u_2 + N_3 u_3 + N_4 u_4$$



- 形状関数N_iは以下のようになる：

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

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

- 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
- Y方向変位vについても同様

$$v = N_1 v_1 + N_2 v_2 + N_3 v_3 + N_4 v_4$$

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
- 変位法
- 自由度：変位
  - 各節点上で3成分（u,v,w）が定義される

3D自然座標系の形状関数

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

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

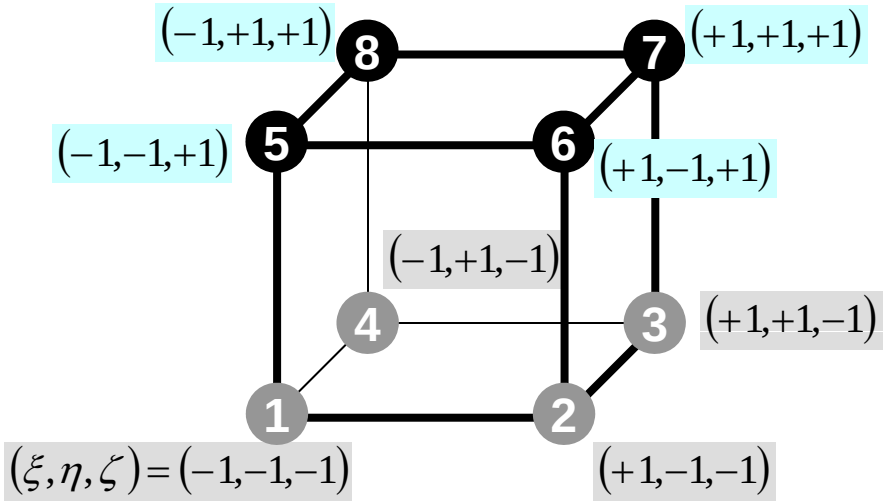
$$N_3(\xi,\eta,\zeta)=\frac{1}{8}(1+\xi)(1+\eta)(1-\zeta) \qquad N_7(\xi,\eta,\zeta)=\frac{1}{8}(1+\xi)(1+\eta)(1+\zeta)$$

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

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

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

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



- 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
- 宿題

- プログラムの実行
- データ構造
- プログラムの構成
- 計算効率について
- ParaViewによる可視化

弾性力学の支配方程式

- つりあい式
- ひずみ～変位関係式
- ひずみ～応力関係式

三次元のつりあい式
応力の独立な成分は6つ

$$\tau_{xy} = \tau_{yx}$$
$$\tau_{yz} = \tau_{zy}$$
$$\tau_{zx} = \tau_{xz}$$

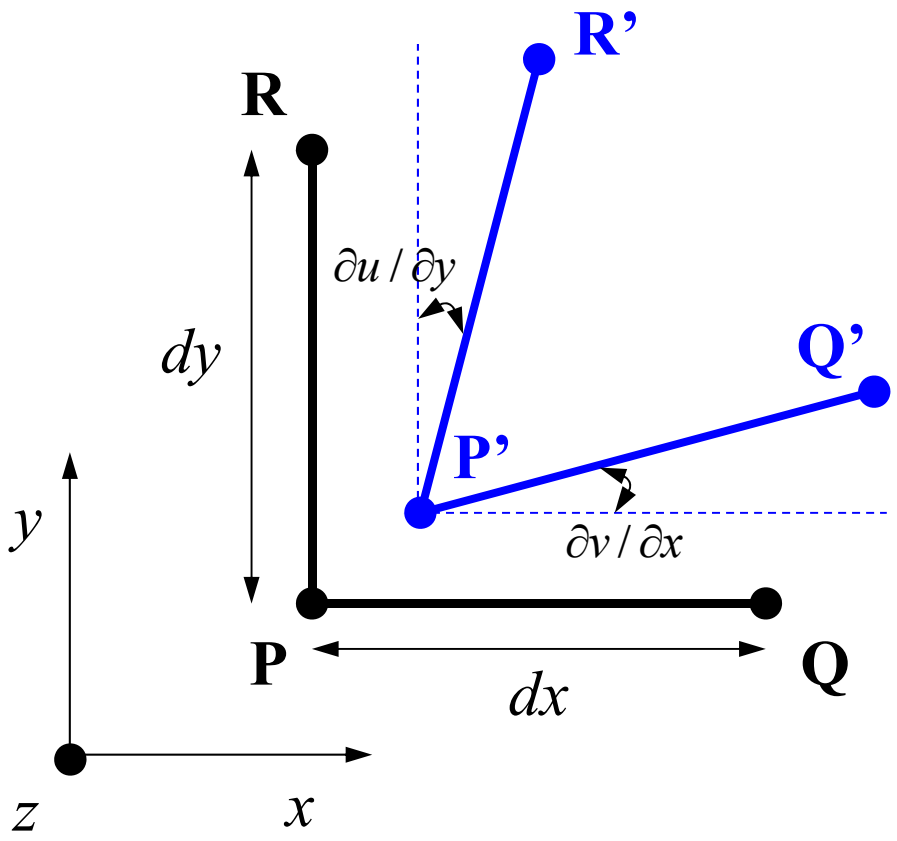
$$\{\boldsymbol{\sigma}\} = \begin{Bmatrix} \sigma_x & \tau_{xy} & \tau_{zx} \\ \tau_{xy} & \sigma_y & \tau_{yz} \\ \tau_{zx} & \tau_{yz} & \sigma_z \end{Bmatrix}$$

$$\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + X = 0$$
$$\frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{yz}}{\partial z} + Y = 0$$
$$\frac{\partial \tau_{zx}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \sigma_z}{\partial z} + Z = 0$$

垂直ひずみ～変位の関係

- PQ⇒P'Q'

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

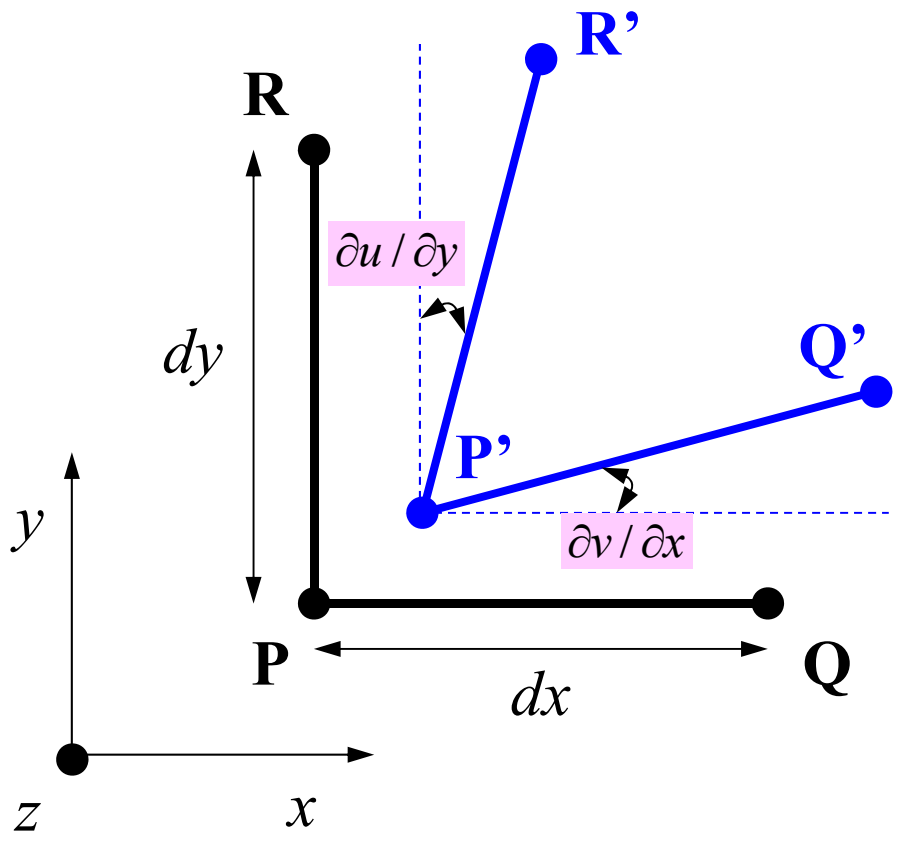


$$\varepsilon_x = \frac{\partial u}{\partial x}$$

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

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

せん断ひずみ～変位の関係



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

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

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

応力⇒ひずみ関係

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

ひずみ⇒応力関係

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

$[D]$

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

- 非圧縮性材料（ν~0.50）の場合，特別な扱い必要

X-方向のつりあい式に注目

$$\frac{\partial \sigma_x}{\partial x}+\frac{\partial \tau_{xy}}{\partial y}+\frac{\partial \tau_{zx}}{\partial z}+X=0$$

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

↓ ガラーキン法

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

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

X-方向のつりあい式（1/3）

$$\int_V [N]^T \{\sigma_{x,x}\} dV = -\int_V [N_{,x}]^T \{\sigma_x\} dV + \int_S [N]^T \{\sigma_x\} n_x dS$$

$$\because \int_V \frac{\partial}{\partial x} \left[[N]^T \{\sigma_x\} \right] dV = \int_S [N]^T \{\sigma_x\} n_x dS$$

一次元ガウスの公式

$$\int_V \frac{\partial}{\partial x} \left[[N]^T \{\sigma_x\} \right] dV = \int_V [N]^T \{\sigma_{x,x}\} dV + \int_V [N_{,x}]^T \{\sigma_x\} dV$$

部分積分の公式

同様にして以下の弱形式が得られる（表面積分省略）:

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

$$-\int_V [N_{,x}]^T \{\sigma_x\} dV - \int_V [N_{,y}]^T \{\tau_{xy}\} dV - \int_V [N_{,z}]^T \{\tau_{zx}\} dV + \int_V [N]^T \{X\} dV = 0$$

X-方向のつりあい式（2/3）

$$-\int_V [N_{,x}]^T \{\sigma_x\} dV - \int_V [N_{,y}]^T \{\tau_{xy}\} dV - \int_V [N_{,z}]^T \{\tau_{zx}\} dV + \int_V [N]^T \{X\} dV = 0 \quad (*)$$

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

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

$$\tau_{xy} = \frac{E}{2(1+\nu)} \gamma_{xy} = \frac{E}{2(1+\nu)} \left[u_{,y} + v_{,x} \right]$$

$$\tau_{zx} = \frac{E}{2(1+\nu)} \gamma_{zx} = \frac{E}{2(1+\nu)} \left[u_{,z} + w_{,x} \right]$$

要素内の変位分布，形状関数

$$u = [N]\{U\}, \quad v = [N]\{V\}, \quad w = [N]\{W\}$$

$$u_{,x} = [N_{,x}]\{U\}, \quad u_{,y} = [N_{,y}]\{U\}, \quad u_{,z} = [N_{,z}]\{U\}$$
$$v_{,x} = [N_{,x}]\{V\}, \quad v_{,y} = [N_{,y}]\{V\}$$
$$w_{,x} = [N_{,x}]\{W\}, \quad w_{,z} = [N_{,z}]\{W\}$$

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

$$\tau_{xy} = \frac{E}{2(1+\nu)}\gamma_{xy} = \frac{E}{2(1+\nu)}\left([N_{,y}]\{U\} + [N_{,x}]\{V\}\right)$$

$$\tau_{zx} = \frac{E}{2(1+\nu)}\gamma_{zx} = \frac{E}{2(1+\nu)}\left([N_{,z}]\{U\} + [N_{,x}]\{W\}\right)$$

X-方向のつりあい式（3/3）

$$D=\frac{(1-\nu)E}{(1+\nu)(1-2\nu)},\quad a=\frac{\nu E}{(1+\nu)(1-2\nu)},\quad b=\frac{E}{2(1+\nu)}$$ とすると

$$\sigma_x = D[N_{,x}]\{U\}+a[N_{,y}]\{V\}+a[N_{,z}]\{W\}$$
$$\tau_{xy} = b[N_{,y}]\{U\}+b[N_{,x}]\{V\}$$
$$\tau_{zx} = b[N_{,z}]\{U\}+b[N_{,x}]\{W\}$$

$$(*)=-\int_V [N_{,x}]^T\{\sigma_x\}dV-\int_V [N_{,y}]^T\{\tau_{xy}\}dV-\int_V [N_{,z}]^T\{\tau_{zx}\}dV+\int_V [N]^T\{X\}dV=$$
$$-\int_V \left\{D[N_{,x}]^T[N_{,x}]+b\left([N_{,y}]^T[N_{,y}]+[N_{,z}]^T[N_{,z}]\right)\right\}dV\{U\}$$
$$-\int_V \left\{a[N_{,x}]^T[N_{,y}]+b\left([N_{,y}]^T[N_{,x}]\right)\right\}dV\{V\}-\int_V \left\{a[N_{,x}]^T[N_{,z}]+b[N_{,z}]^T[N_{,x}]\right\}dV\{W\}$$
$$+\int_V [N]^T\{X\}dV=0$$

Y-方向のつりあい式

$$-\int_V \left\{D\left[N_{,y}\right]^T\left[N_{,y}\right]+b\left(\left[N_{,z}\right]^T\left[N_{,z}\right]+\left[N_{,x}\right]^T\left[N_{,x}\right]\right)\right\}dV\{V\}$$

$$-\int_V \left\{a\left[N_{,y}\right]^T\left[N_{,x}\right]+b\left(\left[N_{,x}\right]^T\left[N_{,y}\right]\right)\right\}dV\{U\}-\int_V \left\{a\left[N_{,y}\right]^T\left[N_{,z}\right]+b\left[N_{,z}\right]^T\left[N_{,y}\right]\right\}dV\{W\}$$

$$+\int_V \left[N\right]^T\{Y\}dV=0$$

Z-方向のつりあい式

$$-\int_V \left\{D\left[N_{,z}\right]^T\left[N_{,z}\right]+b\left(\left[N_{,x}\right]^T\left[N_{,x}\right]+\left[N_{,y}\right]^T\left[N_{,y}\right]\right)\right\}dV\{W\}$$

$$-\int_V \left\{a\left[N_{,z}\right]^T\left[N_{,x}\right]+b\left(\left[N_{,x}\right]^T\left[N_{,z}\right]\right)\right\}dV\{U\}-\int_V \left\{a\left[N_{,z}\right]^T\left[N_{,y}\right]+b\left[N_{,y}\right]^T\left[N_{,z}\right]\right\}dV\{V\}$$

$$+\int_V \left[N\right]^T\{Z\}dV=0$$

3つのつりあい式

- 変位の3成分を未知数，3つの方程式
- 3つの方程式はカップルしている（独立では無い）
- 形状関数ベクトル $[N]$：8×1行列
  - $[N]^T[N]$，　$[N_{,x}]^T[N_{,y}]$等 ：8×8行列

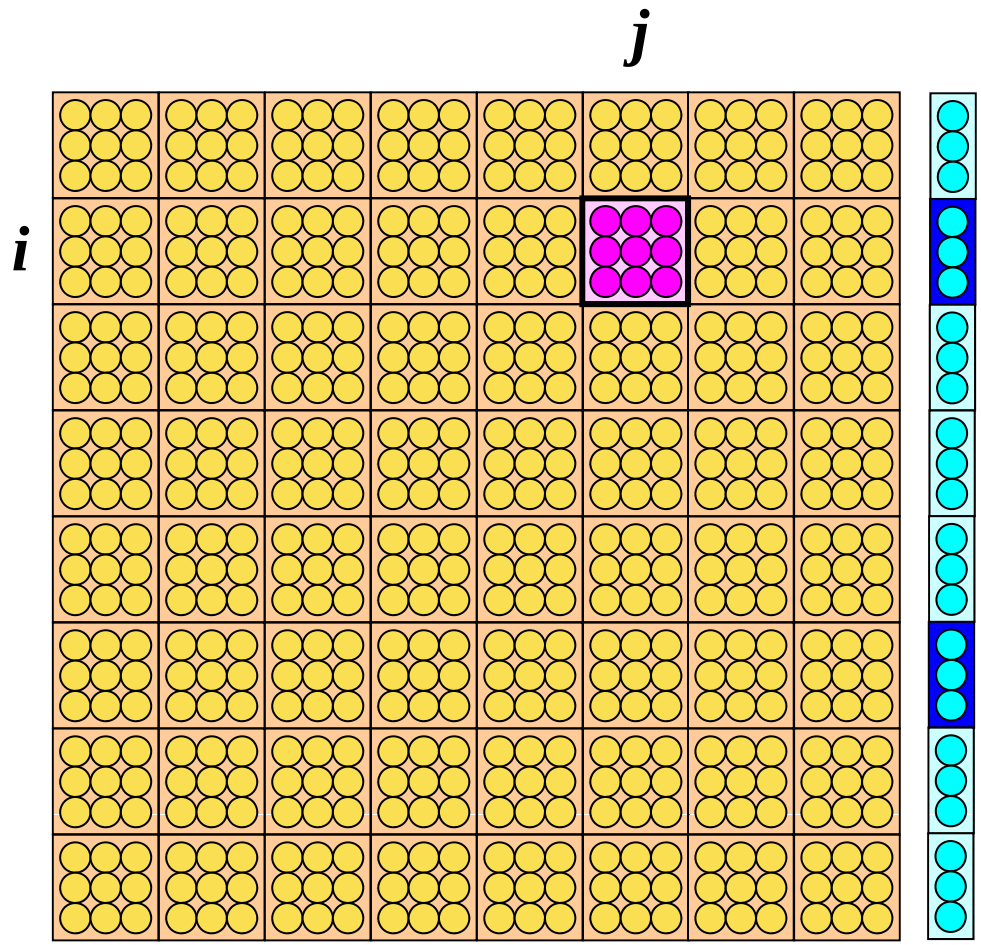
$$\frac{\partial \sigma_x}{\partial x}+\frac{\partial \tau_{xy}}{\partial y}+\frac{\partial \tau_{zx}}{\partial z}+X=0$$

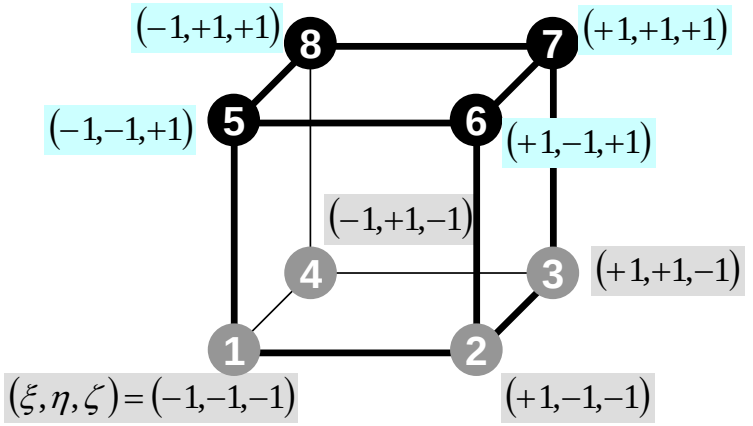
$$\frac{\partial \tau_{xy}}{\partial x}+\frac{\partial \sigma_y}{\partial y}+\frac{\partial \tau_{yz}}{\partial z}+Y=0$$

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

要素マトリクス：24×24行列

各節点上の（u,v,w）成分が物理的にも強くカップルしているので3自由度をまとめて扱う：8×8行列



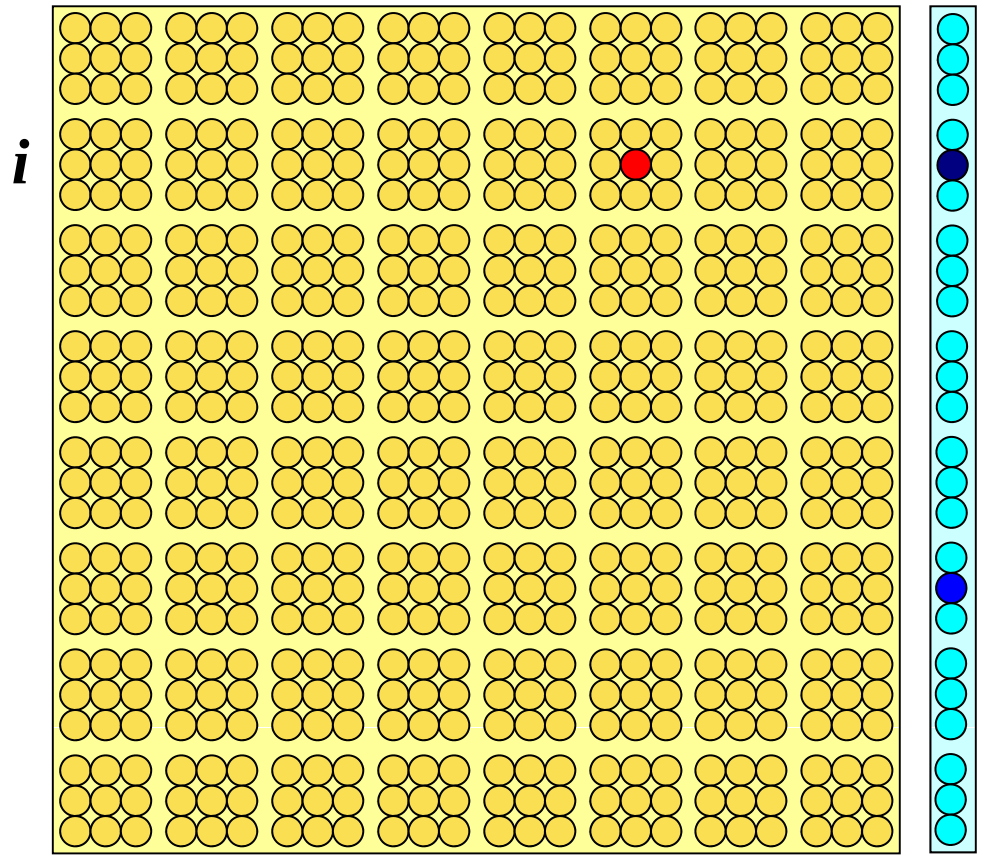


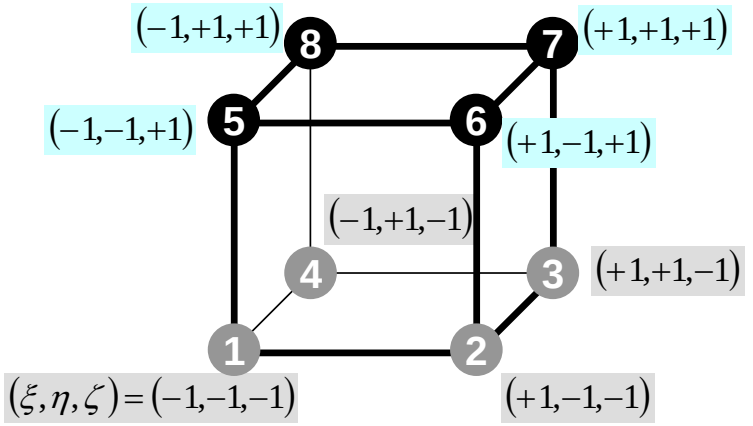
$$\begin{bmatrix} k_{ij}{}_{11} & k_{ij}{}_{12} & k_{ij}{}_{13} \\ k_{ij}{}_{21} & k_{ij}{}_{22} & k_{ij}{}_{23} \\ k_{ij}{}_{31} & k_{ij}{}_{32} & k_{ij}{}_{33} \end{bmatrix} \quad (i, j = 1 \ldots 8)$$

要素マトリクス：24×24行列

各節点上の（u,v,w）成分を独立に扱うことも可能であるが・・・

8×8とする利点については来週以降も説明する

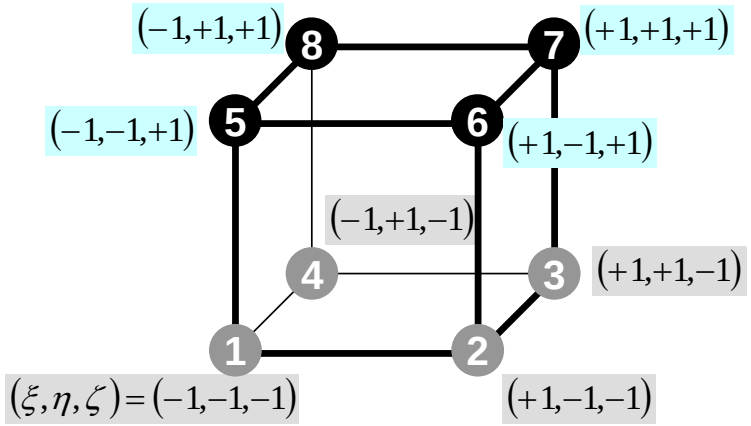




$$\begin{bmatrix} k_{ij}{}_{11} & k_{ij}{}_{12} & k_{ij}{}_{13} \\ k_{ij}{}_{21} & k_{ij}{}_{22} & k_{ij}{}_{23} \\ k_{ij}{}_{31} & k_{ij}{}_{32} & k_{ij}{}_{33} \end{bmatrix} \quad (i,j=1\ldots8)$$

要素マトリクス：i-j成分, X方向（1/3）

$$\begin{bmatrix} k_{ij_{11}} & k_{ij_{12}} & k_{ij_{13}} \\ k_{ij_{21}} & k_{ij_{22}} & k_{ij_{23}} \\ k_{ij_{31}} & k_{ij_{32}} & k_{ij_{33}} \end{bmatrix} \quad (i, j = 1 \ldots 8)$$



$$-\int_V \left\{ D\left[N_{,x}\right]^T\left[N_{,x}\right] + b\left(\left[N_{,y}\right]^T\left[N_{,y}\right] + \left[N_{,z}\right]^T\left[N_{,z}\right]\right)\right\} dV \{U\}$$
$$-\int_V \left\{ a\left[N_{,x}\right]^T\left[N_{,y}\right] + b\left(\left[N_{,y}\right]^T\left[N_{,x}\right]\right)\right\} dV \{V\}$$
$$-\int_V \left\{ a\left[N_{,x}\right]^T\left[N_{,z}\right] + b\left[N_{,z}\right]^T\left[N_{,x}\right]\right\} dV \{W\}$$

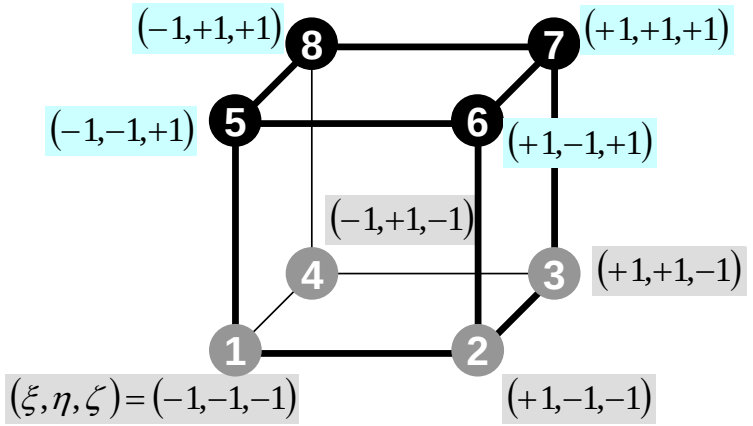
$$k_{ij_{11}} = -\int_V \left\{ D \cdot N_{i,x} \cdot N_{j,x} + b \cdot \left(N_{i,y} \cdot N_{j,y} + N_{i,z} \cdot N_{j,z} \right)\right\} dV$$

$$k_{ij_{12}} = -\int_V \left\{ a \cdot N_{i,x} \cdot N_{j,y} + b \cdot N_{i,y} \cdot N_{j,x} \right\} dV$$

$$k_{ij_{13}} = -\int_V \left\{ a \cdot N_{i,x} \cdot N_{j,z} + b \cdot N_{i,z} \cdot N_{j,x} \right\} dV$$

要素マトリクス：i-j成分, Y方向（2/3）

$$\begin{bmatrix} k_{ij\,11} & k_{ij\,12} & k_{ij\,13} \\ k_{ij\,21} & k_{ij\,22} & k_{ij\,23} \\ k_{ij\,31} & k_{ij\,32} & k_{ij\,33} \end{bmatrix} \quad (i,j=1\ldots8)$$



$$-\int_V \left\{D\left[N_{,y}\right]^T\left[N_{,y}\right]+b\left(\left[N_{,z}\right]^T\left[N_{,z}\right]+\left[N_{,x}\right]^T\left[N_{,x}\right]\right)\right\}dV\{V\}$$

$$-\int_V \left\{a\left[N_{,y}\right]^T\left[N_{,x}\right]+b\left(\left[N_{,x}\right]^T\left[N_{,y}\right]\right)\right\}dV\{U\}$$

$$-\int_V \left\{a\left[N_{,y}\right]^T\left[N_{,z}\right]+b\left[N_{,z}\right]^T\left[N_{,y}\right]\right\}dV\{W\}$$

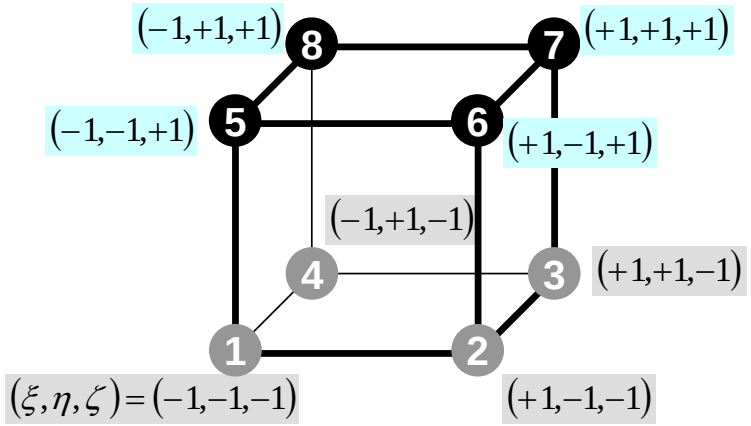
$$k_{ij\,21}=-\int_V \left\{a\cdot N_{i,y}\cdot N_{j,x}+b\cdot N_{i,x}\cdot N_{j,y}\right\}dV$$

$$k_{ij\,22}=-\int_V \left\{D\cdot N_{i,y}\cdot N_{j,y}+b\cdot\left(N_{i,z}\cdot N_{j,z}+N_{i,x}\cdot N_{j,x}\right)\right\}dV$$

$$k_{ij\,23}=-\int_V \left\{a\cdot N_{i,y}\cdot N_{j,z}+b\cdot N_{i,z}\cdot N_{j,y}\right\}dV$$

要素マトリクス：i-j成分, Z方向（3/3）

$$\begin{bmatrix} k_{ij\,11} & k_{ij\,12} & k_{ij\,13} \\ k_{ij\,21} & k_{ij\,22} & k_{ij\,23} \\ k_{ij\,31} & k_{ij\,32} & k_{ij\,33} \end{bmatrix} \quad (i,j=1\ldots8)$$



$$-\int_V \left\{D\left[N_{,z}\right]^T\left[N_{,z}\right]+b\left(\left[N_{,x}\right]^T\left[N_{,x}\right]+\left[N_{,y}\right]^T\left[N_{,y}\right]\right)\right\}dV\{W\}$$

$$-\int_V \left\{a\left[N_{,z}\right]^T\left[N_{,x}\right]+b\left(\left[N_{,x}\right]^T\left[N_{,z}\right]\right)\right\}dV\{U\}$$

$$-\int_V \left\{a\left[N_{,z}\right]^T\left[N_{,y}\right]+b\left[N_{,y}\right]^T\left[N_{,z}\right]\right\}dV\{V\}$$

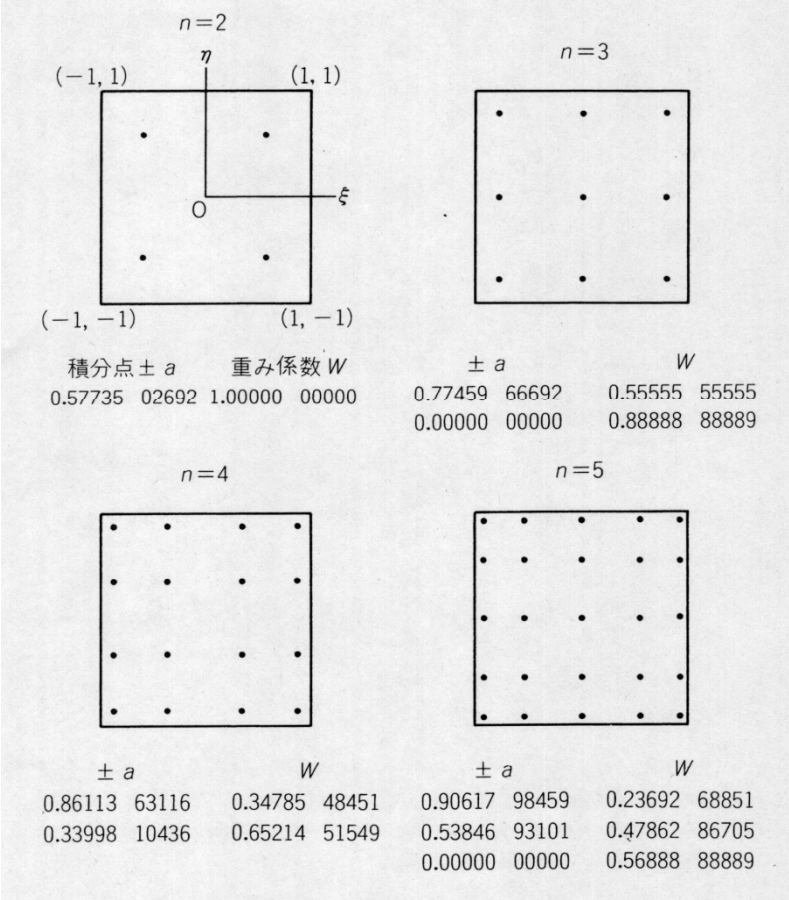
$$k_{ij\,31}=-\int_V \left\{a\cdot N_{i,z}\cdot N_{j,x}+b\cdot N_{i,x}\cdot N_{j,z}\right\}dV$$

$$k_{ij\,32}=-\int_V \left\{a\cdot N_{i,z}\cdot N_{j,y}+b\cdot N_{i,y}\cdot N_{j,z}\right\}dV$$

$$k_{ij\,33}=-\int_V \left\{D\cdot N_{i,z}\cdot N_{j,z}+b\cdot\left(N_{i,x}\cdot N_{j,x}+N_{i,y}\cdot N_{j,y}\right)\right\}dV$$

あとは積分を求めれば良い

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



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

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

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：

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

$[J]$: 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

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

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

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

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

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

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

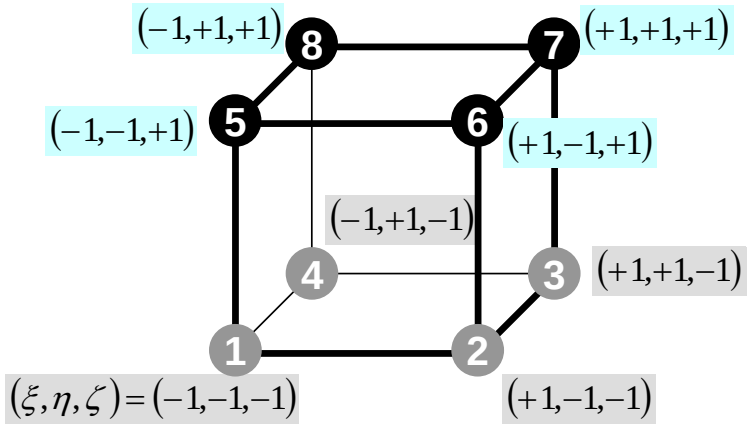
自然座標系における偏微分（4/4）

- 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

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

要素単位での積分

$$\begin{bmatrix} k_{ij\,11} & k_{ij\,12} & k_{ij\,13} \\ k_{ij\,21} & k_{ij\,22} & k_{ij\,23} \\ k_{ij\,31} & k_{ij\,32} & k_{ij\,33} \end{bmatrix} \quad (i, j = 1 \ldots 8)$$

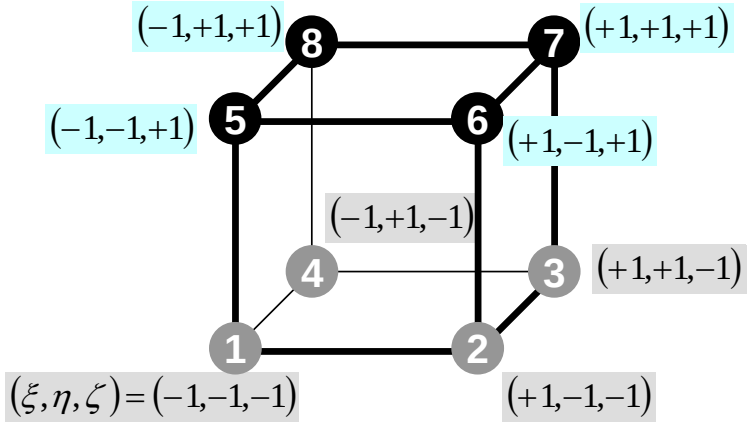


$$k_{ij\,11} = -\int_V \left\{ D \cdot N_{i,x} \cdot N_{j,x} + b\left(N_{i,y} \cdot N_{j,y} + N_{i,z} \cdot N_{j,z} \right) \right\} dV$$

$$= -\int_V \left\{ D \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + b\left(\frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right) \right\} dV$$

自然座標系での積分

$$\begin{bmatrix} k_{ij\,11} & k_{ij\,12} & k_{ij\,13} \\ k_{ij\,21} & k_{ij\,22} & k_{ij\,23} \\ k_{ij\,31} & k_{ij\,32} & k_{ij\,33} \end{bmatrix} \quad (i,j=1\ldots8)$$



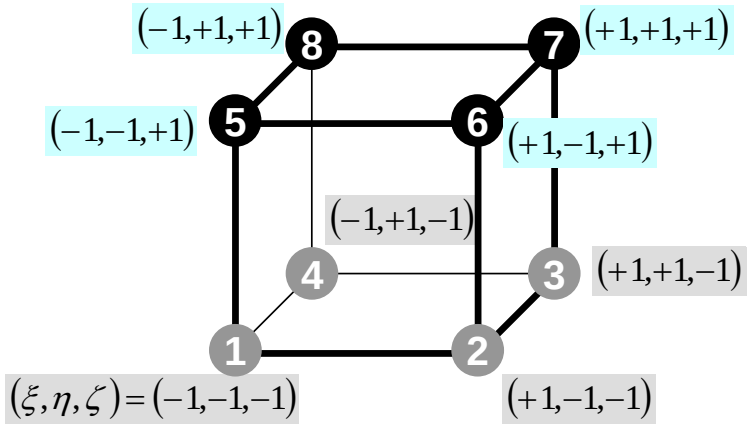
$$-\int_V \left\{ D\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + b\left(\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right) \right\} dV =$$

$$-\int\int\int \left\{ D\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + b\left(\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right) \right\} dxdydz =$$

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

ガウスの積分公式

$$\begin{bmatrix} k_{ij\,11} & k_{ij\,12} & k_{ij\,13} \\ k_{ij\,21} & k_{ij\,22} & k_{ij\,23} \\ k_{ij\,31} & k_{ij\,32} & k_{ij\,33} \end{bmatrix} \quad (i,j=1\ldots8)$$



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

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

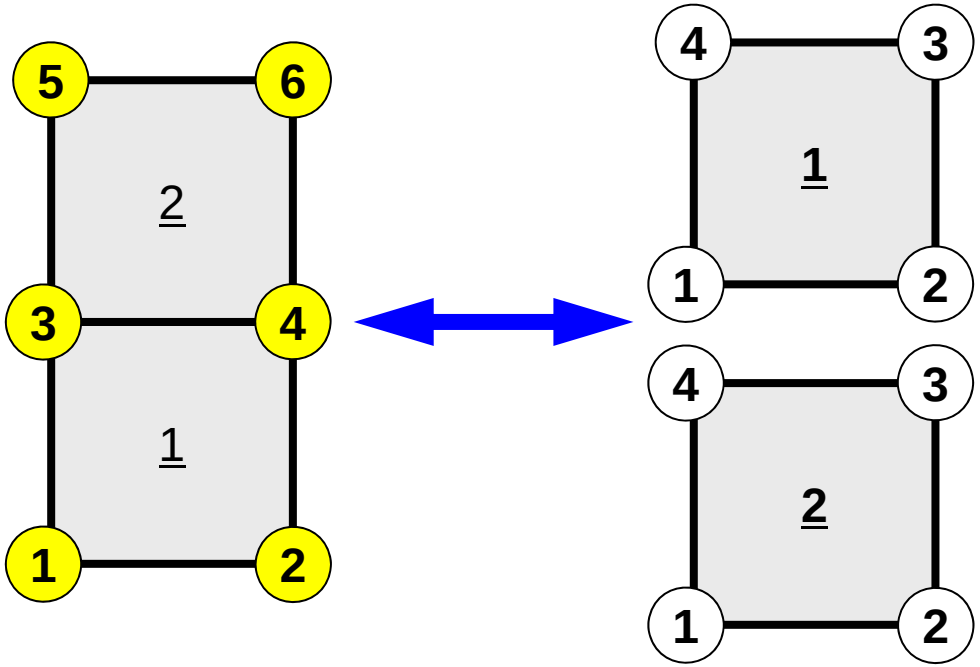
$$=\sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{k=1}^{N}\left[W_i\cdot W_j\cdot W_k\cdot f(\xi_i,\eta_j,\zeta_k)\right]$$

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・・・

- 詳細は来週以降の講義で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



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

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

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

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

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

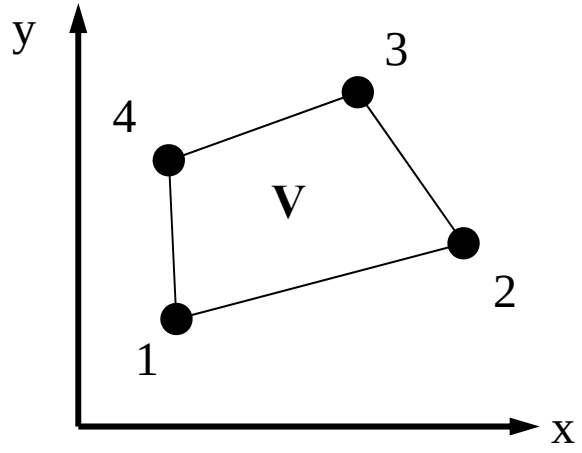
$$\begin{bmatrix} D_1 & X & X & X & & \\ X & D_2 & X & X & & \\ X & X & D_3 & X & X & X \\ X & X & X & D_4 & X & X \\ & & X & X & D_5 & X \\ & & X & X & X & D_6 \end{bmatrix} \begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \end{Bmatrix} = \begin{Bmatrix} B_1 \\ B_2 \\ B_3 \\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

- 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
- 宿題

- プログラムの実行
- データ構造
- プログラムの構成
- 計算効率について
- ParaViewによる可視化

宿題

- 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)

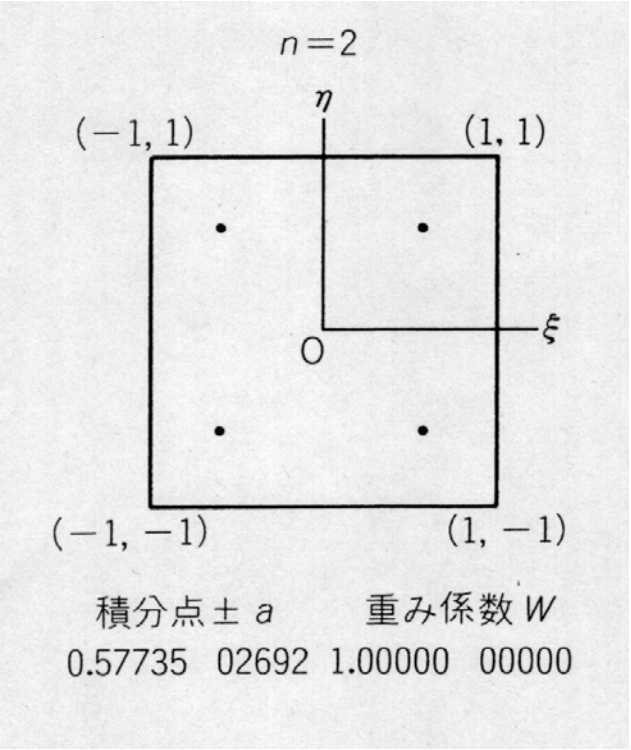
$$I = \int_V dV = \int_{-1}^{+1}\int_{-1}^{+1} \det|J|\, d\xi d\zeta$$

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。

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

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

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

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



ヒント（2/2）

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

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

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

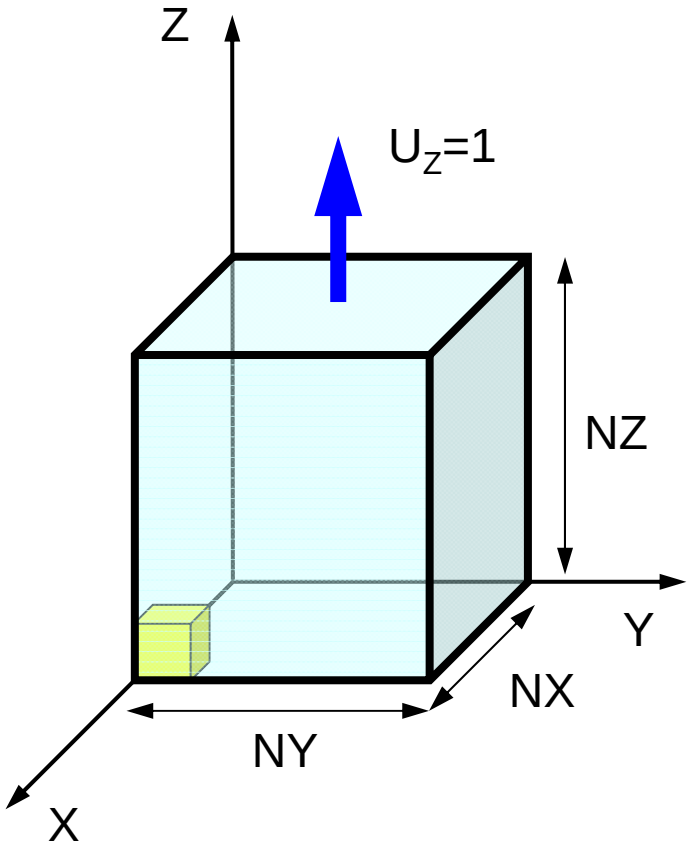
$$N_1(\xi,\eta)=\frac{1}{4}(1-\xi)(1-\eta),\quad N_2(\xi,\eta)=\frac{1}{4}(1+\xi)(1-\eta),$$

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

- 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
- 宿題

- プログラムの実行
- データ構造
- プログラムの構成
- 計算効率について
- ParaViewによる可視化

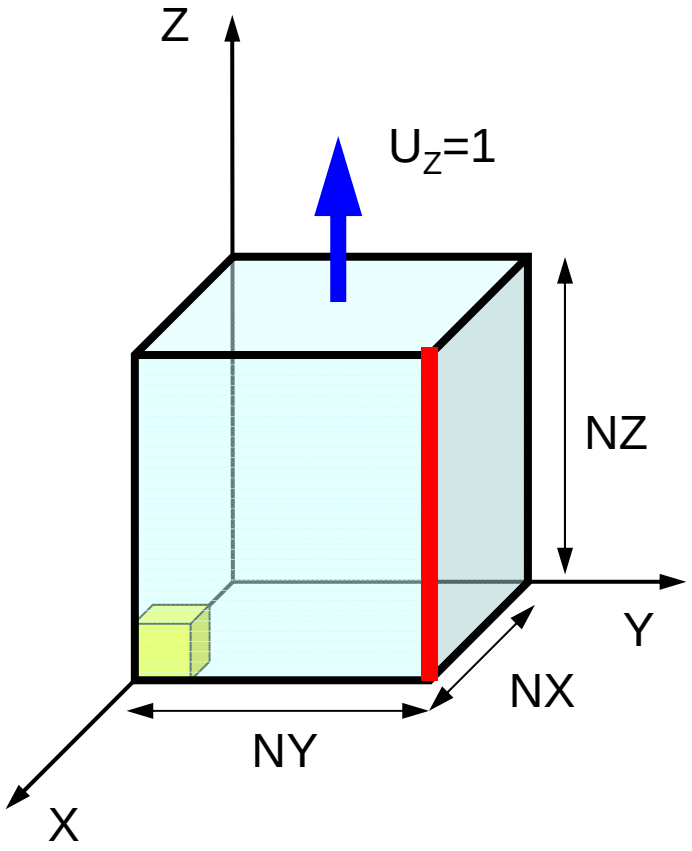
対象とする問題



- 弾性体
  - ヤング率　　E
  - ポアソン比　　ν
- 直方体
  - 一辺長さ1の立方体（六面体）要素
  - 各方向にNX・NY・NZ個
- 境界条件
  - 対称条件
    - U_X=0@X=0
    - U_Y=0@Y=0
    - U_Z=0@Z=0
  - 強制変位
    - U_Z=1@Z=Z_{max}

movie

NX=NY=NZ=10, ν=0.30とすると



$$\varepsilon_z = \frac{1}{10} = 0.10$$

$$\varepsilon_x = \varepsilon_y = -\nu\varepsilon_z = -0.03$$

$$\therefore u_x\big|_{X=10,Y=10} = u_y\big|_{X=10,Y=10} = 10 \times \varepsilon_x = -0.30$$

ファイルコピー，インストール（1/2）

三次元弾性解析コード

```
>$ cd <$fem1>
>$ cp /home03/skengon/Documents/class/fem1/fem3d.tar .
>$ tar xvf fem3d.tar
>$ cd fem3d
>$ ls
   c   f  run
```

FEMインストール（C）

```
>$ cd <$fem1>/fem3d/c
>$ make
>$ ls ../run/sol
   sol
```

FEMインストール（FORTRAN）

```
>$ cd <$fem1>/fem3d/f
>$ make
>$ ls ../run/sol
   sol
```

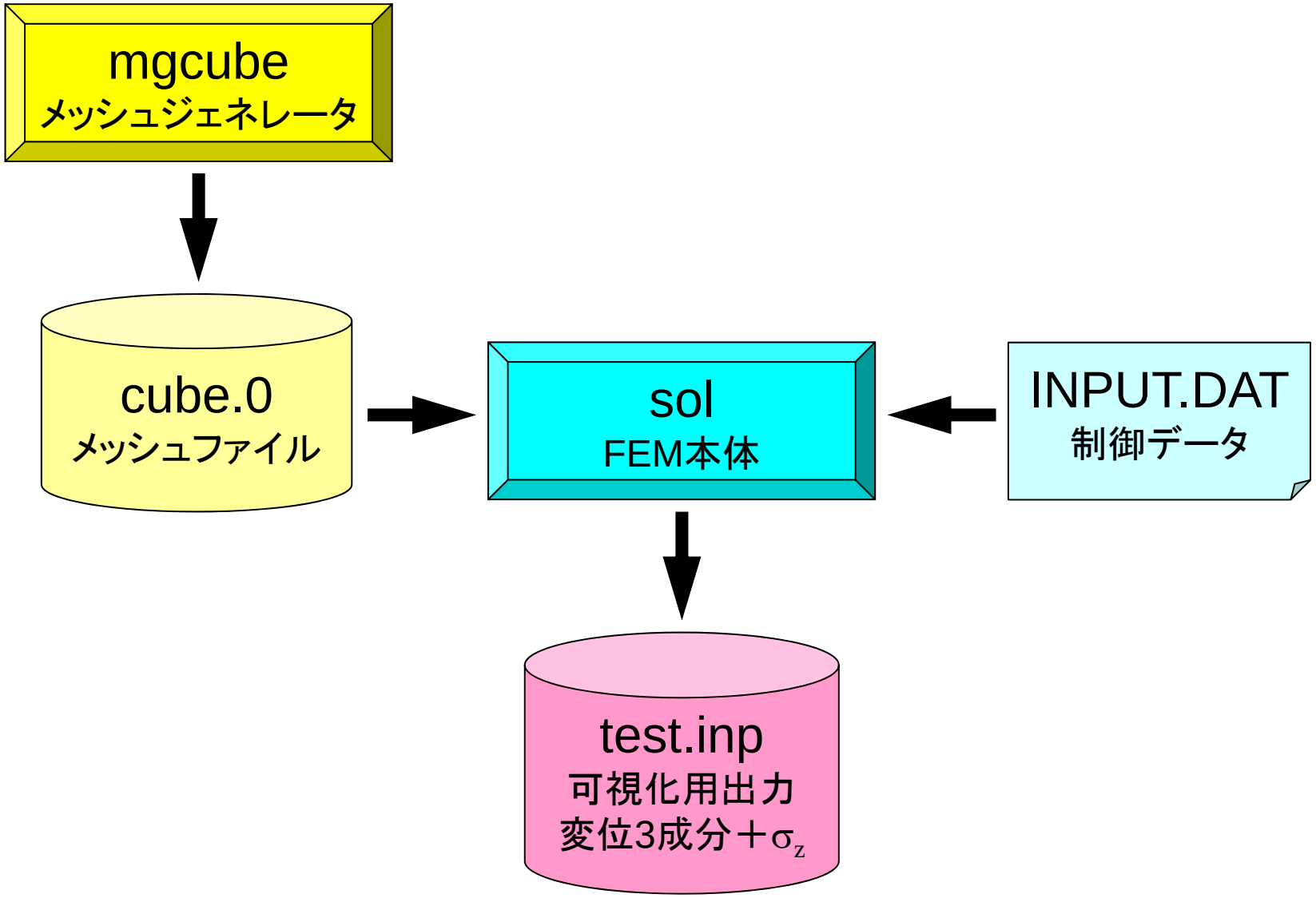
ファイルコピー，インストール（2/2）

メッシュジェネレータインストール

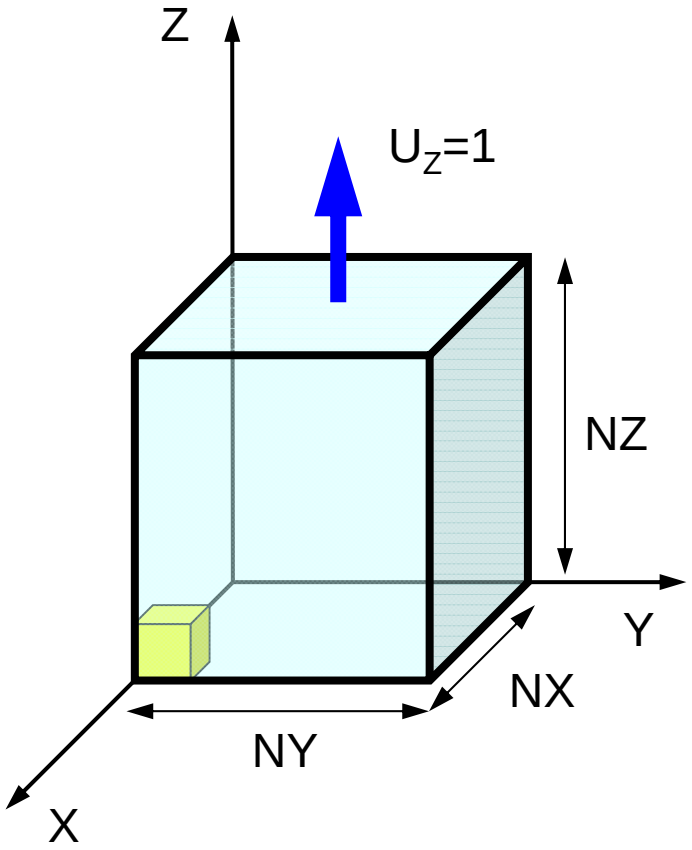
```
>$ cd <$fem1>/fem3d/run
>$ g95 -O3 mgcube.f -o mgcube
```

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




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

   NX, NY, NZ          ← 各辺長さを
                          訊いてくる
   10,10,10            ← このように
                          入れてみる

>$ ls cube.0              生成を確認

    cube.0
```

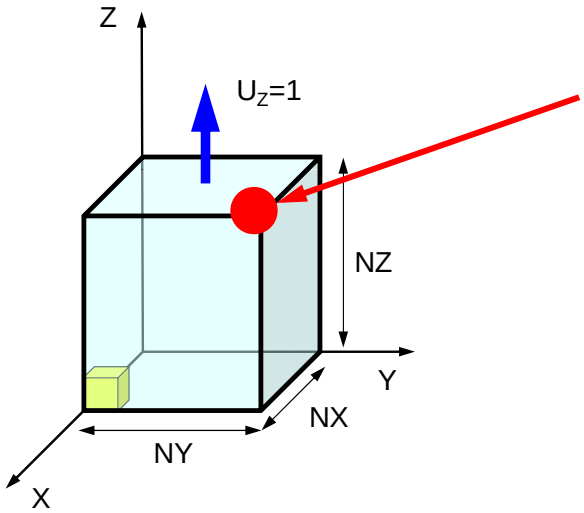
制御ファイル：INPUT.DAT

```
INPUT.DAT

cube.0          fname
1  0            METHOD, PRECOND
1               iterPREmax（不使用）
2000            ITER
1.0 0.3         ELAST, POISSON
```

- fname： メッシュファイル名
- METHOD： 反復解法（1に固定）
- PRECOND： 前処理手法
  - 0：Block-LU-GS，1：Block対角スケーリング
- iterPREmax: （不使用）
- ITER： 反復回数上限
- ELAST： ヤング率
- POISSON： ポアソン比
  - あとで=0.4999などの場合を試してみよ

実 行

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

（反復法の収束履歴）
  33    2.218867E-08
  34    1.325902E-08
  35    7.384341E-09

### DISPLACEMENT at (Xmax,Ymax,Zmax))
    1331   -3.000000E-01   -3.000002E-01    1.000000E+00

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



この点での変位が表示されている
1331番節点（=11^3）

- 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
- 宿題

- プログラムの実行
- データ構造
- プログラムの構成
- 計算効率について
- ParaViewによる可視化

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

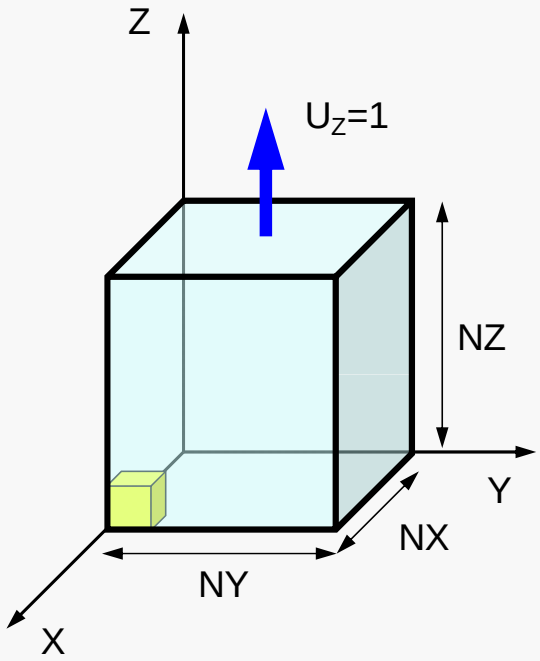
メッシュ生成コード：mgcube.f（1/5）

```
      implicit REAL*8 (A-H,O-Z)
      real(kind=8), dimension(:,:), allocatable :: X , Y
      real(kind=8), dimension(:,:), allocatable :: XO, YO
      character(len=80) :: GRIDFILE, HHH
      integer , dimension(:,:), allocatable :: IW
!C
!C +-------+
!C | INIT. |
!C +-------+
!C===
      write (*,*) 'NX,NY,NZ'
      read  (*,*)  NX,NY,NZ

      NXP1= NX + 1                                X方向節点数
      NYP1= NY + 1                                Y方向節点数
      NZP1= NZ + 1                                Z方向節点数

      DX= 1.d0

      INODTOT=  NXP1*NYP1*NZP1                    総節点数
      ICELTOT=  NX  *NY  *NZ                      総要素数
      IBNODTOT= NXP1*NYP1                         XY平面の節点数

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

メッシュ生成コード：mgcube.f（2/5）

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

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

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

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

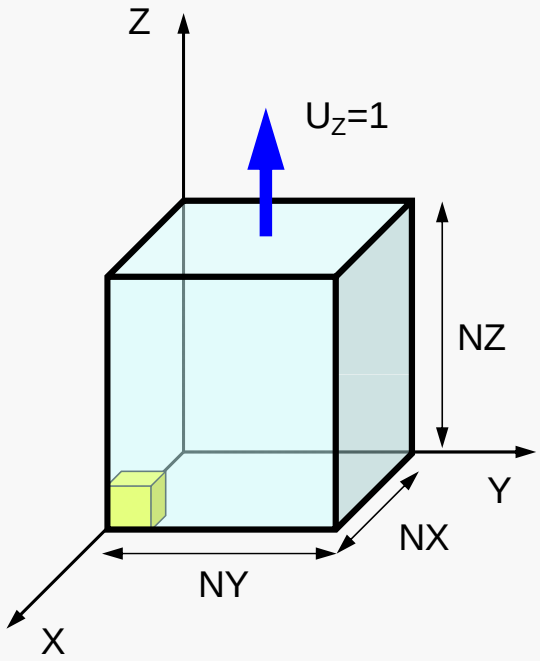
X=Xminの節点をIW(ib,1)に格納
（ib=1,NYP1*NZP1）



メッシュ生成コード：mgcube.f（2/5）

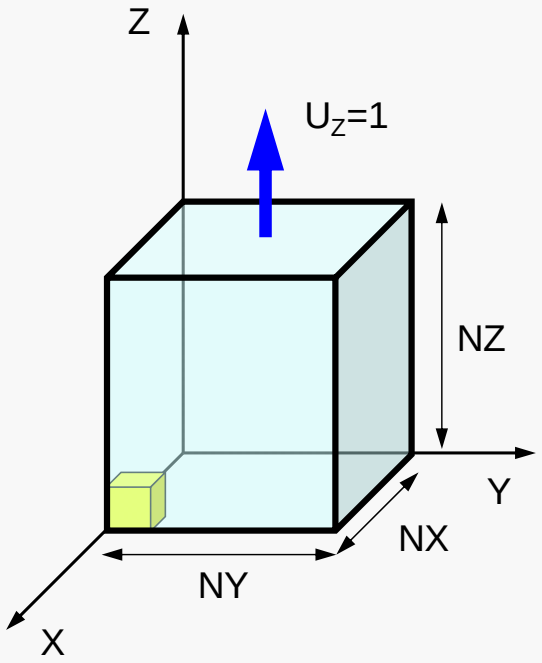
```
      icou= 0
      ib  = 1
      do k= 1, NZP1
      do j= 1, NYP1
        i= 1
        icou= icou + 1
        ii  = (k-1)*IBNODTOT + (j-1)*NXP1 + i
        IW(icou,ib)= ii
      enddo
      enddo

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

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

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

Y=Yminの節点をIW(ib,2)に格納
（ib=1,NXP1*NZP1）



メッシュ生成コード：mgcube.f（2/5）

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

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

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

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



Z=Zminの節点をIW(ib,3)に格納
（ib=1,NXP1*NYP1）

Z=Zmaxの節点をIW(ib,4)に格納
（ib=1,NXP1*NYP1）

メッシュ生成コード：mgcube.f（3/5）

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

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

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

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

          write(12,'(i10)') ICELTOT

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

節点数
節点番号，座標

要素数
要素タイプ（使わないデータ）

cube.0：節点データ，要素数，要素タイプ（NX=NY=NZ=4）

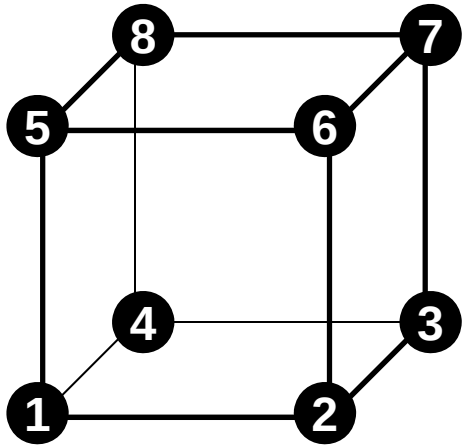
```
      125                                                                  =5*5*5
        1    0.000000E+00    0.000000E+00    0.000000E+00
        2    1.000000E+00    0.000000E+00    0.000000E+00
        3    2.000000E+00    0.000000E+00    0.000000E+00
        4    3.000000E+00    0.000000E+00    0.000000E+00
        5    4.000000E+00    0.000000E+00    0.000000E+00
        6    0.000000E+00    1.000000E+00    0.000000E+00
        7    1.000000E+00    1.000000E+00    0.000000E+00
        8    2.000000E+00    1.000000E+00    0.000000E+00
        9    3.000000E+00    1.000000E+00    0.000000E+00

（途中省略）

      121    0.000000E+00    4.000000E+00    4.000000E+00
      122    1.000000E+00    4.000000E+00    4.000000E+00
      123    2.000000E+00    4.000000E+00    4.000000E+00
      124    3.000000E+00    4.000000E+00    4.000000E+00
      125    4.000000E+00    4.000000E+00    4.000000E+00
       64                                                                  =4*4*4
      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
```

movie

メッシュ生成コード：mgcube.f（4/5）

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

imat：材料番号（=1）



cube.0：要素データ

```
    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

（途中省略）

   42    1   62   63   68   67   87   88   93   92
   43    1   63   64   69   68   88   89   94   93
   44    1   64   65   70   69   89   90   95   94
   45    1   66   67   72   71   91   92   97   96
   46    1   67   68   73   72   92   93   98   97
   47    1   68   69   74   73   93   94   99   98
   48    1   69   70   75   74   94   95  100   99
   49    1   76   77   82   81  101  102  107  106
   50    1   77   78   83   82  102  103  108  107
   51    1   78   79   84   83  103  104  109  108
   52    1   79   80   85   84  104  105  110  109
   53    1   81   82   87   86  106  107  112  111
   54    1   82   83   88   87  107  108  113  112
   55    1   83   84   89   88  108  109  114  113
   56    1   84   85   90   89  109  110  115  114
   57    1   86   87   92   91  111  112  117  116
   58    1   87   88   93   92  112  113  118  117
   59    1   88   89   94   93  113  114  119  118
   60    1   89   90   95   94  114  115  120  119
   61    1   91   92   97   96  116  117  122  121
   62    1   92   93   98   97  117  118  123  122
   63    1   93   94   99   98  118  119  124  123
   64    1   94   95  100   99  119  120  125  124
```

メッシュ生成コード：mgcube.f（5/5）

```
      IGTOT= 4

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


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

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


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

IGTOT	グループ総数 （Xmin, Ymin, Zmin, Zmax）
IBTx	累積数

cube.0：節点グループデータ

```
        4
       25       50       75      100
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）
```

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

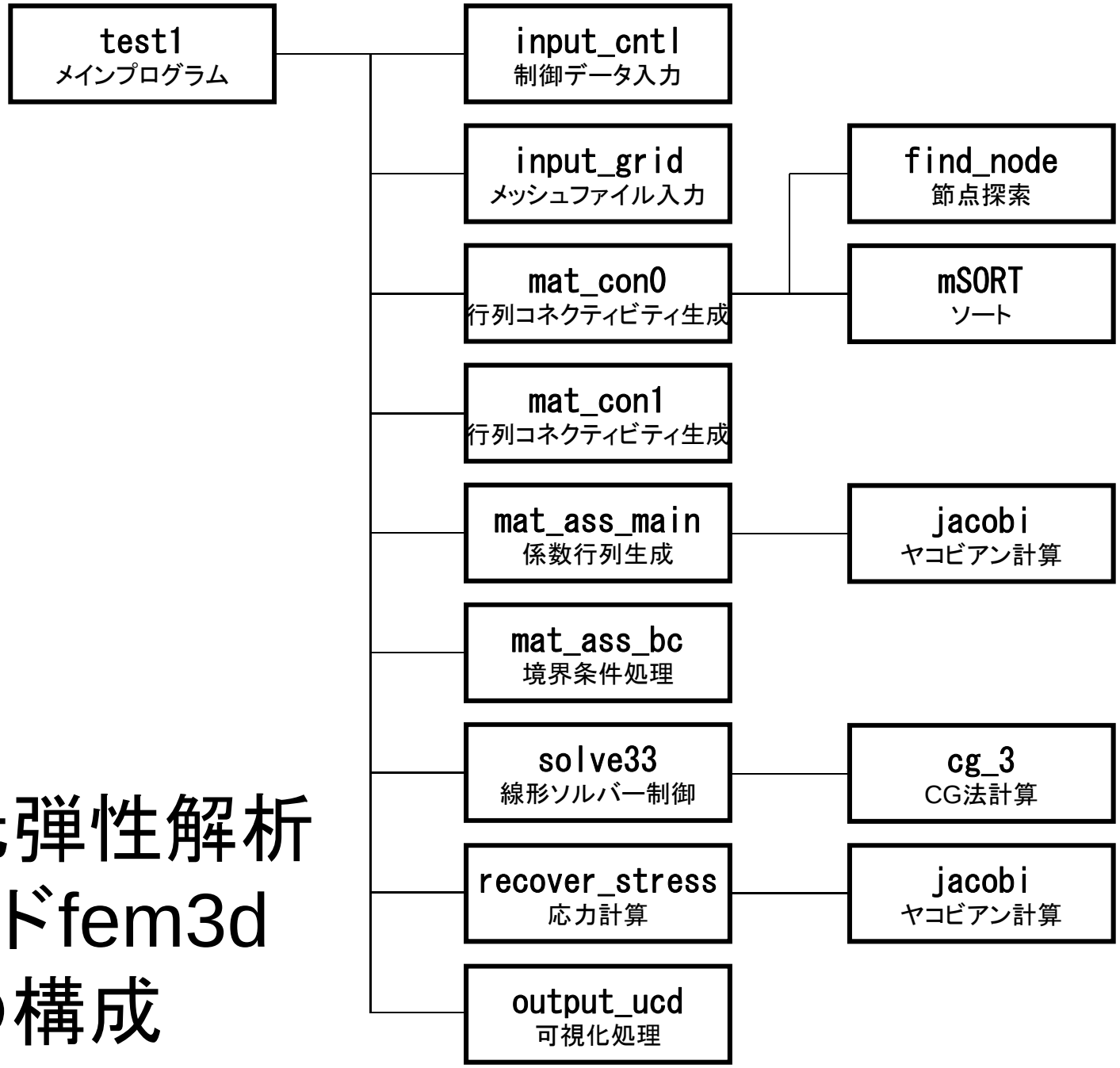
- 三次元要素の定式化
- 三次元弾性力学方程式
  - ガラーキン法
  - 要素マトリクス生成
- 宿題

- プログラムの実行
- データ構造
- プログラムの構成
- 計算効率について
- ParaViewによる可視化

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）
- 応力計算

三次元弾性解析コードfem3dの構成



全体処理

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

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

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

        INPUT_CNTL();
        INPUT_GRID();


        MAT_CON0();
        MAT_CON1();

        MAT_ASS_MAIN();
        MAT_ASS_BC()  ;

        SOLVE33();

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

Global変数表：pfem_util.h/f（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
NL, NU	I		O	各節点非対角成分数（上三角・下三角）
NPL, NPU	I		O	非対角成分総数（上三角・下三角）
D	R	[9*N]	O	全体行列：対角ブロック
B,X	R	[3*N]	O	右辺ベクトル，未知数ベクトル

Global変数表：pfem_util.h/f（2/3）

変数名	種別	サイズ	I/O	内　容
ALUG	R	[9*N]	O	全体行列：対角ブロックの完全LU分解
AL, AU	R	[9*NPL], [9*NPU]	O	全体行列：上・下三角成分
indexL, indexU	I	[N+1]	O	全体行列：非零非対角ブロック数
itemL, itemU	I	[NPL], [NPU]	O	全体行列：上・下三角ブロック（列番号）
INL, INU	I	[N]	O	各節点の上・下三角ブロック数
IAL, IAU	I	[N][NL], [N][NU]	O	各節点の上・下三角ブロック（列番号）
IWKX	I	[N][2]	O	ワーク用配列
METHOD	I		I	反復解法（=1に固定）
PRECOND	I		I	前処理手法（=0：ブロックSSOR，=1：ブロック対角スケーリング）
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）
SIGMA_DIAG	R		I	LU分解時の対角成分係数（=1.0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/f（3/3）

変数名	種別	サイズ	I/O	内　容
O8th	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=1~8)
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
ELAST, POISSON	R		I	ヤング率，ポアソン比
SIGMA_N, TAU_N	R	[N][3]	O	節点における垂直，せん断応力成分

制御ファイル入力：INPUT_CNTL

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

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

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

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

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

INPUT.DAT

```
cube.0          fname
1  0            METHOD, PRECOND
1               iterPREmax（不使用）
2000            ITER
1.0 0.3         ELAST, POISSON
```

メッシュ入力：INPUT_GRID（1/2）

```
#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]);
        }
```

XYZ[N][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（2/2）

```
/**
        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",&ii,&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]);
        }
/**
        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);
}
```

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。

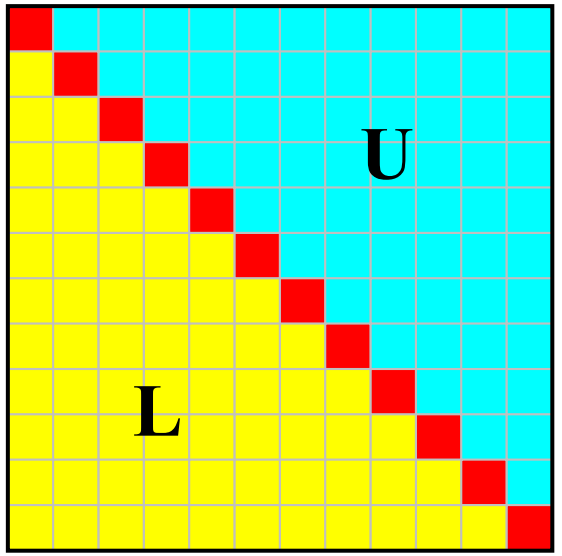
メッシュ入力：INPUT_GRID（2/2）

```
/**
        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",&ii,&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]);
        }
/**
        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。

fem3d：いくつかの特徴

- 非対角成分
  - 上三角，下三角を分けて記憶
- ブロックとして記憶
  - ベクトル：1節点3成分
  - 行列：各ブロック9成分
  - 行列の各成分ではなく，節点上の3変数に基づくブロックとして処理する



有限要素法で得られるマトリクス

- 疎行列
  - 0が多い
- $A(i,j)$のように正方行列の全成分を記憶することは疎行列では非効率的
  - 「密」行列向け

$$
\begin{bmatrix}
D & X & & & X & X & & & & & & & & & & \\
X & D & X & & X & X & X & & & & & & & & & \\
 & X & D & X & & X & X & X & & & & & & & & \\
 & & X & D & & & X & X & & & & & & & & \\
X & X & & & D & X & & & X & X & & & & & & \\
X & X & X & & X & D & X & & X & X & X & & & & & \\
 & X & X & X & & X & D & X & & X & X & X & & & & \\
 & & X & X & & & X & D & & & X & X & & & & \\
 & & & & X & X & & & D & X & & & X & X & & \\
 & & & & X & X & X & & X & D & X & & X & X & X & \\
 & & & & & X & X & X & & X & D & X & & X & X & X \\
 & & & & & & X & X & & & X & D & & & X & X \\
 & & & & & & & & X & X & & & D & X & & \\
 & & & & & & & & X & X & X & & X & D & X & \\
 & & & & & & & & & X & X & & & X & D & X \\
 & & & & & & & & & & X & X & & & X & D
\end{bmatrix}
\begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \\ \Phi_7 \\ \Phi_8 \\ \Phi_9 \\ \Phi_{10} \\ \Phi_{11} \\ \Phi_{12} \\ \Phi_{13} \\ \Phi_{14} \\ \Phi_{15} \\ \Phi_{16} \end{Bmatrix}
=
\begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \\ F_8 \\ F_9 \\ F_{10} \\ F_{11} \\ F_{12} \\ F_{13} \\ F_{14} \\ F_{15} \\ F_{16} \end{Bmatrix}
$$

- 有限要素法：非零非対角成分の数は高々「数百」規模
  - 例えば未知数が10^8個あるとすると記憶容量（ワード数）は
    - 正方行列：$O(10^{16})$
    - 非零非対角成分数：$O(10^{10})$
- 非零成分のみ記憶するのが効率的

1d.f, 1d.cにおけるマトリクス関連変数

変数名	型	サイズ	内容
N	I	-	未知数総数
NPLU	I	-	連立一次方程式係数マトリクス非対角成分総数
Diag(:)	R	N	連立一次方程式係数マトリクス対角成分
U(:)	R	N	連立一次方程式未知数ベクトル
Rhs(:)	R	N	連立一次方程式右辺ベクトル
Index(:)	I	0:N N+1	係数マトリクス非対角成分要素番号用一次元圧縮配列(非対角成分数)
Item(:)	I	NPLU	係数マトリクス非対角成分要素番号用一次元圧縮配列(非対角成分要素(列)番号)
AMat(:)	R	NPLU	係数マトリクス非対角成分要素番号用一次元圧縮配列(非対角成分)

非零非対角成分のみを格納する
Compressed Row Storage法を使用している。

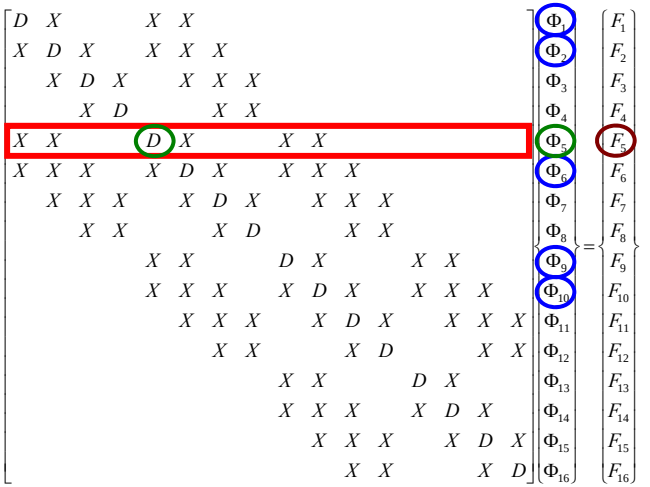
行列ベクトル積への適用

（非零）非対角成分のみを格納，疎行列向け方法

Compressed Row Storage (CRS)

```
Diag (i)  対角成分（実数，i=1,N）
Index(i)  非対角成分数に関する一次元配列（通し番号）
          （整数，i=0,N）
Item(k)   非対角成分の要素（列）番号
          （整数，k=1, index(N)）
AMat(k)   非対角成分
          （実数，k=1, index(N)）
```

```
{Y}= [A]{X}

do i= 1, N
  Y(i)= Diag(i)*X(i)
  do k= Index(i-1)+1, Index(i)
    Y(i)= Y(i) + Amat(k)*X(Item(k))
  enddo
enddo
```



行列ベクトル積への適用

（非零）非対角成分のみを格納，疎行列向け方法
Compressed Row Storage (CRS)

```
{Q}=[A]{P}

for(i=0;i<N;i++){
      W[Q][i] = Diag[i] * W[P][i];
      for(k=Index[i];k<Index[i+1];k++){
            W[Q][i] += AMat[k]*W[P][Item[k]];
      }
}
```

行列ベクトル積：密行列⇒とても簡単

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1,N-1} & a_{1,N} \\ a_{21} & a_{22} & & a_{2,N-1} & a_{2,N} \\ \cdots & & \cdots & & \cdots \\ a_{N-1,1} & a_{N-1,2} & & a_{N-1,N-1} & a_{N-1,N} \\ a_{N,1} & a_{N,2} & \cdots & a_{N,N-1} & a_{N,N} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-1} \\ y_N \end{Bmatrix}$$

```
{Y}= [A]{X}

do j= 1, N
  Y(j)= 0.d0
  do i= 1, N
    Y(j)= Y(j) + A(i,j)*X(i)
  enddo
enddo
```

Compressed Row Storage (CRS)

	1	2	3	4	5	6	7	8
1	1.1	2.4	0	0	3.2	0	0	0
2	4.3	3.6	0	2.5	0	3.7	0	9.1
3	0	0	5.7	0	1.5	0	3.1	0
4	0	4.1	0	9.8	2.5	2.7	0	0
5	3.1	9.5	10.4	0	11.5	0	4.3	0
6	0	0	6.5	0	0	12.4	9.5	0
7	0	6.4	2.5	0	0	1.4	23.1	13.1
8	0	9.5	1.3	9.6	0	3.1	0	51.3

Compressed Row Storage (CRS)：C

	0	1	2	3	4	5	6	7
0	**1.1** ◎	2.4 ①			3.2 ④			
1	4.3 ◎	**3.6** ①		2.5 ③		3.7 ⑤		9.1 ⑦
2			**5.7** ②		1.5 ④		3.1 ⑥	
3		4.1 ①		**9.8** ③	2.5 ④	2.7 ⑤		
4	3.1 ◎	9.5 ①	10.4 ②		**11.5** ④		4.3 ⑥	
5			6.5 ②			**12.4** ⑤	9.5 ⑥	
6		6.4 ①	2.5 ②			1.4 ⑤	**23.1** ⑥	13.1 ⑦
7		9.5 ①	1.3 ②	9.6 ③		3.1 ⑤		**51.3** ⑦

```
N= 8

対角成分
 Diag[0]=  1.1
 Diag[1]=  3.6
 Diag[2]=  5.7
 Diag[3]=  9.8
 Diag[4]= 11.5
 Diag[5]= 12.4
 Diag[6]= 23.1
 Diag[7]= 51.3
```

Compressed Row Storage (CRS)：C

	0	1	2	3	4	5	6	7
0	1.1 ◎		2.4 ①			3.2 ④			
1	3.6 ①	4.3 ◎			2.5 ③		3.7 ⑤		9.1 ⑦
2	5.7 ②					1.5 ④		3.1 ⑥	
3	9.8 ③		4.1 ①			2.5 ④	2.7 ⑤		
4	11.5 ④	3.1 ◎	9.5 ①	10.4 ②				4.3 ⑥	
5	12.4 ⑤			6.5 ②				9.5 ⑥	
6	23.1 ⑥		6.4 ①	2.5 ②			1.4 ⑤		13.1 ⑦
7	51.3 ⑦		9.5 ①	1.3 ②	9.6 ③		3.1 ⑤		

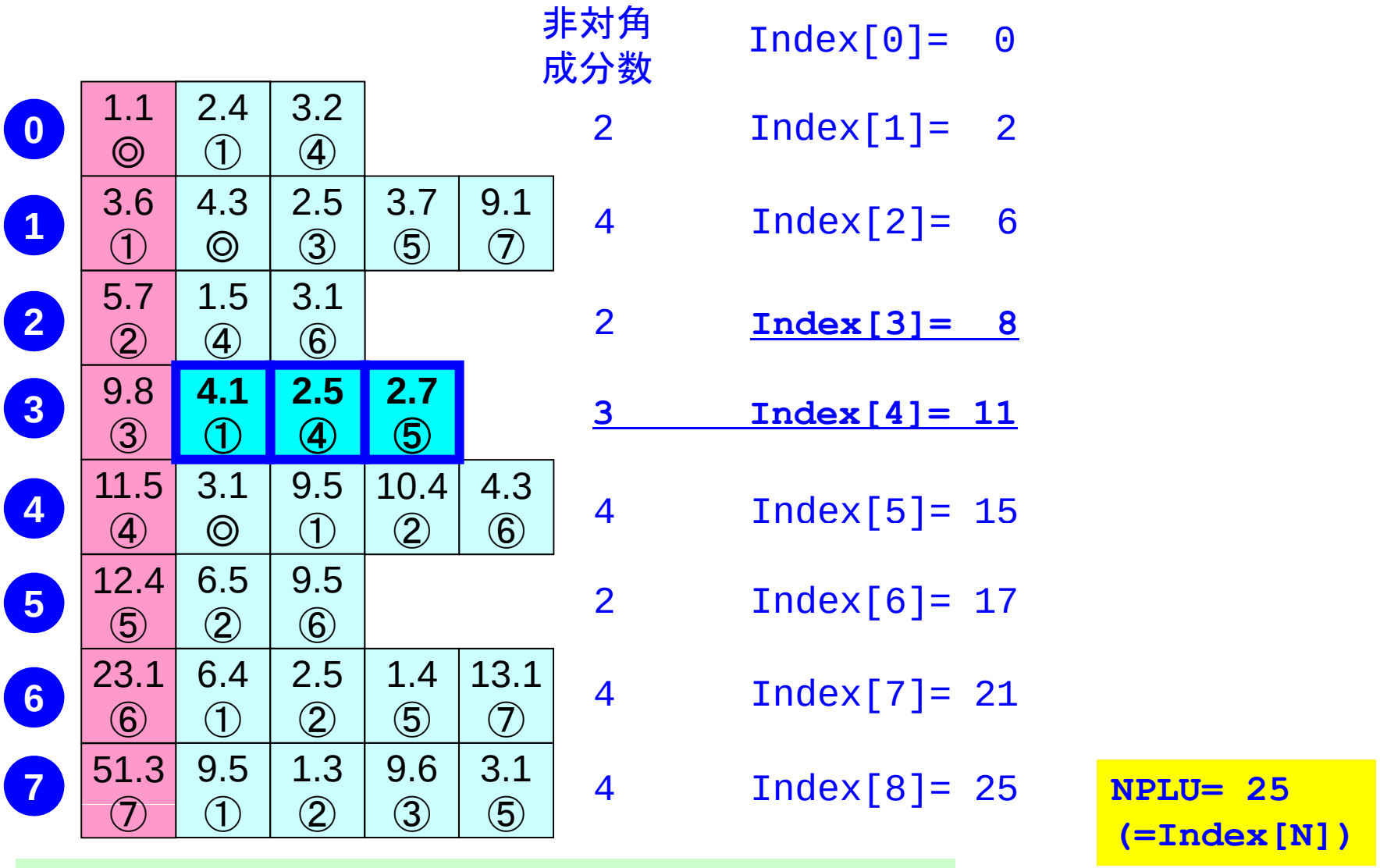
Compressed Row Storage (CRS)：C

						非対角成分数	
							Index[0]= 0
0	1.1 ◎	2.4 ①	3.2 ④			2	Index[1]= 2
1	3.6 ①	4.3 ◎	2.5 ③	3.7 ⑤	9.1 ⑦	4	Index[2]= 6
2	5.7 ②	1.5 ④	3.1 ⑥			2	Index[3]= 8
3	9.8 ③	4.1 ①	2.5 ④	2.7 ⑤		3	Index[4]= 11
4	11.5 ④	3.1 ◎	9.5 ①	10.4 ②	4.3 ⑥	4	Index[5]= 15
5	12.4 ⑤	6.5 ②	9.5 ⑥			2	Index[6]= 17
6	23.1 ⑥	6.4 ①	2.5 ②	1.4 ⑤	13.1 ⑦	4	Index[7]= 21
7	51.3 ⑦	9.5 ①	1.3 ②	9.6 ③	3.1 ⑤	4	Index[8]= 25

NPLU= 25 (=Index[N])

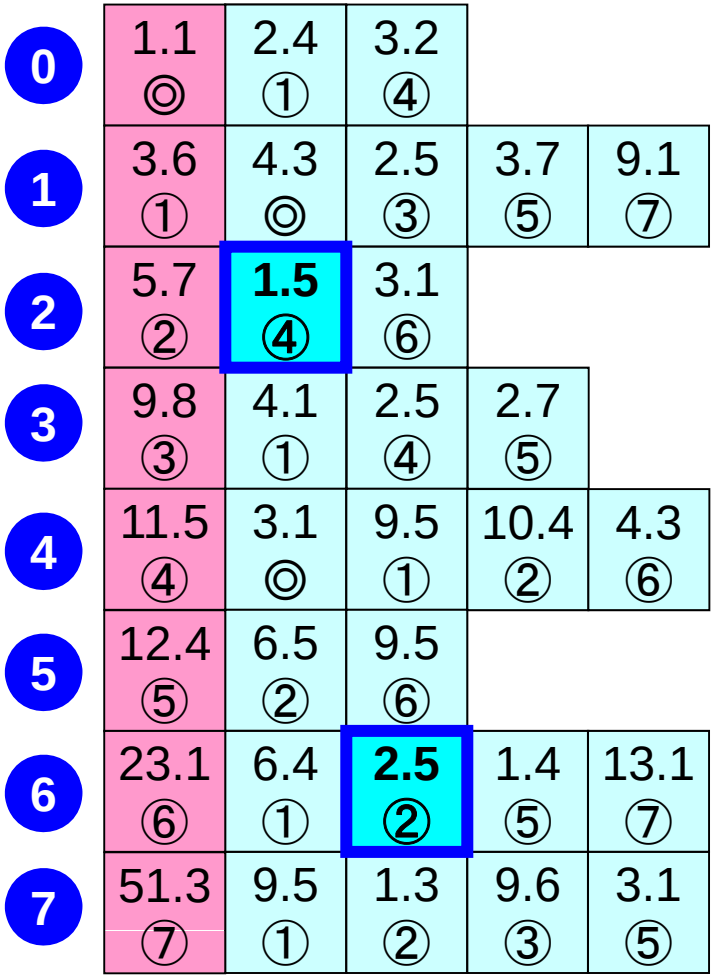
Index[i]~Index[i+1]-1番目がi行目の非対角成分

Compressed Row Storage (CRS)：C



Index[i]~Index[i+1]-1番目がi行目の非対角成分

Compressed Row Storage (CRS)：C



例：

```
Item[ 6]= 4, AMat[ 6]= 1.5
Item[18]= 2, AMat[18]= 2.5
```

Compressed Row Storage (CRS)：C

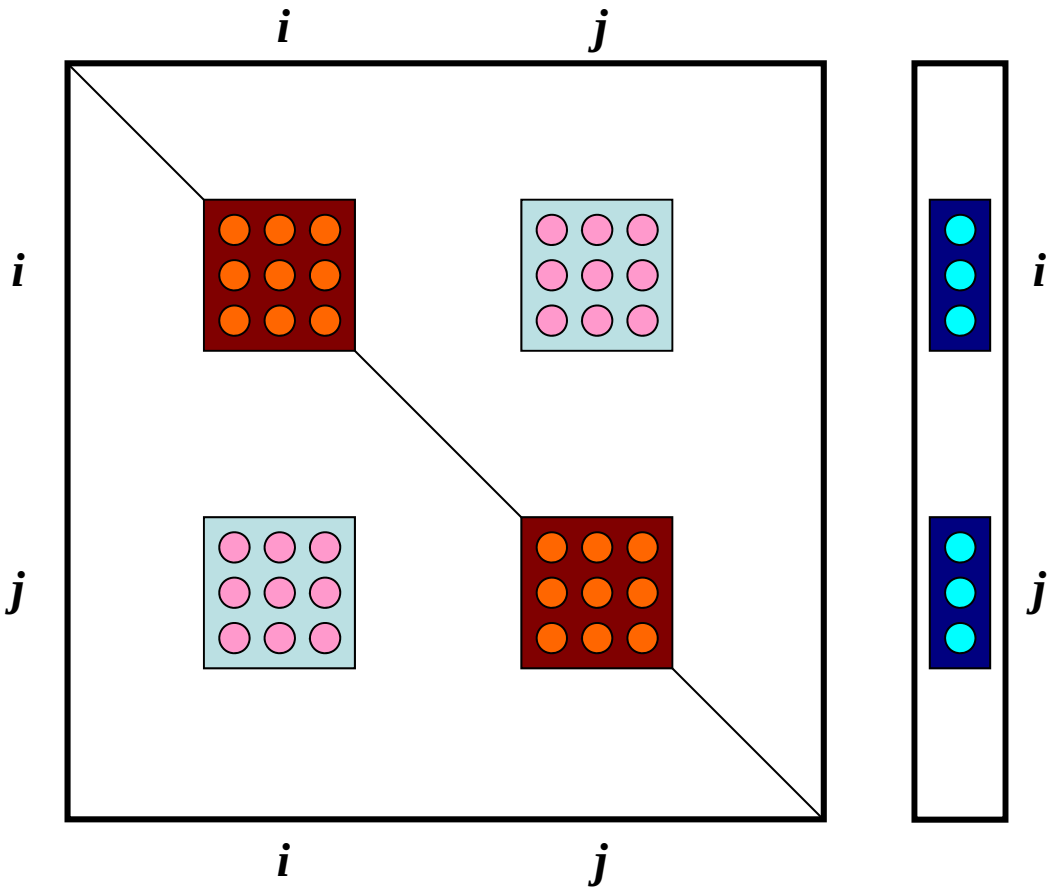
0	1.1 ◎	2.4 ①,0	3.2 ④,1		
1	3.6 ①	4.3 ◎,2	2.5 ③,3	3.7 ⑤,4	9.1 ⑦,5
2	5.7 ②	1.5 ④,6	3.1 ⑥,7		
3	9.8 ③	4.1 ①,8	2.5 ④,9	2.7 ⑤,10	
4	11.5 ④	3.1 ◎,11	9.5 ①,12	10.4 ②,13	4.3 ⑥,14
5	12.4 ⑤	6.5 ②,15	9.5 ⑥,16		
6	23.1 ⑥	6.4 ①,17	2.5 ②,18	1.4 ⑤,19	13.1 ⑦,20
7	51.3 ⑦	9.5 ①,21	1.3 ②,22	9.6 ③,23	3.1 ⑤,24

```
Diag [i]  対角成分(実数, [N])
Index[i]  非対角成分数に関する一次元配列
          (通し番号)(整数, [N+1])
Item[k]   非対角成分の要素(列)番号
          (整数, [Index[N]])
AMat[k]   非対角成分
          (実数, [Index[N]])
```

```
{Y}=[A]{X}

for(i=0;i<N;i++){
    Y[i] = Diag[i] * X[i];
    for(k=Index[i];k<Index[i+1];k++){
        Y[i] += AMat[k]*X[Item[k]];
      }
}
```

ブロックとして記憶（1/3）

- 記憶容量が減る
  - Index, Itemに関する記憶容量を削減できる



ブロックとして記憶（2/3）

- 計算効率
  - 間接参照（メモリに負担）と計算の比が大きくなる
  - ベクトル，スカラー共に効く：2倍以上の性能
    - 連続領域，キャッシュに載る，ループあたりの計算量増加

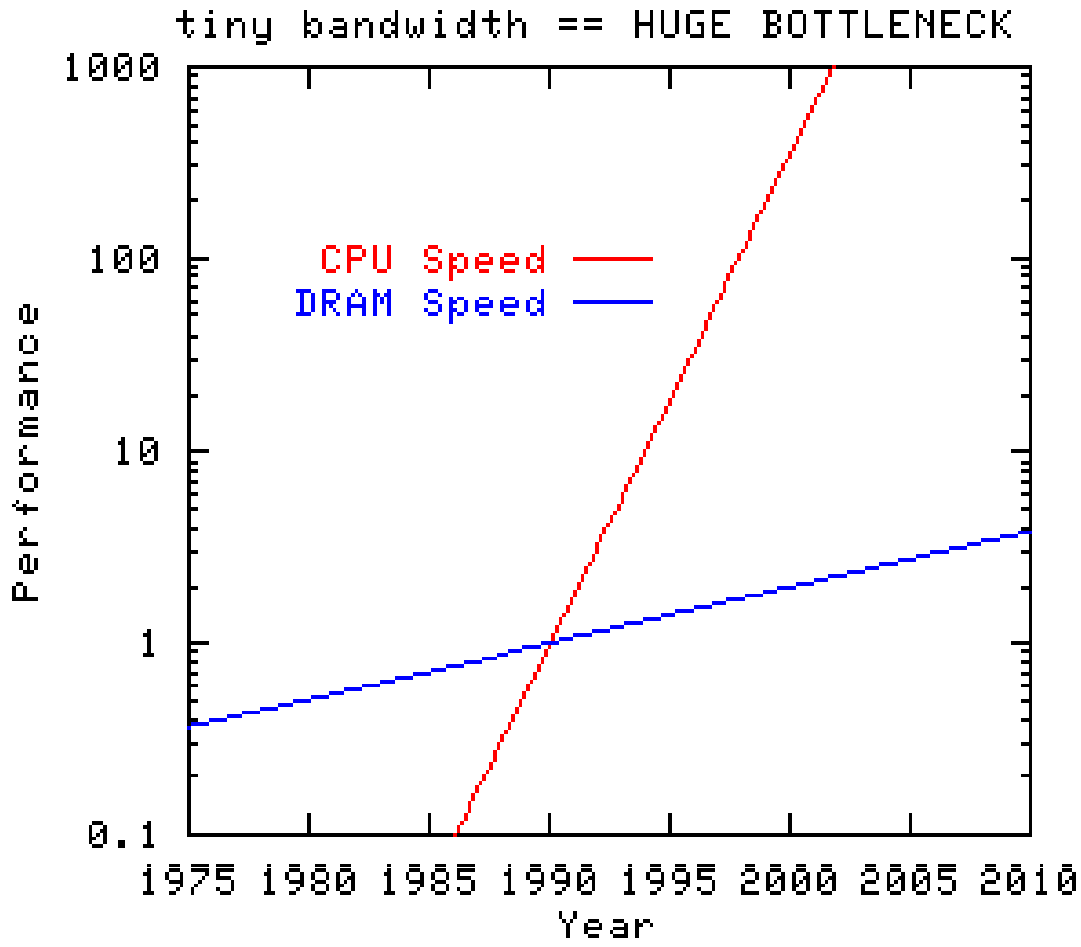
```
do i= 1, 3*N
  Y(i)= D(i)*X(i)
  do k= index(i-1)+1, index(i)
    kk= item(k)
    Y(i)= Y(i) + AMAT(k)*X(kk)
  enddo
enddo
```

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

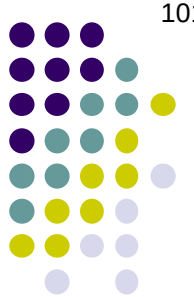
- 三次元要素の定式化
- 三次元弾性力学方程式
  - ガラーキン法
  - 要素マトリクス生成
- 宿題

- プログラムの実行
- データ構造
- プログラムの構成
- **計算効率について**
- ParaViewによる可視化

マイクロプロセッサの動向
CPU性能，メモリバンド幅のギャップ

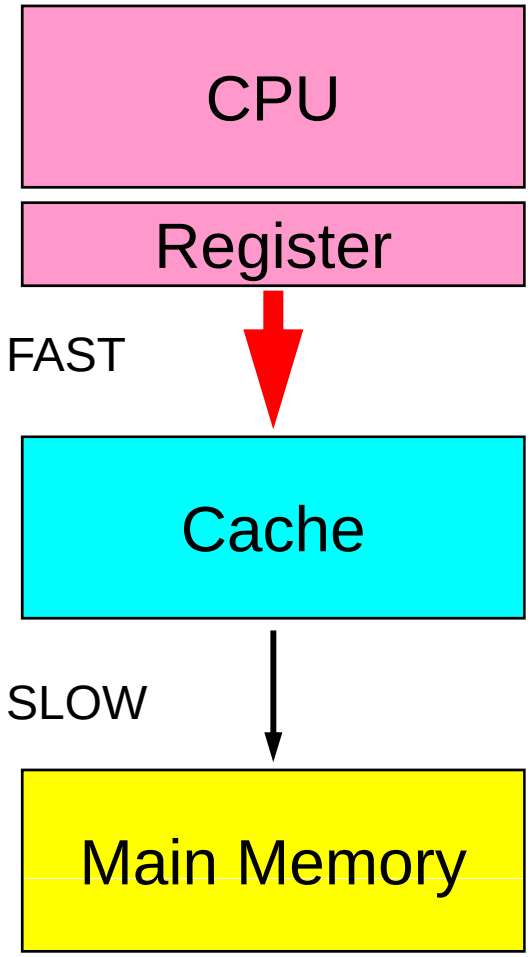


http://www.streambench.org/



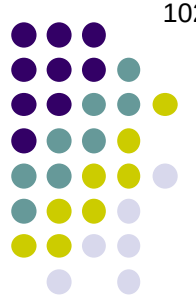
スカラープロセッサ

CPU–キャッシュ–メモリの階層構造

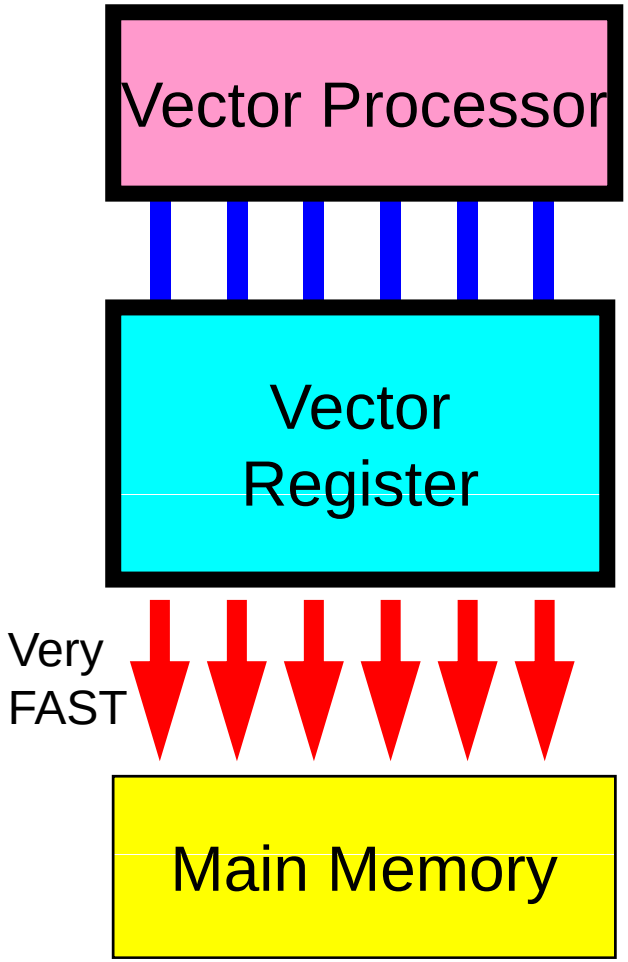


小容量 (MB) : 一時置き場
高価
大きい (1億以上のトランジスタ)

大容量 (GB)
廉価



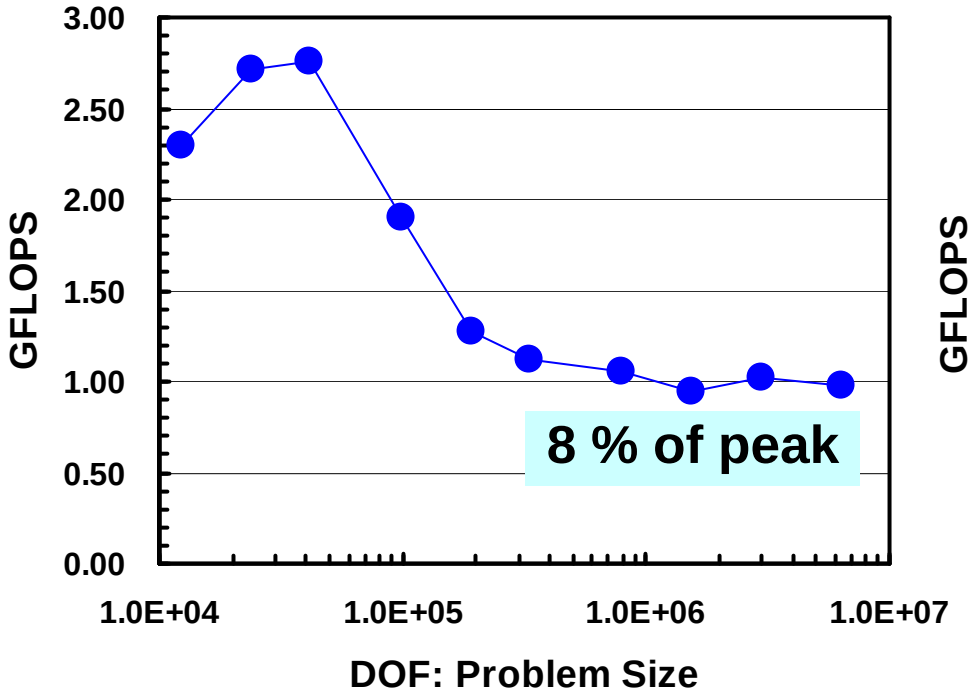
ベクトルプロセッサ
ベクトルレジスタと高速メモリ

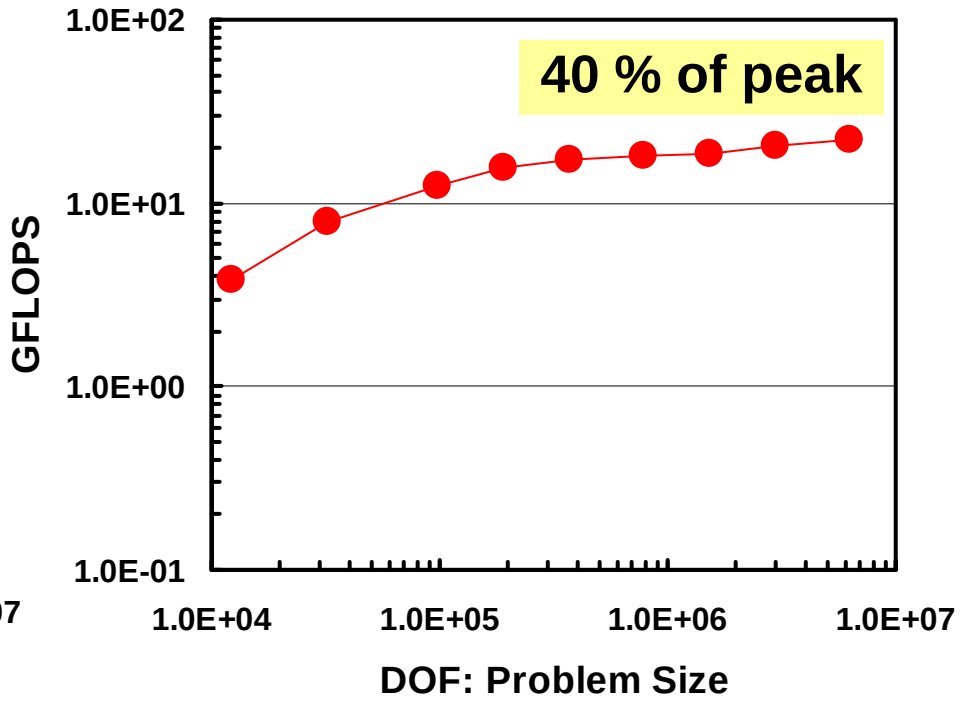


- 単純構造のDOループの並列処理
- 単純，大規模な演算に適している

```
do i= 1, N
  A(i)= B(i) + C(i)
enddo
```

典型的な挙動：CG法（疎行列，FEM）



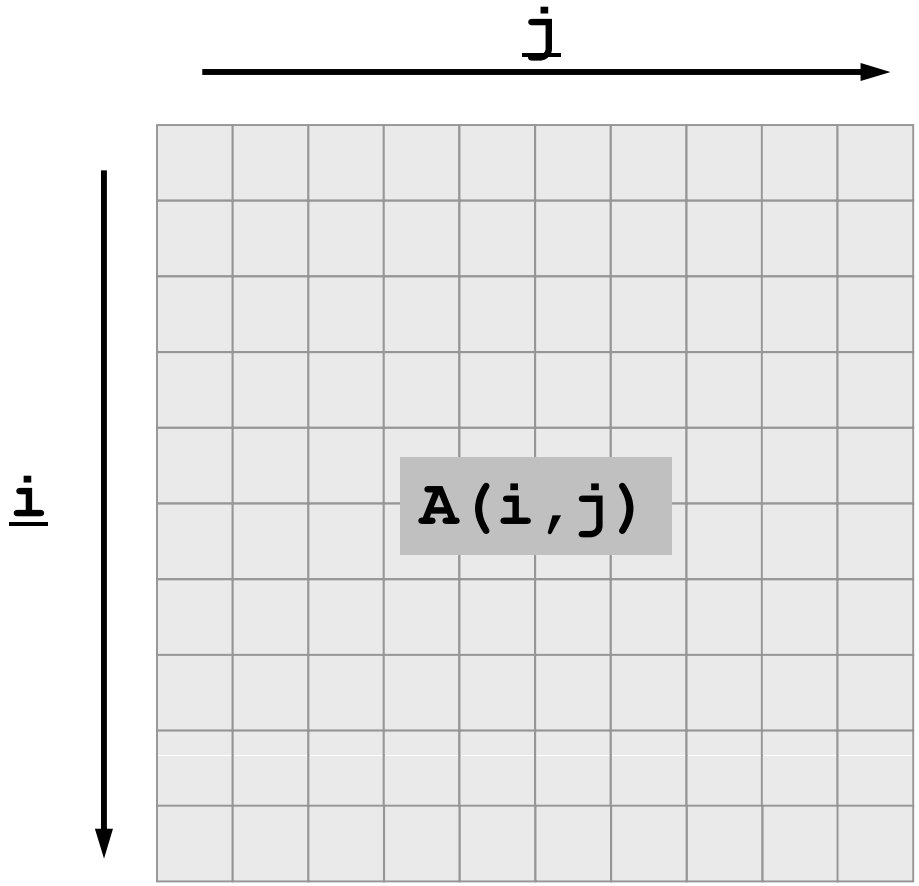


IBM-SP3:
問題サイズが小さい場合はキャッシュの影響のため性能が良い

Earth Simulator:
大規模な問題ほどベクトル長が長くなり，性能が高い

プロセッサに応じたチューニング

- メモリ参照の最適化・・・に尽きる



プロセッサに応じたチューニング（続き）

- ベクトルプロセッサ
  - ループ長を大きくとる。
- スカラープロセッサ
  - キャッシュを有効利用，細切れにしたデータを扱う。
  - PCクラスタなどでは，キャッシュサイズを大きくできない一方で，メモリレイテンシ（立ち上がりオーバーヘッド）の減少，メモリバンド幅の増加の傾向。しかしマルチコア化で帳消し。
- 共通事項
  - メモリアクセスの連続性
  - 局所性
  - 計算順序の変更によって計算結果が変わる可能性について注意すること

スカラープロセッサの代表的なチューニング

- ループアンローリング
  - ループのオーバーヘッドの削減
  - ロード・ストアの削減
- ブロック化
  - キャッシュミスの削減

Scalar

ループアンローリング

ロード・ストアの削減（1/4）

- ループ処理に対する演算の割合が増す。

```
N= 10000

do j= 1, N
  do i= 1, N
    A(i)= A(i) + B(i)*C(i,j)
  enddo
enddo

do j= 1, N-1, 2
  do i= 1, N
    A(i)= A(i) + B(i)*C(i,j)
    A(i)= A(i) + B(i)*C(i,j+1)
  enddo
enddo

do j= 1, N-3, 4
  do i= 1, N
    A(i)= A(i) + B(i)*C(i,j)
    A(i)= A(i) + B(i)*C(i,j+1)
    A(i)= A(i) + B(i)*C(i,j+2)
    A(i)= A(i) + B(i)*C(i,j+3)
  enddo
enddo
```

T2K東大での計算時間

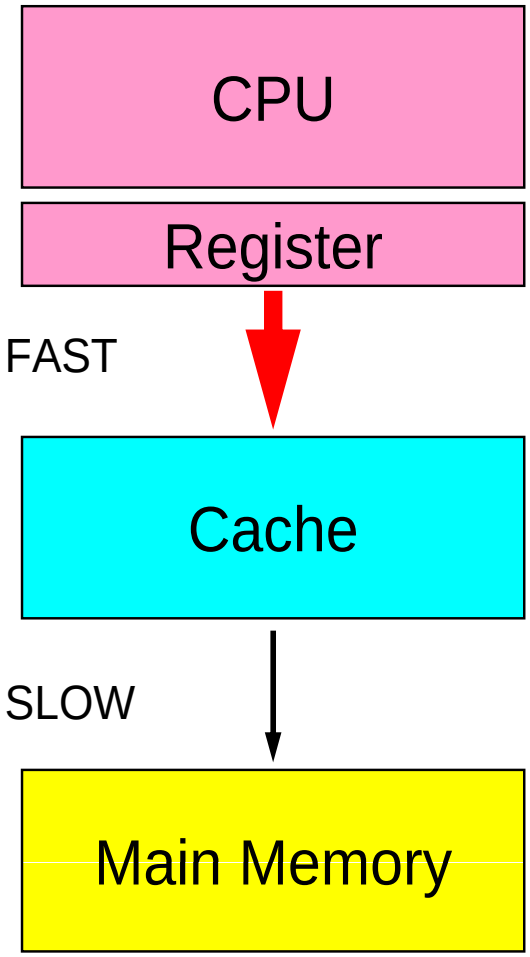
```
4.023438E-01
3.085938E-01
2.617188E-01
```

Scalar

ループアンローリング

ロード・ストアの削減（2/4）

- ロード：メモリ⇒キャッシュ⇒レジスタ
- ストア：ロードの逆
- ロード・ストアが少ないほど効率良い



Scalar

ループアンローリング

ロード・ストアの削減（3/4）

```
do j= 1, N
  do i= 1, N
    A(i)= A(i) + B(i)*C(i,j)
   ストア  ロード   ロード ロード
  enddo
enddo
```

- A(i)，B(i)，C(i,j) に対して，各ループでロード・ストアが発生する。

Scalar

ループアンローリング

ロード・ストアの削減（4/4）

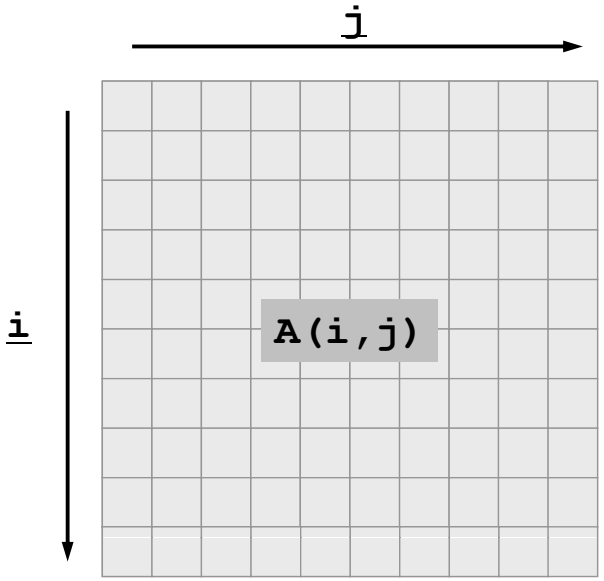
```
do j= 1, N-3, 4
  do i= 1, N
    A(i)= A(i) + B(i)*C(i,j)
          ロード  ロード ロード
    A(i)= A(i) + B(i)*C(i,j+1)
    A(i)= A(i) + B(i)*C(i,j+2)
    A(i)= A(i) + B(i)*C(i,j+3)
    ストア
  enddo
enddo
```

- 同一ループ内で複数回表れている場合には，最初にロード，最後にストアが実施され，その間レジスターに保持される。
- 計算順序に注意。

Scalar

ループ入替によるメモリ参照最適化（1/2）

```
TYPE-A
do i= 1, N
  do j= 1, N
    A(i,j)= A(i,j) + B(i,j)
  enddo
enddo

TYPE-B
do j= 1, N
  do i= 1, N
    A(i,j)= A(i,j) + B(i,j)
  enddo
enddo
```



- FORTRANでは，A(i,j)のアドレスは，A(1,1), A(2,1), A(3,1),…, A(N,1), A(1,2), A(2,2),…, A(1,N), A(2,N),…, A(N,N)のように並んでいる
  - Cは逆：A[0][0], A[0][1], A[0][2], …, A[N-1][0], A[N-1][1],…,A[N-1][N-1]
- この順番にアクセスしないと効率悪い（ベクトル，スカラーに共通）。

Scalar

ループ入替によるメモリ参照最適化（2/2）

TYPE-A

```
do i= 1, N
  do j= 1, N
    A(i,j)= A(i,j) + B(i,j)
  enddo
enddo
```

TYPE-B

```
do j= 1, N
  do i= 1, N
    A(i,j)= A(i,j) + B(i,j)
  enddo
enddo
```

TYPE-A

```
for (j=0; j<N; j++){
  for (i=0; i<N; i++){
    A[i][j]= A[i][j] + B[i][j];
  }
}
```

TYPE-B

```
for (i=0; i<N; i++){
  for (j=0; j<N; j++){
    A[i][j]= A[i][j] + B[i][j];
  }
}
```

T2K東大での計算時間（FORTRAN）

```
### N ###      512
 A     3.125000E-02
 B     3.906250E-03
### N ###     1024
 A     2.343750E-01
 B     7.812500E-03
### N ###     1536
 A     3.476563E-01
 B     1.562500E-02
### N ###     2048
 A     9.296875E-01
 B     3.125000E-02
### N ###     2560
 A     9.687500E-01
 B     4.687500E-02
### N ###     3072
 A     2.152344E+00
 B     7.421875E-02
### N ###     3584
 A     1.921875E+00
 B     9.765625E-02
### N ###     4096
 A     3.804688E+00
 B     1.250000E-01
```

Scalar

ブロック化によるキャッシュミス削減（1/7）

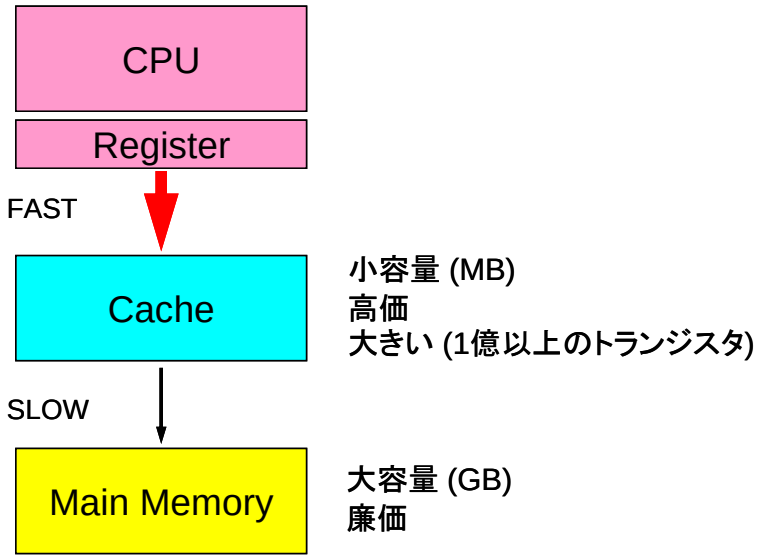
```
do i= 1, NN
  do j= 1, NN
    A(j,i)= A(j,i) + B(i,j)
  enddo
enddo
```

```
for (j=0;j<NN;j++){
  for (i=0;i<NN;i++){
    A[j][i]= A[j][i] + B[i][j];
  }
}
```

- 計算の処理の都合でこのような順番で計算せざるを得ない場合。

キャッシュの有効利用

- キャッシュ
  - 実際は64byte～128byteの細かいキャッシュラインに分かれている。
  - キャッシュライン単位でメモリへのリクエストが行われる。

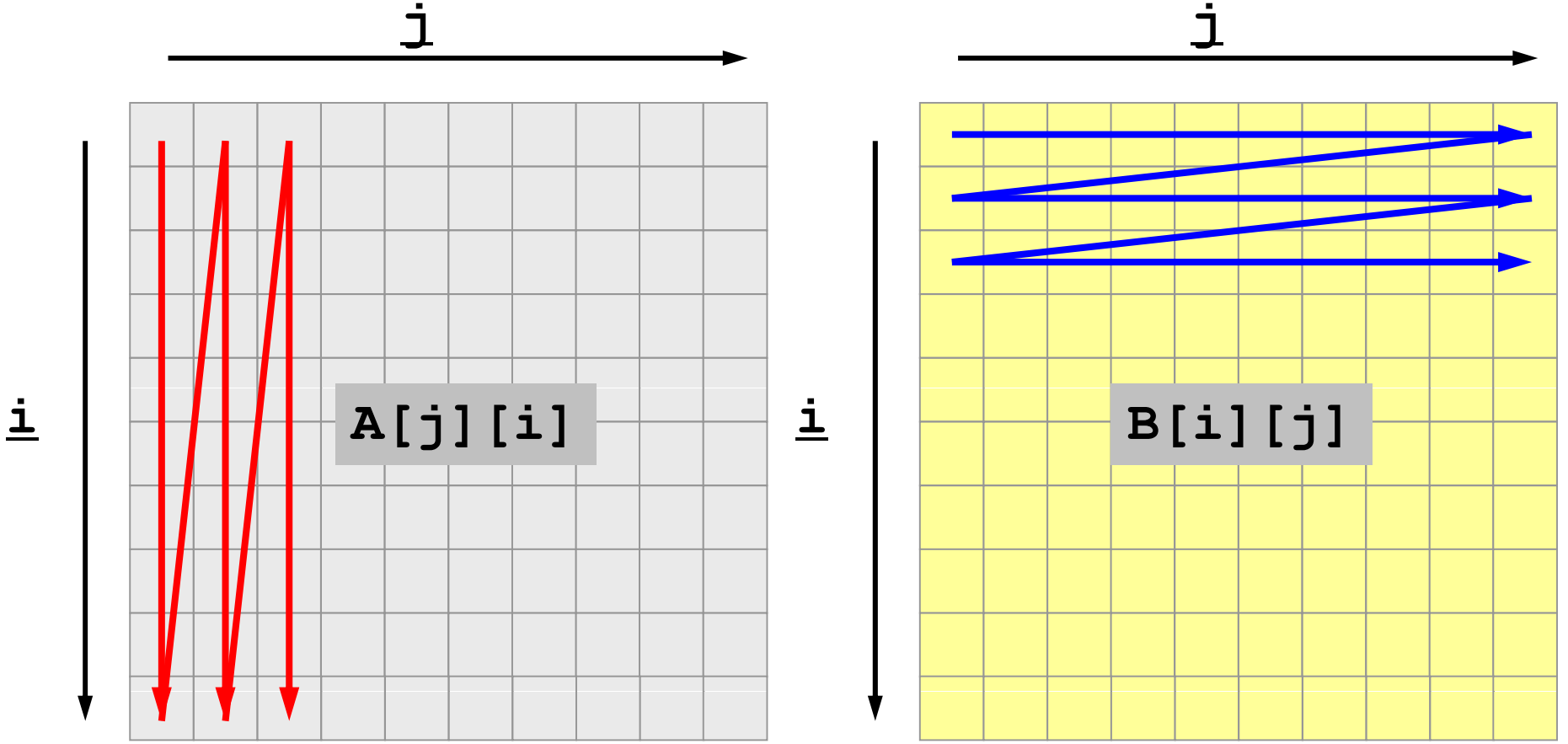


- TLB（Translate Lookaside Buffer）
  - アドレス変換バッファ
    - 仮想アドレスから実アドレスへの変換機能
  - TLB用のキャッシュ
    - 通常128×8 kbyte程度：リンク時に可変
  - 「キャッシュ」が有効に使われていればTLBミスも起こりにくい

Scalar

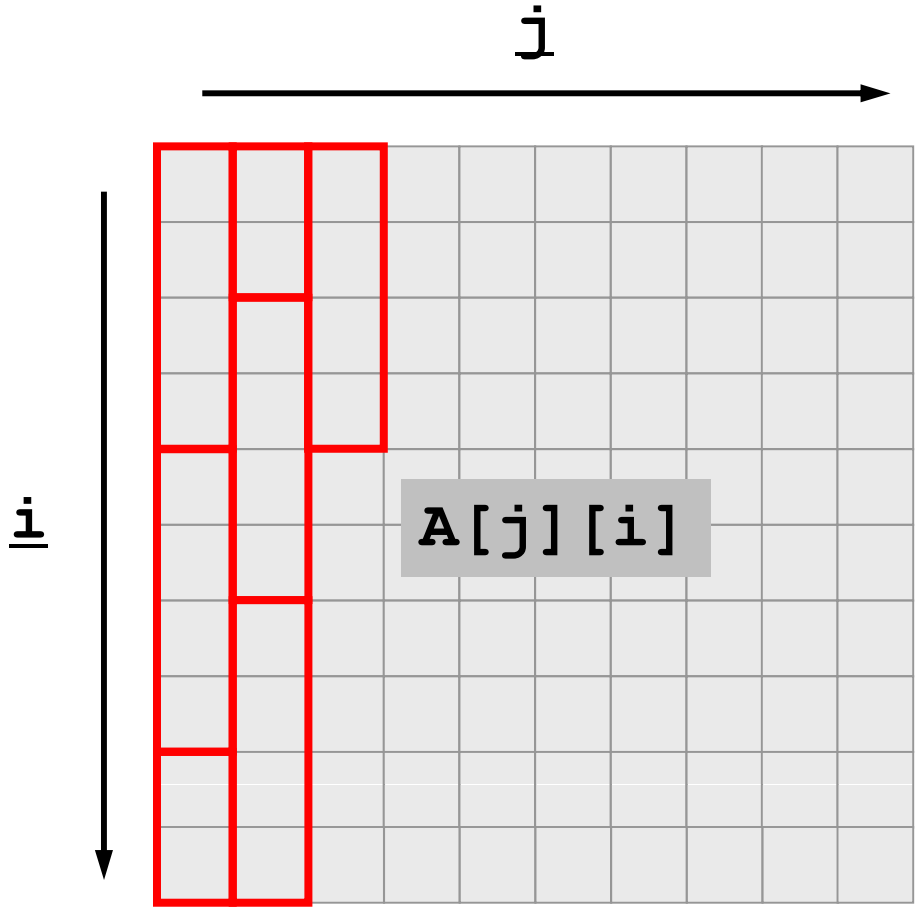
ブロック化によるキャッシュミス削減（2/7）

- A，Bでメモリアクセスパターンが相反
  - 特にBは効率が悪い



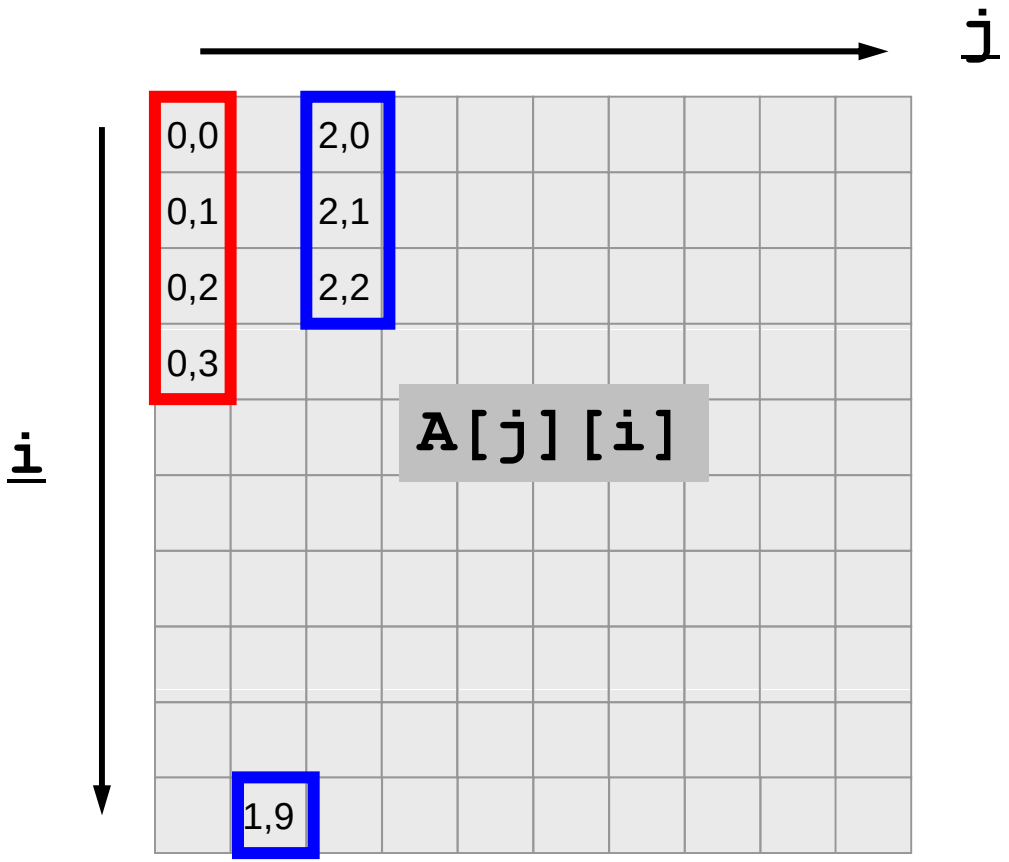
ブロック化によるキャッシュミス削減（3/7）

- 例えば，キャッシュラインサイズを4ワードとすると配列の値は以下のようにキャッシュに転送される。



ブロック化によるキャッシュミス削減（4/7）

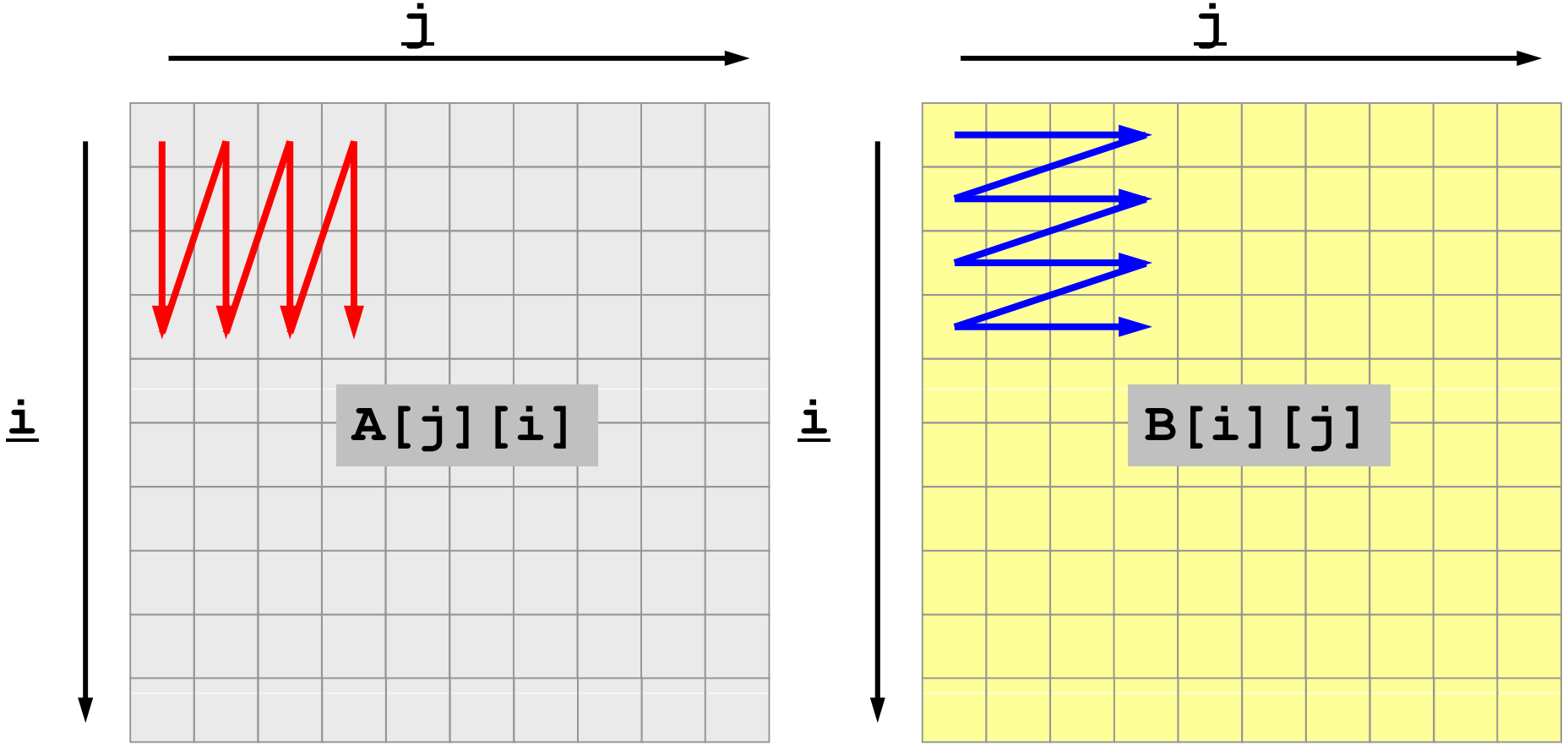
- したがって，A[0][0]をアクセスしたら，A[0][0], A[0][1], A[0][2], A[0][3]が，A[1][9]をアクセスしたらA[1][9], A[2][0], A[2][1], A[2][2]がそれぞれキャッシュ上にあるということになる。



Scalar

ブロック化によるキャッシュミス削減（5/7）

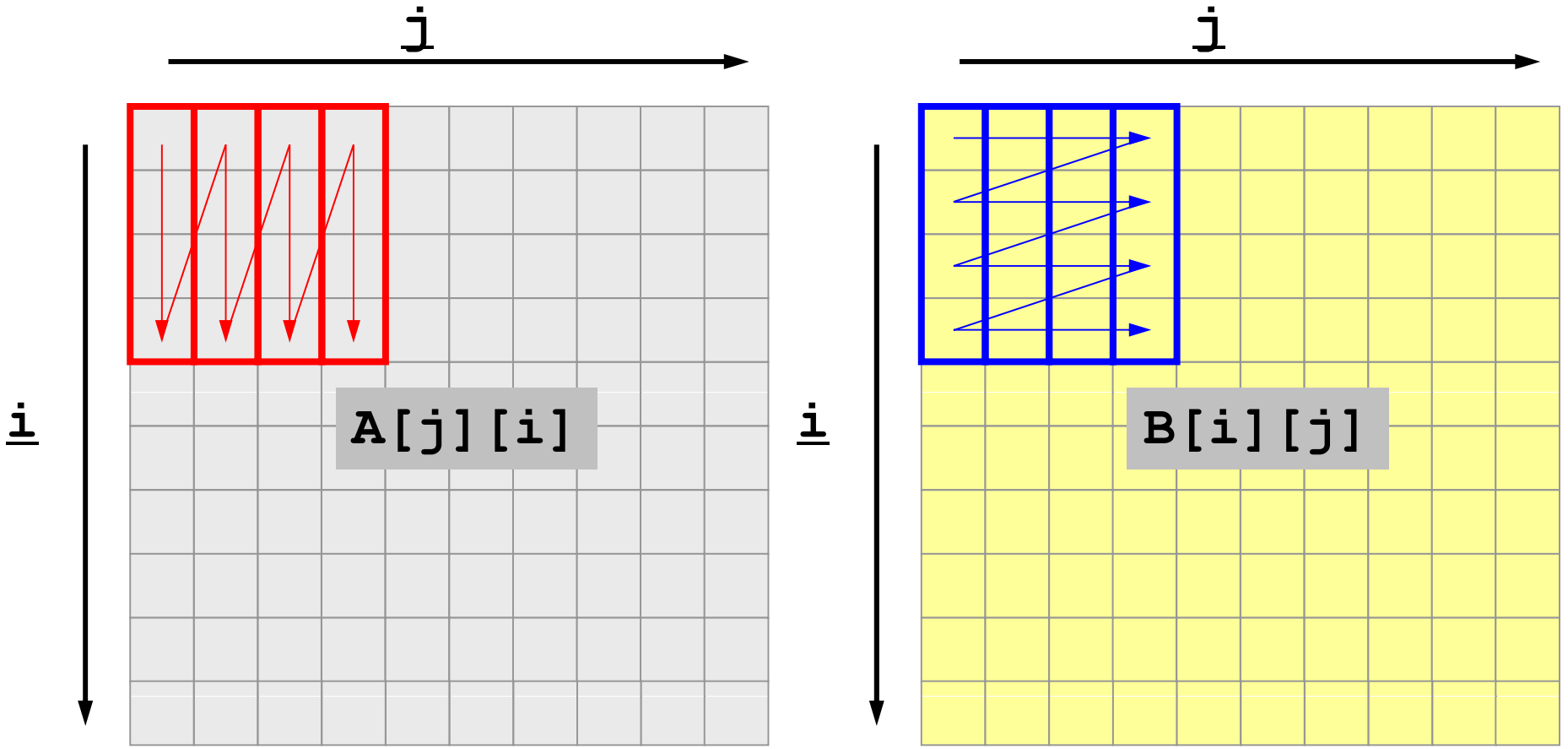
- したがって，以下のようなブロック型のパターンでアクセスすれば，キャッシュを有効利用できる



Scalar

ブロック化によるキャッシュミス削減（6/7）

- □，□で囲んだ部分はキャッシュに載っている。



Scalar

ブロック化によるキャッシュミス削減（7/7）

- ２×２ブロック

```
for (j=0;j<NN;j=++){
  for (i=0;i<NN;i=++){
    A[j][i]= A[j][i] + B[i][j];
  }
}


for (j=0;j<NN-1;j=+2){
  for (i=0;i<NN-1;i=+2){
    A[j  ][i  ]= A[j  ][i  ] + B[i  ][j  ];
    A[j  ][i+1]= A[j  ][i+1] + B[i  ][j+1];
    A[j+1][i  ]= A[j+1][i  ] + B[i+1][j  ];
    A[j+1][i+1]= A[j+1][i+1] + B[i+1][j+1];
  }
}
```

T2K東大での実行時間（FORTRAN）

```
### N ###     1024
 BASIC      2.734375E-02
 2x2        1.562500E-02

### N ###     1536
 BASIC      6.250000E-02
 2x2        3.515625E-02

...

### N ###     3072
 BASIC      2.578125E-01
 2x2        1.484375E-01

### N ###     3584
 BASIC      3.710938E-01
 2x2        2.031250E-01

### N ###     4096
 BASIC      8.437500E-01
 2x2        4.375000E-01
```

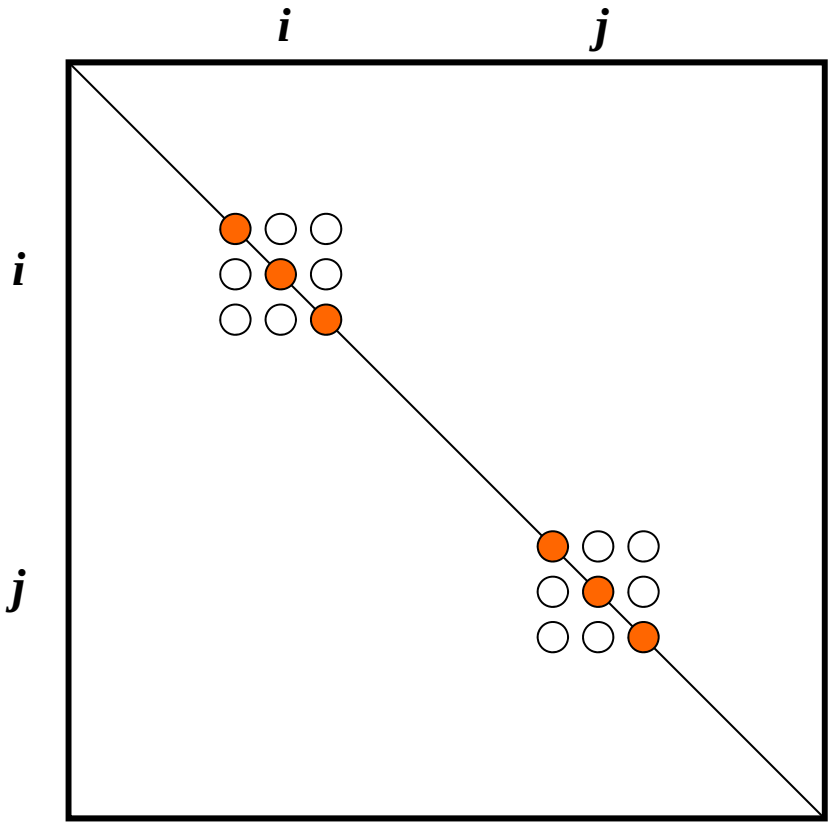
チューニング：まとめ

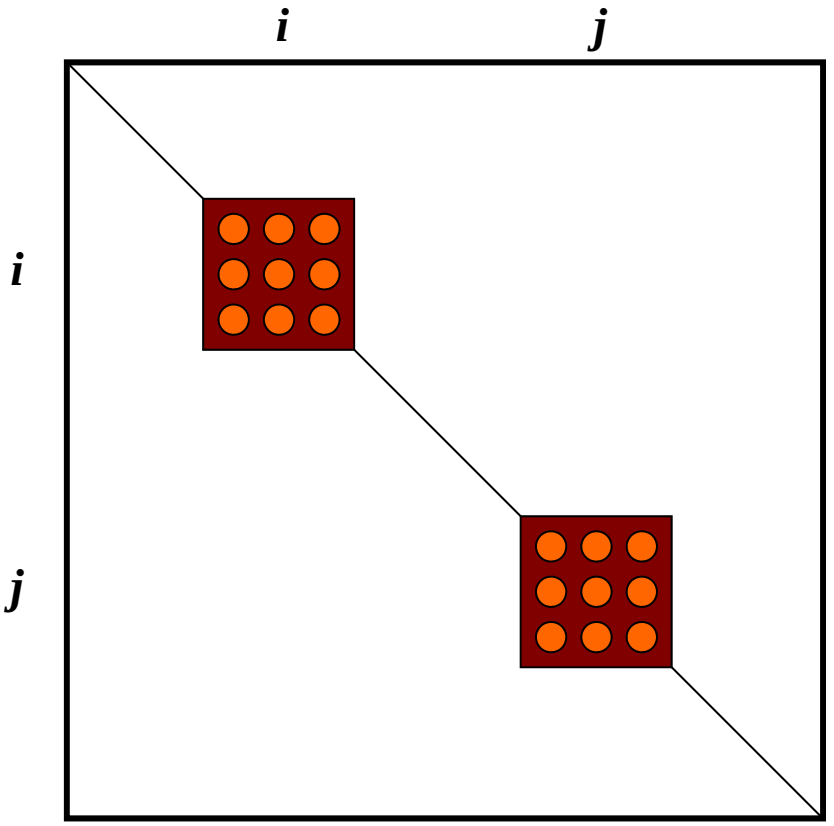
- スカラープロセッサ
- 密行列

- 疎行列の場合はもっと難しい（研究途上の課題）
  - しかし，基本的な考え方は変わらない
  - メモリアクセスの効率化

ブロックとして記憶（3/3）

- 計算の安定化
  - 対角成分で割るのではなく，対角ブロックの完全LU分解を求めて解く
  - 特に悪条件問題で有効





- 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
- 宿題

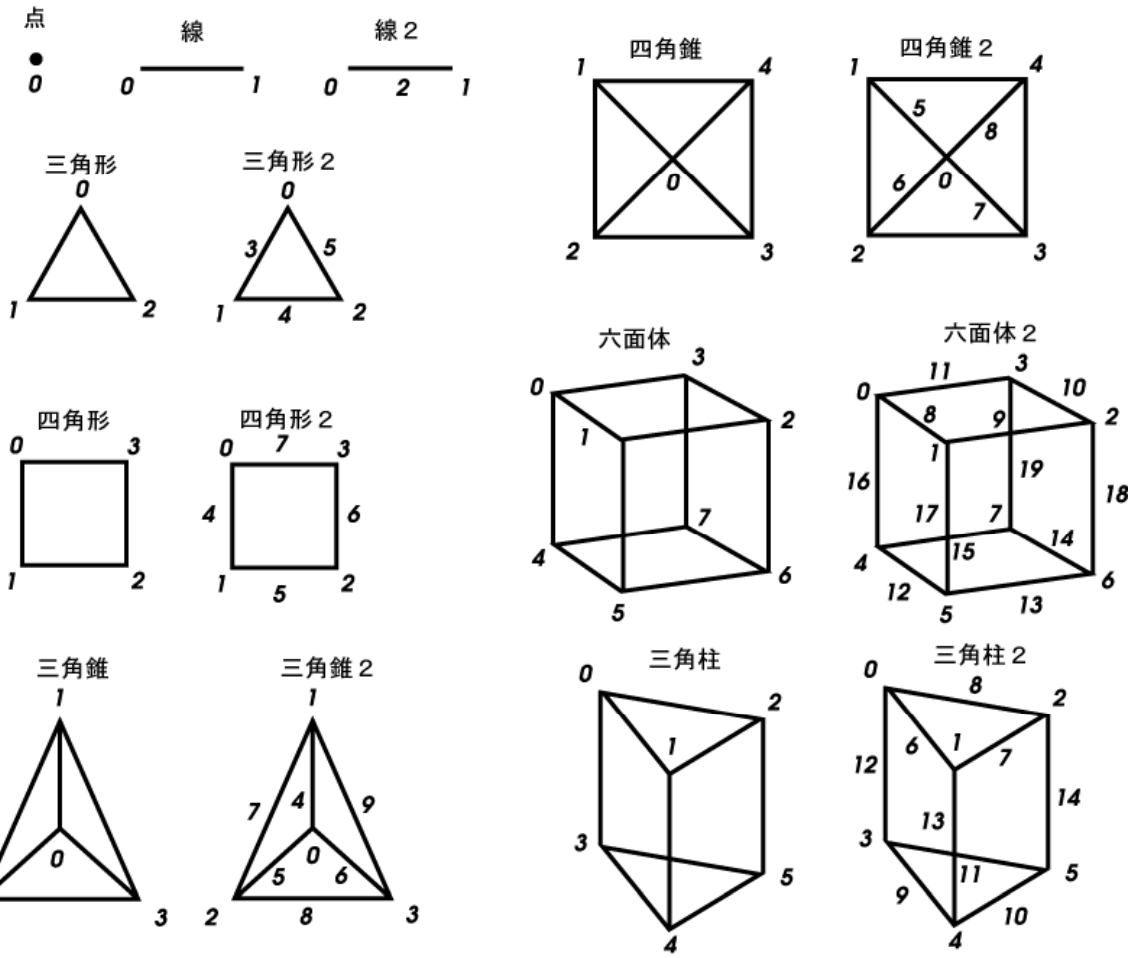
- プログラムの実行
- データ構造
- プログラムの構成
- 計算効率について
- **ParaViewによる可視化**

ParaView

- ファイルを開く
- 図の表示
- イメージファイルの保存

UCD Format (1/3)
Unstructured Cell Data

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



UCD Format (2/3)

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

UCD Format (3/3): Old Format

(全節点数) (全要素数) (各節点のデータ数) (各要素のデータ数) (モデルのデータ数)
(節点番号1) (X座標) (Y座標) (Z座標)
(節点番号2) (X座標) (Y座標) (Z座標)

.
.
.

(要素番号1) (材料番号) (要素の種類) (要素を構成する節点のつながり)
(要素番号2) (材料番号) (要素の種類) (要素を構成する節点のつながり)

.
.
.

(節点のデータ成分数) (成分1の構成数) (成分2の構成数) ・・・(各成分の構成数)
(節点データ成分1のラベル), (単位)
(節点データ成分2のラベル), (単位)

.
.
.

(各節点データ成分のラベル), (単位)
(節点番号1) (節点データ1) (節点データ2) ・・・・・
(節点番号2) (節点データ1) (節点データ2) ・・・・・

.
.
.

(要素のデータ成分数) (成分1の構成数) (成分2の構成数) ・・・(各成分の構成数)
(要素データ成分1のラベル), (単位)
(要素データ成分2のラベル), (単位)

.
.
.

(各要素データ成分のラベル), (単位)
(要素番号1) (要素データ1) (要素データ2) ・・・・・
(要素番号2) (要素データ1) (要素データ2) ・・・・・

.
.
.