OpenMPによるマルチコア・メニィコア 並列プログラミング入門 C言語編

Part-B1：ICCG法による求解

中島研吾

東京大学情報基盤センター

ファイルの用意 on your PC

コピー，展開

http://nkl.cc.u-tokyo.ac.jp/files/multicore-c.tar

```
>$ cd
>$ tar xvf multicore-c.tar

>$ cd multicore
```

以下のディレクトリが出来ていることを確認

L1 L2

これらを以降 **<$P-L1>**，**<$P-L2>**

PC

Odyssey

- 背景
  - 有限体積法
  - 前処理付反復法
- ICCG法によるポアソン方程式法ソルバーについて
  - 実行方法
    - データ構造
  - プログラムの説明
    - 初期化
    - 係数マトリクス生成
    - ICCG法

本編の目的より

- 「有限体積法から導かれる疎行列を対象としたICCG法」を題材とした，データ配置，reorderingなど，科学技術計算のためのマルチコアプログラミングにおいて重要なアルゴリズムについての講習

- 有限体積法
- 疎行列
- ICCG法

対象とするアプリケーションの概要

- 支配方程式：三次元ポアソン方程式

$$\frac{\partial^2 \phi}{\partial x^2}+\frac{\partial^2 \phi}{\partial y^2}+\frac{\partial^2 \phi}{\partial z^2}+f=0$$

- 有限体積法（Finite Volume Method, **FVM**）による空間離散化
  - 任意形状の要素，要素中心で変数を定義。
  - 直接差分法（Direct Finite Difference Method）とも呼ばれる。
- 境界条件他
  - ディリクレ境界条件@Z=Zmax，体積フラックスf
- 反復法による連立一次方程式解法
  - 共役勾配法（CG）＋前処理

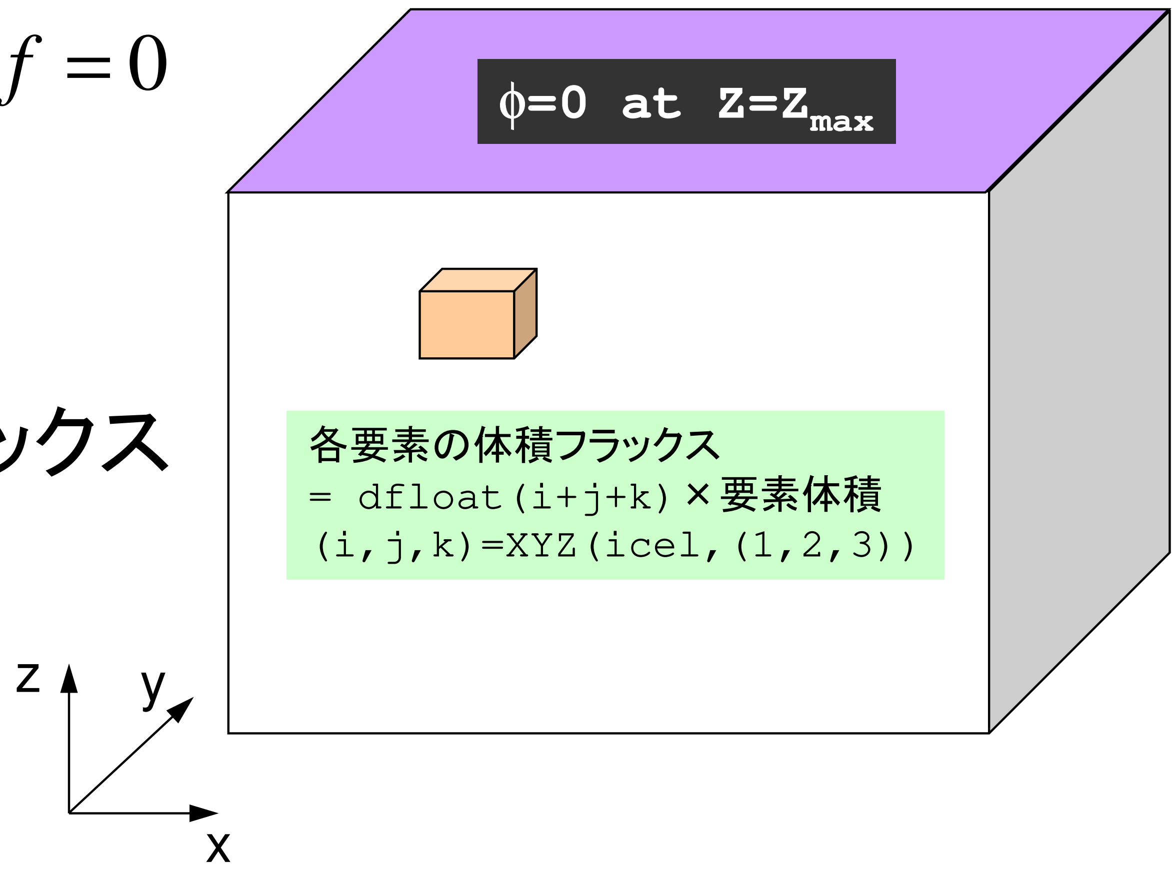
解いている問題：三次元ポアソン方程式 変数：要素中心で定義

ポアソン方程式

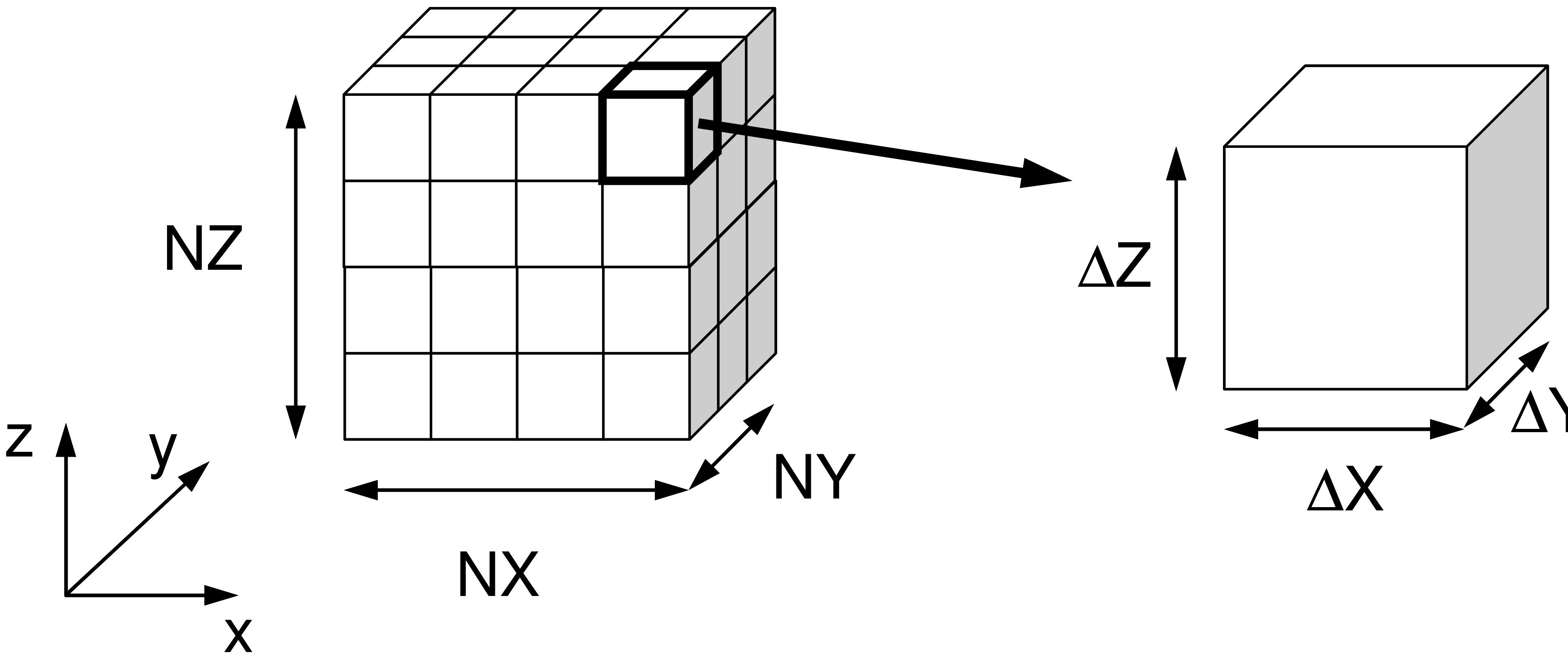
$$\frac{\partial^2 \phi}{\partial x^2}+\frac{\partial^2 \phi}{\partial y^2}+\frac{\partial^2 \phi}{\partial z^2}+f=0$$

<u>境界条件他</u>

- 各要素で体積フラックス
- $Z=Z_{max}$面で$\phi=0$



対象：規則正しい三次元差分格子
半非構造的に扱う



体積フラックスfの内容

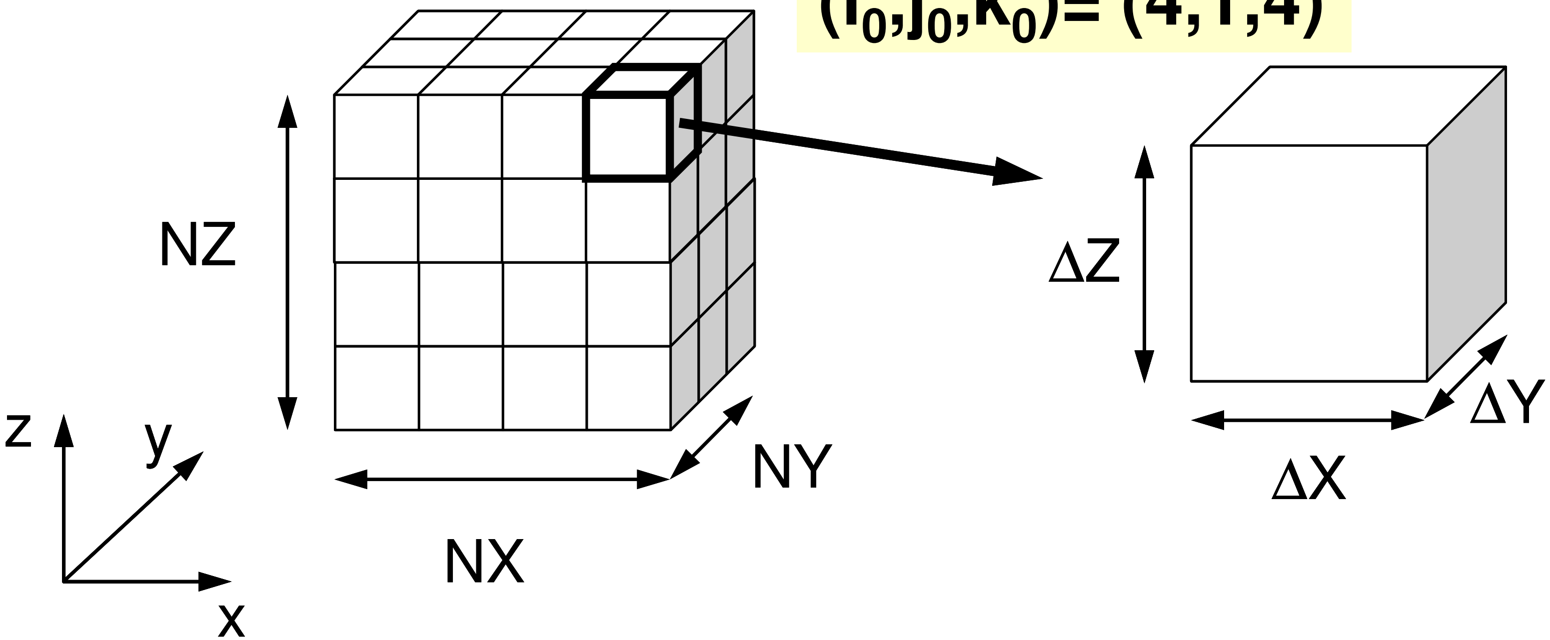
$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}+\frac{\partial^2\phi}{\partial z^2}+f=0$$

$$f=dfloat(i_0+j_0+k_0)$$

$$i_0=XYZ(icel,1),\quad j_0=XYZ(icel,2),\quad k_0=XYZ(icel,3)$$

$XYZ(icel,k)$（k=1,2,3）は
X, Y, Z方向の差分格子のインデックス
各メッシュがX, Y, Z方向の何番目にあるかを示している。

(i_0,j_0,k_0)= (4,1,4)

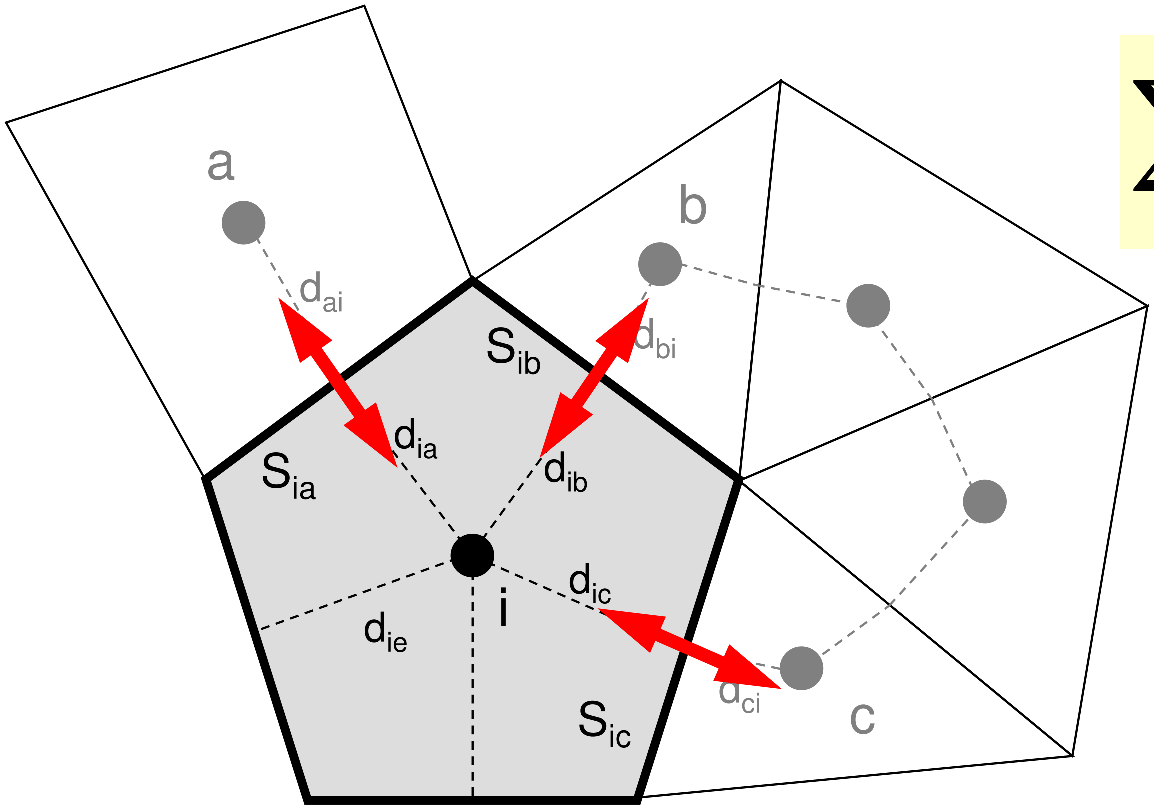


ポアソン方程式：有限体積法による離散化

$$\frac{\partial^2 \phi}{\partial x^2}+\frac{\partial^2 \phi}{\partial y^2}+\frac{\partial^2 \phi}{\partial z^2}+f=0$$

Poisson Eq. by Finite Volume Method（FVM）

面を通過するフラックス（flux，流束）の保存に着目



隣接要素との拡散

$$\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\left(\phi_k-\phi_i\right)+V_i\dot{Q}_i=0$$

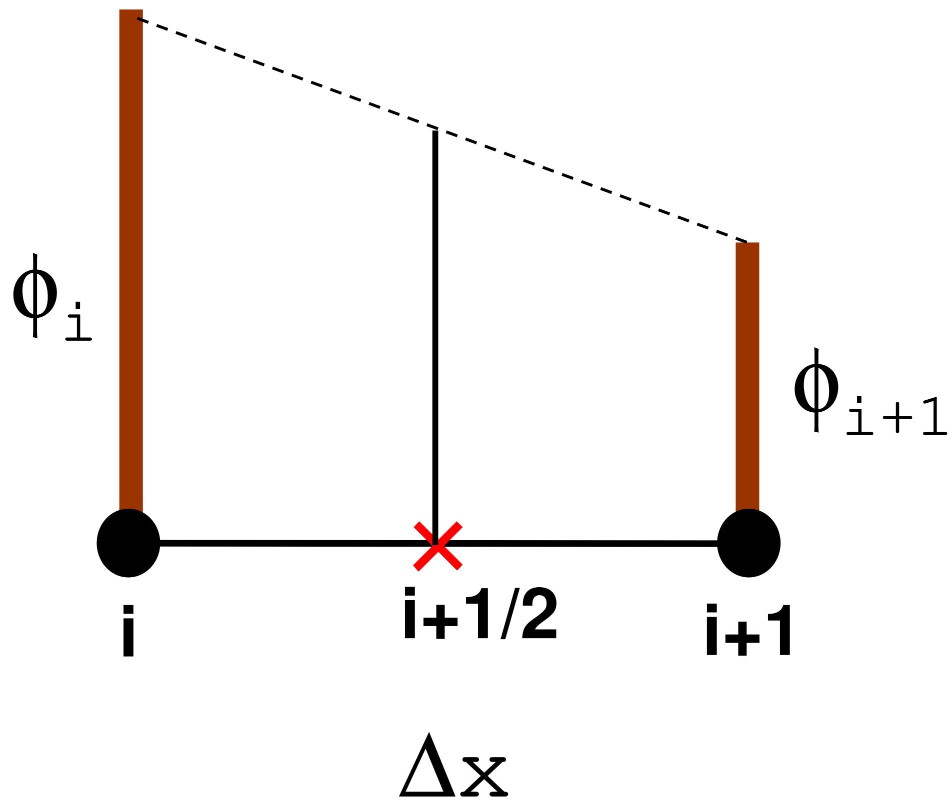
体積フラックス

- V_i ：要素体積
- S ：表面面積
- d_{ij} ：要素中心から表面までの距離
- Q ：体積フラックス

Finite Difference Method (FDM)
(有限)差分法：巨視的微分
macroscopic differentiation

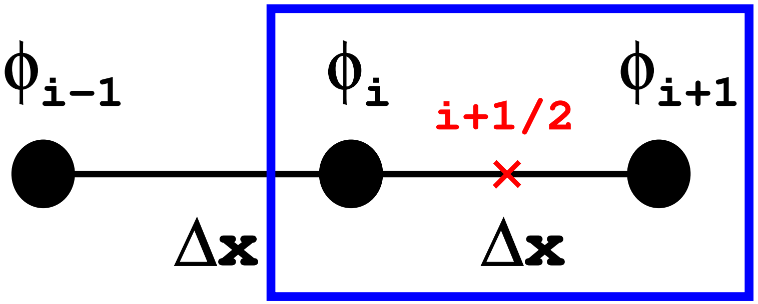
$$\left(\frac{d\phi}{dx}\right)_{i+1/2} \approx \frac{\phi_{i+1}-\phi_i}{\Delta x}$$

$$\left(\frac{d\phi}{dx}\right)_{i+1/2} = \lim_{\Delta x \to 0} \frac{\phi_{i+1}-\phi_i}{\Delta x}$$



2nd Order Differentiation in FDM Taylor Series Expansion

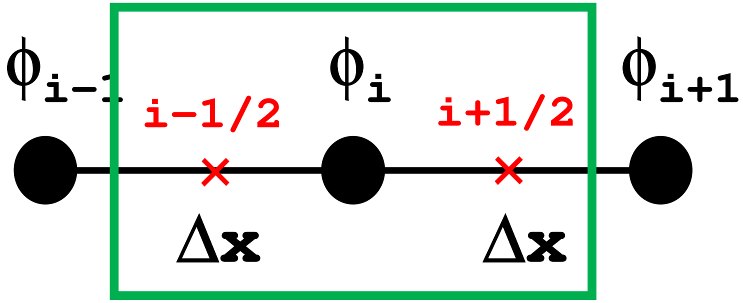
- **Approximate Derivative at × (center of i and $i+1$)**



$$\left(\frac{d\phi}{dx}\right)_{i+1/2} \approx \frac{\phi_{i+1}-\phi_i}{\Delta x}$$

Δx→0: Real Derivative

- **2nd-Order Diff. at i**



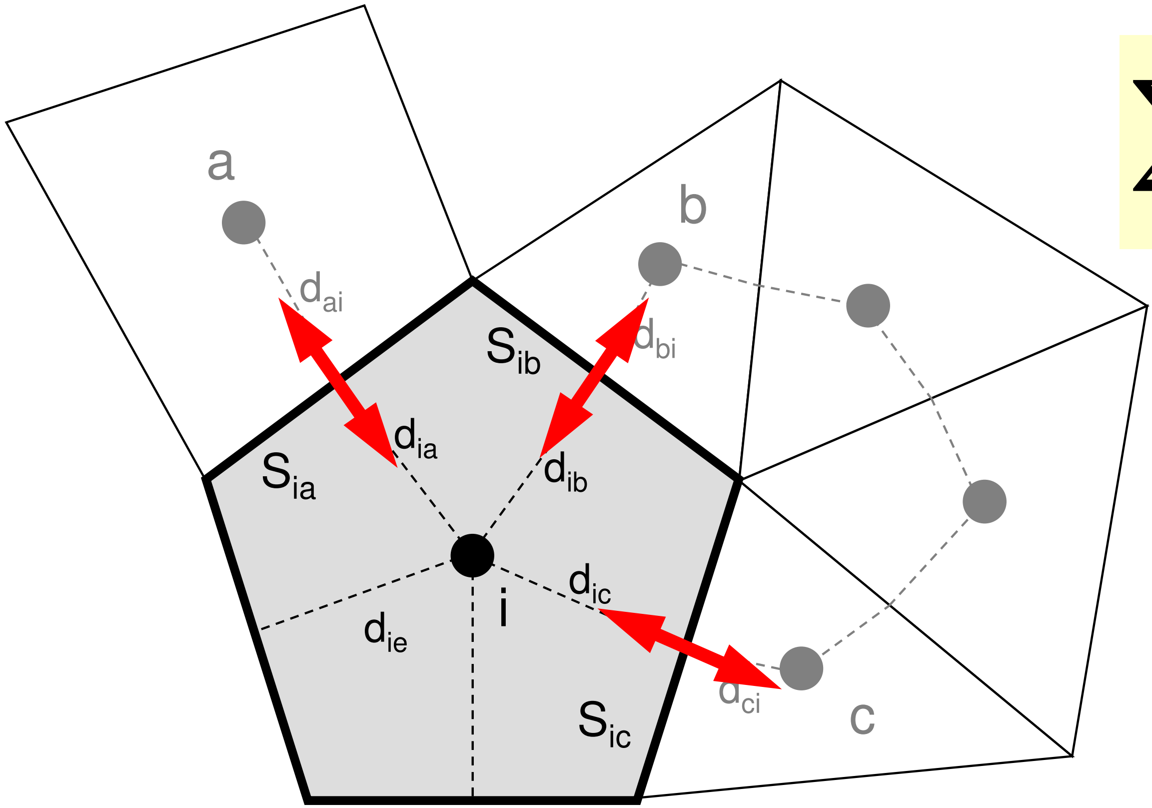
$$\left(\frac{d^2\phi}{dx^2}\right)_i \approx \frac{\left(\frac{d\phi}{dx}\right)_{i+1/2}-\left(\frac{d\phi}{dx}\right)_{i-1/2}}{\Delta x} = \frac{\frac{\phi_{i+1}-\phi_i}{\Delta x}-\frac{\phi_i-\phi_{i-1}}{\Delta x}}{\Delta x} = \frac{\phi_{i+1}-2\phi_i+\phi_{i-1}}{\Delta x^2}$$

ポアソン方程式：有限体積法による離散化

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}+\frac{\partial^2\phi}{\partial z^2}+f=0$$

Poisson Eq. by Finite Volume Method（FVM）

面を通過するフラックス（flux，流束）の保存に着目



隣接要素との拡散

$$\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\left(\phi_k-\phi_i\right)+V_i\dot{Q}_i=0$$

体積フラックス

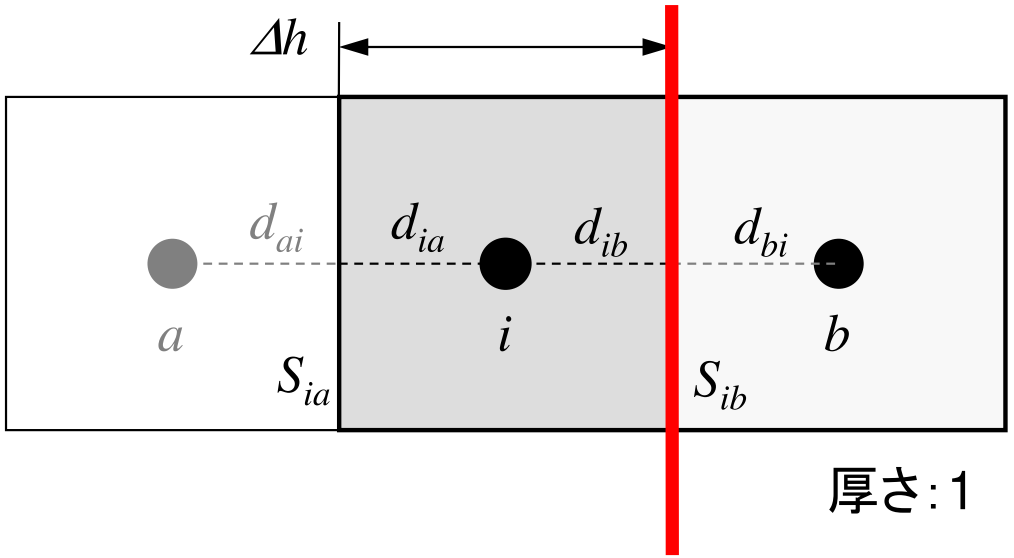
V_i ：要素体積

S ：表面面積

d_{ij} ：要素中心から表面までの距離

Q ：体積フラックス

一次元差分法との比較(1/3)



一辺の長さΔhの正方形メッシュ

接触面積:　$S_{ik} = \Delta h$

要素体積:　$V_i = \Delta h^2$

接触面までの距離:　$d_{ij} = \Delta h/2$

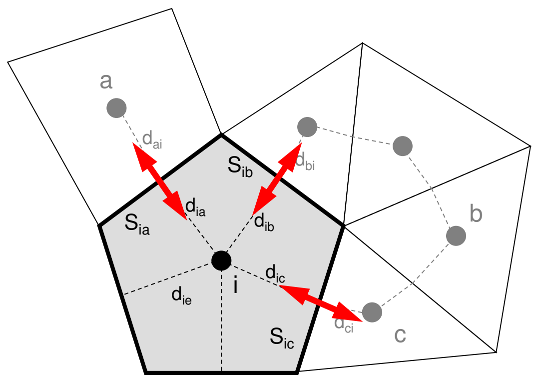
この面を通過するフラックス:Qs_{ib}

$$Qs_{ib} = -\frac{\phi_b - \phi_i}{d_{ib} + d_{bi}} \cdot S_{ib}$$

フーリエ(Fourier)の法則

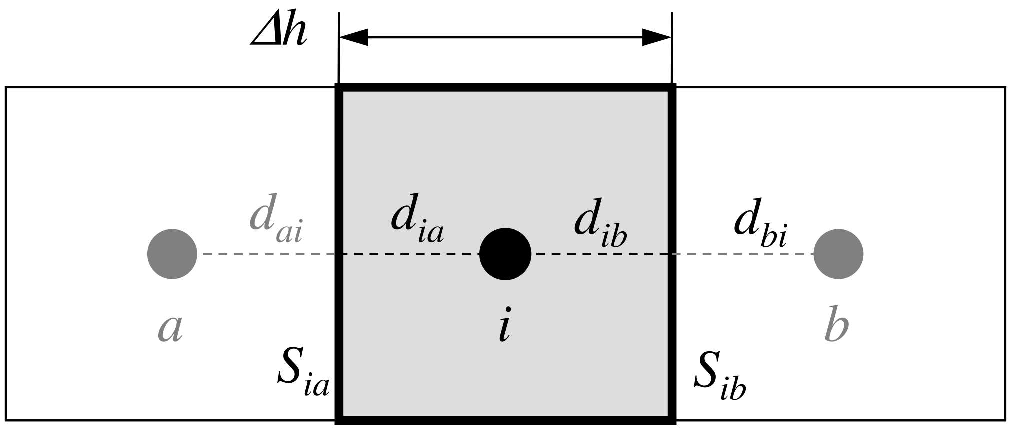
面を通過するフラックス(流束)

=−(ポテンシャル勾配)



$$\sum_k \frac{S_{ik}}{d_{ik} + d_{ki}} \left(\phi_k - \phi_i\right) + V_i \dot{Q}_i = 0$$

一次元差分法との比較(2/3)



厚さ:1

一辺の長さΔhの正方形メッシュ

接触面積: $S_{ik} = \Delta h$

要素体積: $V_i = \Delta h^2$

接触面までの距離: $d_{ij} = \Delta h/2$

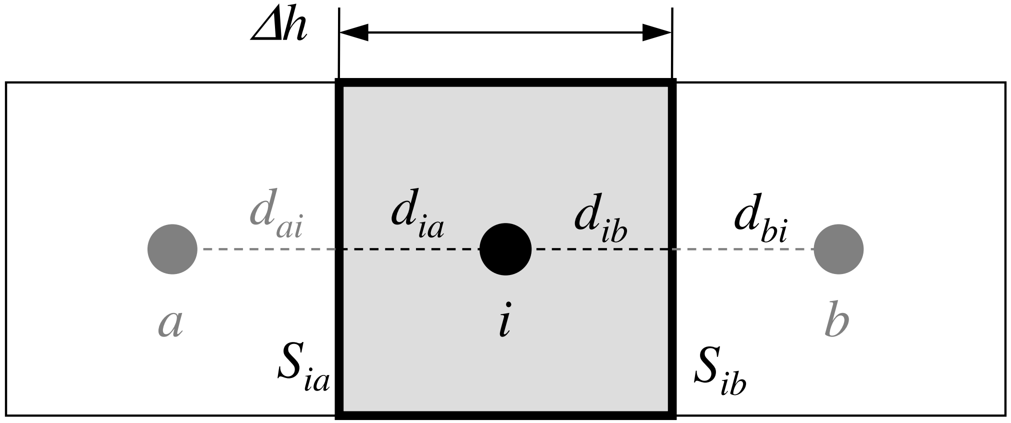
$$\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\left(\phi_k - \phi_i\right) + V_i \dot{Q}_i = 0$$

両辺をV_iで割る:

$$\frac{1}{V_i}\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\left(\phi_k - \phi_i\right) + \dot{Q}_i = 0$$

この部分に注目すると

一次元差分法との比較(3/3)



一辺の長さΔhの正方形メッシュ

接触面積: $S_{ik}= \Delta h$

要素体積: $V_i = \Delta h^2$

接触面までの距離: $d_{ij}=\Delta h/2$

厚さ:1

$$\frac{1}{V_i}\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}(\phi_k-\phi_i)=\frac{1}{(\Delta h)^2}\sum_{k=a,b}\frac{\Delta h}{\dfrac{\Delta h}{2}+\dfrac{\Delta h}{2}}(\phi_k-\phi_i)$$

$$=\frac{1}{(\Delta h)^2}\sum_{k=a,b}\frac{\Delta h}{\dfrac{\Delta h}{2}+\dfrac{\Delta h}{2}}(\phi_k-\phi_i)=\frac{1}{(\Delta h)^2}\sum_{k=a,b}\frac{\Delta h}{\Delta h}(\phi_k-\phi_i)=\frac{1}{(\Delta h)^2}\sum_{k=a,b}(\phi_k-\phi_i)$$

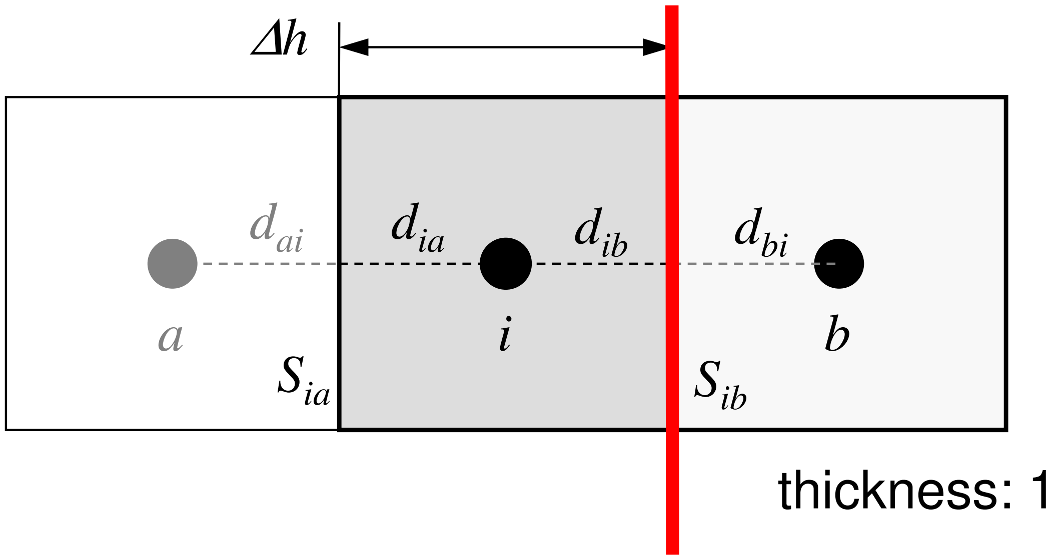
$$=\frac{1}{(\Delta h)^2}(\phi_a-\phi_i)+\frac{1}{(\Delta h)^2}(\phi_b-\phi_i)=\boxed{\frac{\phi_a-2\phi_i+\phi_b}{(\Delta h)^2}}$$

要素iについて成立
連立一次方程式

Heat Equation (1/3)

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

λ: Thermal Conductivity



Δh x Δh Square Mesh

Surface Area: $S_{ik} = \Delta h$

Volume: $V_i = \Delta h^2$

Distance (Ctr.-Suf): $d_{ij} = \Delta h/2$

Heat Flux through this surface: Qs_{ib}

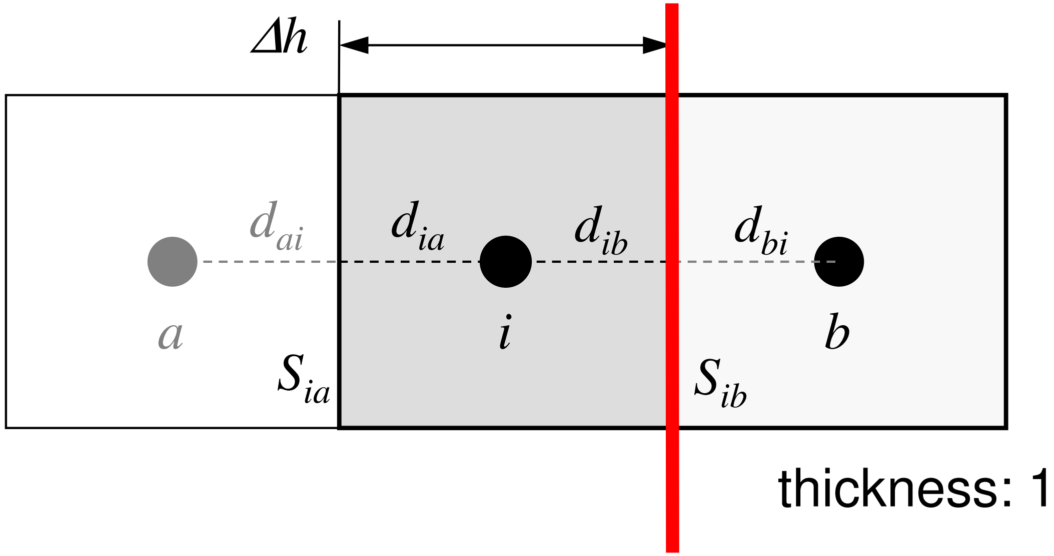
$$Qs_{ib} = -\lambda\frac{T_b - T_i}{\Delta h}\cdot S_{ib}$$

$$\lambda_i = \lambda_b = \lambda$$

Heat Equation (2/3)

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

λ: Thermal Conductivity



Δh x Δh Square Mesh

Surface Area: $S_{ik}=\Delta h$

Volume: $V_i=\Delta h^2$

Distance (Ctr.-Suf): $d_{ij}=\Delta h/2$

Heat Flux through this surface: Qs_{ib}

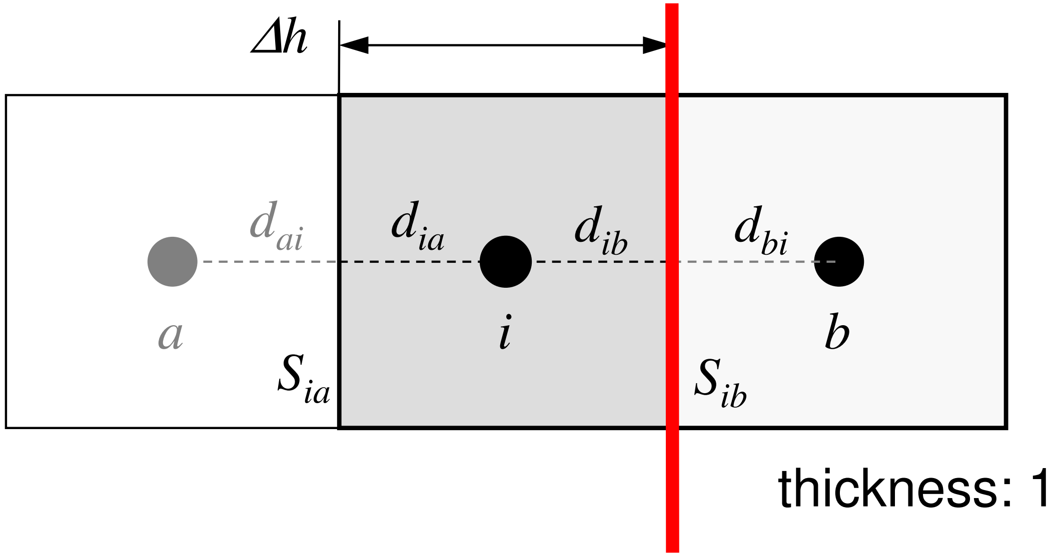
$$Qs_{ib}=-\lambda\frac{T_b-T_i}{\frac{\Delta h}{2}+\frac{\Delta h}{2}}\cdot S_{ib}=-\frac{T_b-T_i}{\left[\left(\frac{\Delta h}{2}\right)/\lambda\right]+\left[\left(\frac{\Delta h}{2}\right)/\lambda\right]}\cdot S_{ib}$$

$$\lambda_i=\lambda_b=\lambda$$

Heat Equation (3/3)

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

λ: Thermal Conductivity



Δh x Δh Square Mesh

Surface Area: $S_{ik}= \Delta h$

Volume: $V_i = \Delta h^2$

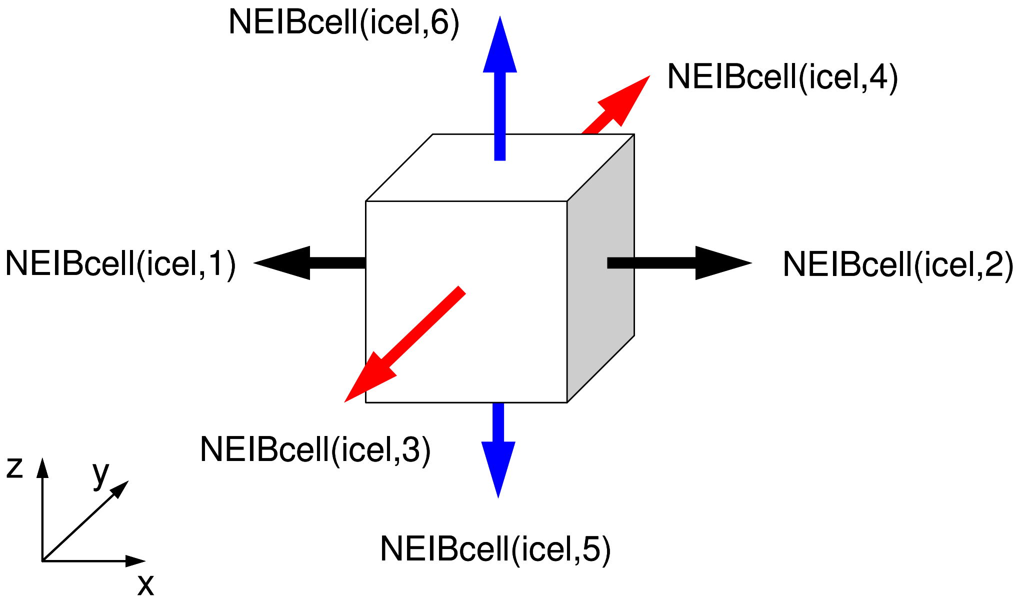
Distance (Ctr.-Suf): $d_{ij}=\Delta h/2$

Heat Flux through this surface: Qs_{ib}

$$Qs_{ib}=-\frac{T_b-T_i}{\left[\left(\frac{\Delta h}{2}\right)/\lambda_i\right]+\left[\left(\frac{\Delta h}{2}\right)/\lambda_b\right]}\cdot S_{ib}$$

$$\lambda_i \neq \lambda_b$$

三次元では・・・



$$\frac{\phi_{neib(icel,1)} - \phi_{icel}}{\Delta x} \Delta y \Delta z + \frac{\phi_{neib(icel,2)} - \phi_{icel}}{\Delta x} \Delta y \Delta z +$$

$$\frac{\phi_{neib(icel,3)} - \phi_{icel}}{\Delta y} \Delta z \Delta x + \frac{\phi_{neib(icel,4)} - \phi_{icel}}{\Delta y} \Delta z \Delta x +$$

$$\frac{\phi_{neib(icel,5)} - \phi_{icel}}{\Delta z} \Delta x \Delta y + \frac{\phi_{neib(icel,6)} - \phi_{icel}}{\Delta z} \Delta x \Delta y + f_{icel} \Delta x \Delta y \Delta z = 0$$

整理すると:連立一次方程式

$$\frac{\phi_{neib(icel,1)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+\frac{\phi_{neib(icel,2)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+$$

$$\frac{\phi_{neib(icel,3)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+\frac{\phi_{neib(icel,4)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+$$

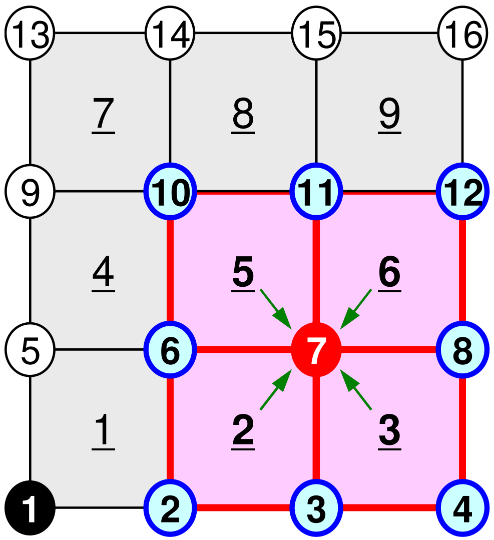
$$\frac{\phi_{neib(icel,5)}-\phi_{icel}}{\Delta z}\Delta x\Delta y+\frac{\phi_{neib(icel,6)}-\phi_{icel}}{\Delta z}\Delta x\Delta y+f_{icel}\Delta x\Delta y\Delta z=0$$

$$\sum_k \frac{S_{icel-k}}{d_{icel-k}}\left(\phi_k-\phi_{icel}\right)=-f_{icel}V_i$$

$$-\underbrace{\left[\sum_k \frac{S_{icel-k}}{d_{icel-k}}\right]}_{\text{対角項}}\phi_{icel}+\underbrace{\left[\sum_k \frac{S_{icel-k}}{d_{icel-k}}\phi_k\right]}_{\text{非対角項}}=-f_{icel}V_i \quad (icel=1,N)$$

$$\longrightarrow \boxed{[A]\{\phi\}=\{f\}}$$

有限要素法の係数マトリクス ゼロが多い:疎行列



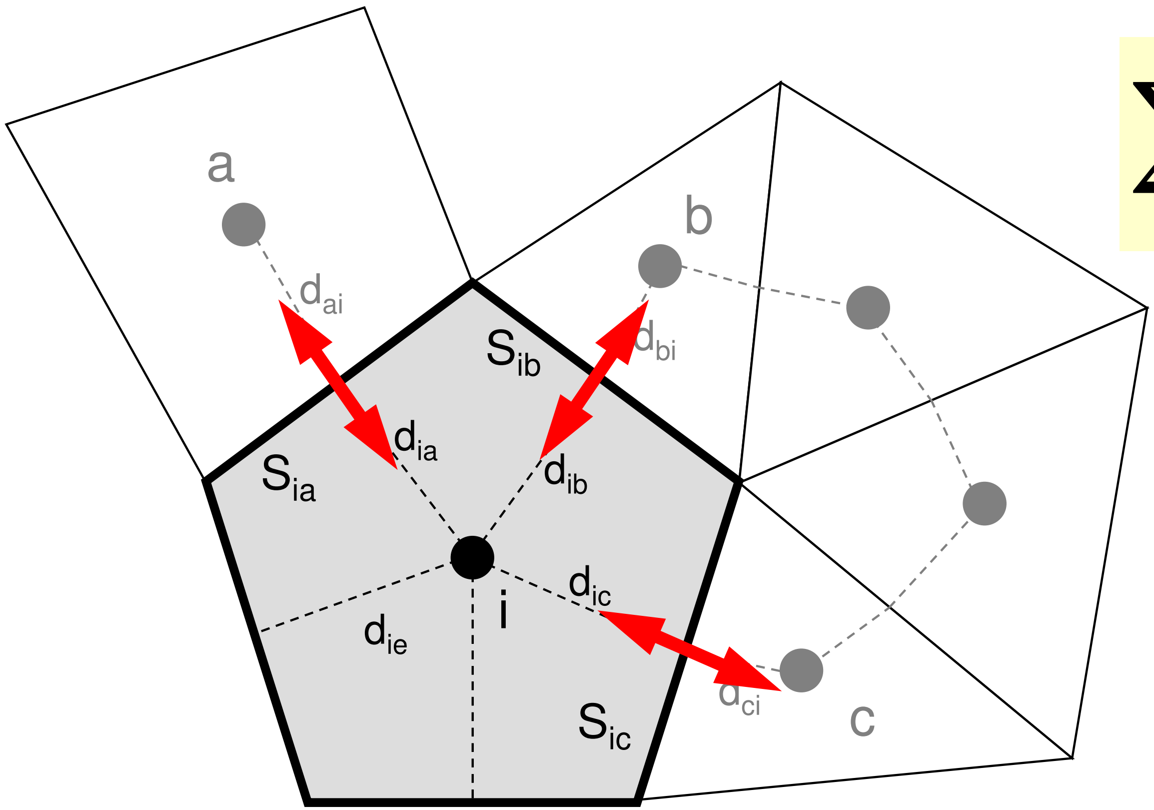
$$[K]\{\Phi\}=\{F\}$$

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

FVMの係数行列も疎行列

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}+\frac{\partial^2\phi}{\partial z^2}+f=0$$

面を通過するフラックスの保存に着目
周囲の要素とのみ関係がある



隣接要素との拡散

$$\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\left(\phi_k-\phi_i\right)+V_i\dot{Q}_i=0$$

体積
フラックス

V_i :要素体積
S :表面面積
d_{ij} :要素中心から表面までの距離
Q :体積フラックス

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

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

- 有限体積法：非零非対角成分の数は高々「数百」規模（この講義ではわずか6）
  - 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：$O(10^{16})$ words $\Rightarrow$ $O(10^{17})$ Bytes = 100PB（OBCXの400倍）
    - 非零非対角成分数：$O(10^{10})$ words $\Rightarrow$ 100 GB（OBCXは192 GB/node）
- **非零成分のみ記憶するのが効率的**

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
```

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

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)

$$
\begin{array}{c|cccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
1 & 1.1 & 2.4 & 0 & 0 & 3.2 & 0 & 0 & 0 \\
2 & 4.3 & 3.6 & 0 & 2.5 & 0 & 3.7 & 0 & 9.1 \\
3 & 0 & 0 & 5.7 & 0 & 1.5 & 0 & 3.1 & 0 \\
4 & 0 & 4.1 & 0 & 9.8 & 2.5 & 2.7 & 0 & 0 \\
5 & 3.1 & 9.5 & 10.4 & 0 & 11.5 & 0 & 4.3 & 0 \\
6 & 0 & 0 & 6.5 & 0 & 0 & 12.4 & 9.5 & 0 \\
7 & 0 & 6.4 & 2.5 & 0 & 0 & 1.4 & 23.1 & 13.1 \\
8 & 0 & 9.5 & 1.3 & 9.6 & 0 & 3.1 & 0 & 51.3
\end{array}
$$

Compressed Row Storage (CRS): C
Numbering starts from 0 in program

	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)

		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)

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

```
Index[0]=  0
Index[1]=  2
Index[2]=  6
Index[3]=  8
Index[4]= 11
Index[5]= 15
Index[6]= 17
Index[7]= 21
Index[8]= 25
```

```
NPLU= 25
(=Index[N])
```

`(Index[i])`th~`(Index[i+1])` th:
Non-Zero Off-Diag. Components corresponding to *i*-th row.

Compressed Row Storage (CRS)

						# Non-Zero Off-Diag.	
							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])th~(Index[i+1]) th:
Non-Zero Off-Diag. Components corresponding to *i*-th row.

Compressed Row Storage (CRS)

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

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

Compressed Row Storage (CRS)

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

```
{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]];
      }
}
```

- 背景
  - 有限体積法
  - 前処理付反復法
- ICCG法によるポアソン方程式法ソルバーについて
  - 実行方法
    - データ構造
  - プログラムの説明
    - 初期化
    - 係数マトリクス生成
    - ICCG法

科学技術計算における 大規模線形方程式の解法

- 多くの科学技術計算は，最終的に大規模線形方程式Ax=bを解くことに帰着される。
  - important, expensive
- アプリケーションに応じて様々な手法が提案されている
  - 疎行列（sparse），密行列（dense）
  - 直接法（direct），反復法（iterative）
- 密行列（dense）
  - グローバルな相互作用：BEM，スペクトル法，MO，MD（気液）
- 疎行列（sparse）
  - ローカルな相互作用：FEM，FDM，MD（固），高速多重極展開付BEM

直接法（Direct Method）

- Gaussの消去法，完全LU分解
  - 逆行列A^{-1}を直接求める（または同等の計算をする）

- 利点
  - 安定，幅広いアプリケーションに適用可能
    - Partial Pivoting
  - 疎行列，密行列いずれにも適用可能
- 欠点
  - 反復法よりもメモリ，計算時間を必要とする
    - 密行列の場合，$O(N^3)$の計算量
  - 大規模な計算向けではない
    - $O(N^2)$の記憶容量，$O(N^3)$の計算量

反復法とは・・・

Linear Equations
連立一次方程式

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

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

Initial Solution
初期解

$$\mathbf{x}^{(0)} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ \vdots \\ x_n^{(0)} \end{pmatrix}$$

適当な初期解 $\mathrm{x}^{(0)}$から始めて，繰り返し計算によって真の解に収束(converge)させていく

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots$$

反復法（Iterative Method）

- 定常（stationary）法
  - 反復計算中，解ベクトル以外の変数は変化せず
  - SOR，Gauss-Seidel，Jacobiなど
  - 概して遅い

$$\mathbf{Ax} = \mathbf{b} \Rightarrow$$
$$\mathbf{x}^{(k+1)} = \mathbf{Mx}^{(k)} + \mathbf{Nb}$$

- 非定常（nonstationary）法
  - 拘束，最適化条件が加わる
  - Krylov部分空間（subspace）への写像を基底として使用するため，Krylov部分空間法とも呼ばれる
  - CG（Conjugate Gradient：共役勾配法）
  - BiCGSTAB（Bi-Conjugate Gradient Stabilized）
  - GMRES（Generalized Minimal Residual）

反復法(Iterative Method)(続き)

- 利点
  - 直接法と比較して，メモリ使用量，計算量が少ない。
  - 並列計算には適している。

- 欠点
  - 収束性が，アプリケーション，境界条件の影響を受けやすい。
  - 前処理(preconditioning)が重要。

非定常反復法:クリロフ部分空間法(1/2) Krylov Subspace Method

$$\mathbf{Ax} = \mathbf{b} \Rightarrow \mathbf{x} = \mathbf{b} + (\mathbf{I} - \mathbf{A})\mathbf{x}$$

以下の反復式を導入しx_0, x_1, x_2, ..., x_kを求める:

$$\begin{aligned}\mathbf{x}_k &= \mathbf{b} + (\mathbf{I} - \mathbf{A})\mathbf{x}_{k-1} \\ &= (\mathbf{b} - \mathbf{Ax}_{k-1}) + \mathbf{x}_{k-1} \\ &= \mathbf{r}_{k-1} + \mathbf{x}_{k-1}\end{aligned}$$

$where\ \mathbf{r}_k = \mathbf{b} - \mathbf{Ax}_k$:残差ベクトル(residual)

$$\mathbf{x}_k = \mathbf{x}_0 + \sum_{i=0}^{k-1} \mathbf{r}_i$$

$$\begin{aligned}\mathbf{r}_k &= \mathbf{b} - \mathbf{Ax}_k = \mathbf{b} - \mathbf{A}(\mathbf{r}_{k-1} + \mathbf{x}_{k-1}) \\ &= (\mathbf{b} - \mathbf{Ax}_{k-1}) - \mathbf{Ar}_{k-1} = \mathbf{r}_{k-1} - \mathbf{Ar}_{k-1} = (\mathbf{I} - \mathbf{A})\mathbf{r}_{k-1}\end{aligned}$$

非定常反復法：クリロフ部分空間法（2/2）
Krylov Subspace Method

$$\mathbf{x}_k = \mathbf{x}_0 + \sum_{i=0}^{k-1} \mathbf{r}_i = \mathbf{x}_0 + \mathbf{r}_0 + \sum_{i=0}^{k-2} (\mathbf{I}-\mathbf{A})\mathbf{r}_i = \mathbf{x}_0 + \mathbf{r}_0 + \sum_{i=1}^{k-1} (\mathbf{I}-\mathbf{A})^i \mathbf{r}_0$$

$$\mathbf{z}_k = \mathbf{r}_0 + \sum_{i=1}^{k-1} (\mathbf{I}-\mathbf{A})^i \mathbf{r}_0 = \left[\mathbf{I} + \sum_{i=1}^{k-1} (\mathbf{I}-\mathbf{A})^i \right] \mathbf{r}_0$$

$\mathbf{z}_\mathbf{k}$はk次のクリロフ部分空間（Krylov Subspace）に属するベクトル，問題はクリロフ部分空間からどのようにして解の近似ベクトル$\mathbf{x}_\mathbf{k}$を求めるかにある：

$$\left[\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0 \right]$$

代表的な非定常反復法：共役勾配法

- Conjugate Gradient法，略して「CG」法
  - 最も代表的な「非定常」反復法
- 対称正定値行列（Symmetric Positive Definite：SPD）
  - 任意のベクトル$\{x\}$に対して$\{x\}^T[A]\{x\}>0$
  - 全対角成分>0，全固有値>0，全部分行列式（主小行列式・首座行列式）>0と同値
- アルゴリズム
  - 最急降下法（Steepest Descent Method）の変種
  - $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
    - $x^{(i)}$：反復解，$p^{(i)}$：探索方向，α_i：定数）
  - 厳密解をyとするとき$\{x-y\}^T[A]\{x-y\}$を最小とするような$\{x\}$を求める。
  - 詳細は参考文献参照
    - 例えば：森正武「数値解析（第2版）」（共立出版）

$$
\det\begin{bmatrix}
a_{11} & a_{12} & a_{13} & a_{14} & \cdots & a_{1n} \\
a_{21} & a_{22} & a_{23} & a_{24} & \cdots & a_{2n} \\
a_{31} & a_{32} & a_{33} & a_{34} & \cdots & a_{3n} \\
a_{41} & a_{42} & a_{43} & a_{44} & \cdots & a_{4n} \\
\vdots & \vdots & \vdots & \vdots & & \vdots \\
a_{n1} & a_{n2} & a_{n3} & a_{n4} & \cdots & a_{nn}
\end{bmatrix}
$$

共役勾配法（CG法）のアルゴリズム

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

- 行列ベクトル積
- ベクトル内積
- ベクトル定数倍の加減（DAXPY）

$x^{(i)}$: Vector

α_i : Scalar

共役勾配法（CG法）のアルゴリズム

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

- 行列ベクトル積
- ベクトル内積
- ベクトル定数倍の加減（DAXPY）

$x^{(i)}$: Vector
α_i : Scalar

共役勾配法（CG法）のアルゴリズム

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

- 行列ベクトル積
- ベクトル内積
- ベクトル定数倍の加減（DAXPY）

$x^{(i)}$: Vector

α_i : Scalar

共役勾配法（CG法）のアルゴリズム

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

- 行列ベクトル積
- ベクトル内積
- ベクトル定数倍の加減（DAXPY）
  - Double
  - {y}= a{x} + {y}

$x^{(i)}$: Vector

α_i : Scalar

共役勾配法(CG法)のアルゴリズム

Compute $r^{(0)} = b - [A]x^{(0)}$
for $i = 1, 2, \ldots$
    $\rho_{i-1} = r^{(i-1)}\, r^{(i-1)}$
    if $i=1$
        $p^{(1)} = r^{(0)}$
    else
        $\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$
        $p^{(i)} = r^{(i-1)} + \beta_{i-1}\, p^{(i-1)}$
    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

$x^{(i)}$: Vector
α_i : Scalar

CG法アルゴリズムの導出(1/5)

yを厳密解（ $Ay=b$ ）とするとき，下式を最小にするxを求める：

$$(x-y)^T[A](x-y)$$

$$(x-y)^T[A](x-y)=(x,Ax)-(y,Ax)-(x,Ay)+(y,Ay)$$
$$=(x,Ax)-2(x,Ay)+(y,Ay)=(x,Ax)-2(x,b)+\underline{(y,b)}$$

定数

従って，下記 $f(x)$ を最小にするxを求めればよい：

$$f(x)=\frac{1}{2}(x,Ax)-(x,b)$$

$$f(x+h)=f(x)+(h,Ax-b)+\frac{1}{2}(h,Ah)$$

任意のベクトル h

$$f(x) = \frac{1}{2}(x, Ax) - (x, b)$$

$$f(x+h) = f(x) + (h, Ax-b) + \frac{1}{2}(h, Ah)$$

• 任意のベクトルh

$$f(x+h) = \frac{1}{2}(x+h, A(x+h)) - (x+h, b)$$

$$= \frac{1}{2}(x+h, Ax) + \frac{1}{2}(x+h, Ah) - (x, b) - (h, b)$$

$$= \frac{1}{2}(x, Ax) + \underline{\frac{1}{2}(h, Ax)} + \underline{\frac{1}{2}(x, Ah)} + \frac{1}{2}(h, Ah) - (x, b) - (h, b)$$

$$= \frac{1}{2}(x, Ax) - (x, b) + (h, Ax) - (h, b) + \frac{1}{2}(h, Ah)$$

$$= f(x) + (h, Ax-b) + \frac{1}{2}(h, Ah)$$

CG法アルゴリズムの導出(2/5)

CG法は任意の $x^{(0)}$ から始めて, $f(x)$の最小値を逐次探索する。
今, k番目の近似値 $x^{(k)}$と探索方向$p^{(k)}$が決まったとすると:

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$$

$f(x^{(k+1)})$ を最小にするためには:

$$f\left(x^{(k)} + \alpha_k p^{(k)}\right) = \frac{1}{2}\alpha_k{}^2\left(p^{(k)}, Ap^{(k)}\right) - \alpha_k\left(p^{(k)}, b - Ax^{(k)}\right) + f\left(x^{(k)}\right)$$

$$\frac{\partial f\left(x^{(k)} + \alpha_k p^{(k)}\right)}{\partial \alpha_k} = 0 \Rightarrow \alpha_k = \frac{\left(p^{(k)}, b - Ax^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)} = \frac{\left(p^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)} \quad \textbf{(1)}$$

$r^{(k)} = b - Ax^{(k)}$ は第k近似に対する残差

CG法アルゴリズムの導出(3/5)

残差 $r^{(k)}$も以下の式によって計算できる:

$$r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)}$$ **(2)**

$$r^{(k+1)} = b - Ax^{(k+1)},\ r^{(k)} = b - Ax^{(k)}$$

$$r^{(k+1)} - r^{(k)} = -Ax^{(k+1)} + Ax^{(k)} = -\alpha_k A p^{(k)}$$

探索方向を以下の漸化式によって求める:

$$p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)},\ r^{(0)} = p^{(0)}$$ **(3)**

本当のところは下記のように(k+1)回目に厳密解 y が求まれば良いのであるが，解がわかっていない場合は困難・・・

$$y = x^{(k+1)} + \alpha_{k+1} p^{(k+1)}$$

CG法アルゴリズムの導出(4/5)

ところで，下式のような都合の良い直交関係がある:

$$\left(Ap^{(k)}, y-x^{(k+1)}\right)=0$$

$$\left(Ap^{(k)}, y-x^{(k+1)}\right)=\left(p^{(k)}, Ay-Ax^{(k+1)}\right)=\left(p^{(k)}, b-Ax^{(k+1)}\right)$$
$$=\left(p^{(k)}, b-A\left[x^{(k)}+\alpha_k p^{(k)}\right]\right)=\left(p^{(k)}, b-Ax^{(k)}-\alpha_k Ap^{(k)}\right)$$
$$=\left(p^{(k)}, r^{(k)}-\alpha_k Ap^{(k)}\right)=\left(p^{(k)}, r^{(k)}\right)-\alpha_k\left(p^{(k)}, Ap^{(k)}\right)=0$$

$$\because \alpha_k=\frac{\left(p^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$$

従って以下が成立する:

$$\left(Ap^{(k)}, y-x^{(k+1)}\right)=\left(Ap^{(k)}, \alpha_{k+1} p^{(k+1)}\right)=0 \Rightarrow \left(p^{(k+1)}, Ap^{(k)}\right)=0$$

CG法アルゴリズムの導出(5/5)

$$\left(p^{(k+1)}, Ap^{(k)}\right) = \left(r^{(k+1)} + \beta_k p^{(k)}, Ap^{(k)}\right) = \left(r^{(k+1)}, Ap^{(k)}\right) + \beta_k \left(p^{(k)}, Ap^{(k)}\right) = 0$$

$$\Rightarrow \beta_k = \frac{-\left(r^{(k+1)}, Ap^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)} \quad \textbf{(4)}$$

$$\left(p^{(k+1)}, Ap^{(k)}\right) = 0$$

$p^{(k)}$ と $p^{(k+1)}$ が行列Aに関して共役（**conjugate**）

$p^{(k)}$:探索方向ベクトル，勾配（gradient）ベクトル

```
Compute p(0)=r(0)= b-[A]x(0)
for i= 1, 2, …
    calc. αi-1
    x(i)= x(i-1) + αi-1p(i-1)
    r(i)= r(i-1) - αi-1[A]p(i-1)

    check convergence |r|
    (if not converged)
    calc. βi-1
    p(i)= r(i) + βi-1 p(i-1)
end
```

$$\alpha_{i-1} = \frac{\left(p^{(i-1)}, r^{(i-1)}\right)}{\left(p^{(i-1)}, Ap^{(i-1)}\right)}$$

$$\beta_{i-1} = \frac{-\left(r^{(i)}, Ap^{(i-1)}\right)}{\left(p^{(i-1)}, Ap^{(i-1)}\right)}$$

CG法アルゴリズム

任意の(i,j)に対して以下の共役関係が得られる:

$$\left(p^{(i)}, Ap^{(j)}\right) = 0 \ (i \neq j)$$

探索方向p(k)，残差ベクトルr(k)についても以下の関係が成立する:

$$\left(r^{(i)}, r^{(j)}\right) = 0 \ (i \neq j), \quad \left(p^{(k)}, r^{(k)}\right) = \left(r^{(k)}, r^{(k)}\right)$$

N次元空間で互いに直交で一次独立な残差ベクトル $r^{(k)}$ はN個しか存在しない，従って共役勾配法は未知数がN個のときにN回以内に収束する ⇒ 実際は丸め誤差の影響がある（条件数が大きい場合）

Top 10 Algorithms in the 20th Century (SIAM)

http://www.siam.org/news/news.php?id=637

モンテカルロ法，シンプレックス法，クリロフ部分空間法，行列分解法，
最適化Fortranコンパイラ，QR法，クイックソート，FFT，
整数関係アルゴリズム，FMM（高速多重極法）

Proof (1/3)
Mathematical Induction
数学的帰納法

$$\left(r^{(i)}, r^{(j)}\right) = 0 \; (i \neq j)$$ 直交性

$$\left(p^{(i)}, Ap^{(j)}\right) = 0 \; (i \neq j)$$ 共役性

(1) $$\alpha_k = \frac{\left(p^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$$

(2) $$r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)}$$

(3) $$p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)}, \; r^{(0)} = p^{(0)}$$

(4) $$\beta_k = \frac{-\left(r^{(k+1)}, Ap^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$$

Proof (2/3)

Mathematical Induction

数学的帰納法

$$(r^{(i)}, r^{(j)}) = 0 \ (i \neq j)$$
$$(p^{(i)}, Ap^{(j)}) = 0 \ (i \neq j)$$
(*)

(*) **is satisfied for** $i \leq k,\ j \leq k$ where $i \neq j$

if $i < k$

$$(r^{(k+1)}, r^{(i)}) = (r^{(i)}, r^{(k+1)}) \overset{(2)}{=} (r^{(i)}, r^{(k)} - \alpha_k Ap^{(k)})$$
$$\overset{(*)}{=} -\alpha_k (r^{(i)}, Ap^{(k)}) \overset{(3)}{=} -\alpha_k (p^{(i)} - \beta_{i-1} p^{(i-1)}, Ap^{(k)})$$
$$= -\alpha_k (p^{(i)}, Ap^{(k)}) + \alpha_k \beta_{i-1} (p^{(i-1)}, Ap^{(k)}) \overset{(*)}{=} 0$$

if $i = k$

$$(r^{(k+1)}, r^{(k)}) \overset{(2)}{=} (r^{(k)}, r^{(k)}) - (r^{(k)}, \alpha_k Ap^{(k)})$$
$$\overset{(3)}{=} (r^{(k)}, r^{(k)}) - (p^{(k)} - \beta_{k-1} p^{(k-1)}, \alpha_k Ap^{(k)})$$
$$\overset{(*)}{=} (r^{(k)}, r^{(k)}) - \alpha_k (p^{(k)}, Ap^{(k)}) \overset{(1)}{=} (r^{(k)}, r^{(k)}) - (p^{(k)}, r^{(k)})$$
$$\overset{(3)}{=} (r^{(k)}, r^{(k)}) - (\beta_{k-1} p^{(k-1)} + r^{(k)}, r^{(k)})$$
$$= -\beta_{k-1} (p^{(k-1)}, r^{(k)}) \overset{(2)}{=} -\beta_{k-1} (p^{(k-1)}, r^{(k-1)} - \alpha_{k-1} Ap^{(k-1)})$$
$$= -\beta_{k-1} \{ (p^{(k-1)}, r^{(k-1)}) - \alpha_{k-1} (p^{(k-1)}, Ap^{(k-1)}) \} \overset{(1)}{=} 0$$

(1) $\alpha_k = \dfrac{(p^{(k)}, r^{(k)})}{(p^{(k)}, Ap^{(k)})}$

(2) $r^{(k+1)} = r^{(k)} - \alpha_k Ap^{(k)}$

(3) $p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)}$

(4) $\beta_k = \dfrac{-(r^{(k+1)}, Ap^{(k)})}{(p^{(k)}, Ap^{(k)})}$

Proof (3/3)

Mathematical Induction
数学的帰納法

$$\left(r^{(i)}, r^{(j)}\right) = 0 \ (i \neq j)$$
$$\left(p^{(i)}, Ap^{(j)}\right) = 0 \ (i \neq j) \quad (*)$$

(*) is satisfied for $i \le k,\ j \le k$ where $i \neq j$

if $i < k$

$$\left(p^{(k+1)}, Ap^{(i)}\right) \overset{(3)}{=} \left(r^{(k+1)} + \beta_k p^{(k)}, Ap^{(i)}\right)$$
$$\overset{(*)}{=} \left(r^{(k+1)}, Ap^{(i)}\right)$$
$$\overset{(2)}{=} \frac{1}{\alpha_i}\left(r^{(k+1)}, r^{(i)} - r^{(i+1)}\right) = 0$$

if $i = k$

$$\left(p^{(k+1)}, Ap^{(k)}\right) \overset{(3)}{=} \left(r^{(k+1)}, Ap^{(k)}\right) + \beta_k \left(p^{(k)}, Ap^{(k)}\right)$$
$$\overset{(4)}{=} 0$$

(1) $\alpha_k = \dfrac{\left(p^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$

(2) $r^{(k+1)} = r^{(k)} - \alpha_k Ap^{(k)}$

(3) $p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)}$

(4) $\beta_k = \dfrac{-\left(r^{(k+1)}, Ap^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$

$$\left(r^{(k+1)}, r^{(k)}\right) = 0$$

$$\left(r^{(k+1)}, r^{(k)}\right) \overset{(2)}{=} \left(r^{(k)}, r^{(k)}\right) - \left(r^{(k)}, \alpha_k A p^{(k)}\right)$$

$$\overset{(3)}{=} \left(r^{(k)}, r^{(k)}\right) - \left(p^{(k)} - \beta_{k-1} p^{(k-1)}, \alpha_k A p^{(k)}\right)$$

$$\overset{(*)}{=} \left(r^{(k)}, r^{(k)}\right) - \alpha_k \left(p^{(k)}, A p^{(k)}\right) \overset{(1)}{=} \left(r^{(k)}, r^{(k)}\right) - \left(p^{(k)}, r^{(k)}\right) = 0$$

$$\therefore \left(r^{(k)}, r^{(k)}\right) = \left(p^{(k)}, r^{(k)}\right)$$

(1) $\alpha_k = \dfrac{\left(p^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, A p^{(k)}\right)}$

(2) $r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)}$

(3) $p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)}$

(4) $\beta_k = \dfrac{-\left(r^{(k+1)}, A p^{(k)}\right)}{\left(p^{(k)}, A p^{(k)}\right)}$

$\alpha_k,\ \beta_k$

実際は$\alpha_k,\ \beta_k$はもうちょっと簡単な形に変形できる:

$$\alpha_k = \frac{\left(p^{(k)}, b - Ax^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)} = \frac{\left(p^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)} = \frac{\left(r^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$$

$$\because \left(p^{(k)}, r^{(k)}\right) = \left(r^{(k)}, r^{(k)}\right)$$

$$\beta_k = \frac{-\left(r^{(k+1)}, Ap^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)} = \frac{\left(r^{(k+1)}, r^{(k+1)}\right)}{\left(r^{(k)}, r^{(k)}\right)}$$

$$\because \left(r^{(k+1)}, Ap^{(k)}\right) = \frac{\left(r^{(k+1)}, r^{(k)} - r^{(k+1)}\right)}{\alpha_k} = -\frac{\left(r^{(k+1)}, r^{(k+1)}\right)}{\alpha_k}$$

共役勾配法（CG法）のアルゴリズム

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

$x^{(i)}$: Vector

α_i : Scalar

$$\beta_{i-1} = \frac{\left(r^{(i-1)}, r^{(i-1)}\right)}{\left(r^{(i-2)}, r^{(i-2)}\right)} \qquad \begin{matrix}\left(= \rho_{i-1}\right) \\ \left(= \rho_{i-2}\right)\end{matrix}$$

$$\alpha_i = \frac{\left(r^{(i-1)}, r^{(i-1)}\right)}{\left(p^{(i)}, Ap^{(i)}\right)} \qquad \left(= \rho_{i-1}\right)$$

前処理（preconditioning）とは?

- 反復法の収束は係数行列の固有値分布に依存
  - 固有値分布が少なく，かつ1に近いほど収束が早い（単位行列）
  - 条件数（condition number）（対称正定）＝最大最小固有値比
    - 条件数が1に近いほど収束しやすい
- もとの係数行列 **[A]** に良く似た前処理行列 **[M]** を適用することによって固有値分布を改善する。
  - 前処理行列 **[M]** によって元の方程式$Ax=b$を $A'x=b'$ へと変換する。ここで $A'=M^{-1}A$，$b'=M^{-1}b$である。
  - $A'=M^{-1}A$が単位行列に近ければ良いということになる。
    - 一般には $A'x'=b'$，$A'=M_L^{-1}AM_R^{-1}$，$b'=M_L^{-1}b$，$x'=M_Rx$
    - M_L/M_R : 左/右前処理（Left/Right Preconditioning）
- 「前処理」は密行列，疎行列ともに使用するが，普通は疎行列を対象にすることが多い。

前処理付共役勾配法
Preconditioned Conjugate Gradient Method（PCG）

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

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

$[A'] x' = b'$

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

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

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

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

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

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

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

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

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

CG法では通常，$[\mathrm{M}_2]=[\mathrm{M}_1]^T$ である（例：不完全コレスキー分解）
従って$[\mathrm{M}_1]$ と $[\mathrm{M}_2]$ を以下のように定義する：

$$[\mathrm{M}_1]=[\mathrm{X}]^T,[\mathrm{M}_2]=[\mathrm{X}],[\mathrm{M}]=[\mathrm{M}_1][\mathrm{M}_2]$$

$$[\mathrm{A}']x'=b'$$

$$[\mathrm{A}']=[\mathrm{M}_1]^{-1}[\mathrm{A}][\mathrm{M}_2]^{-1}=[[\mathrm{X}]^T]^{-1}[\mathrm{A}][\mathrm{X}]^{-1}=[\mathrm{X}]^{-T}[\mathrm{A}][\mathrm{X}]^{-1}$$

$$x'=[\mathrm{X}]x,\quad b'=[\mathrm{X}]^{-T}b,\quad r'=[\mathrm{X}]^{-T}r$$

$$\begin{aligned}\alpha'_{i-1}&=\frac{\left(r'^{(i-1)},r'^{(i-1)}\right)}{\left(p'^{(i-1)},A'p'^{(i-1)}\right)}=\frac{\left([\mathrm{X}]^{-T}r^{(i-1)},[\mathrm{X}]^{-T}r^{(i-1)}\right)}{\left([\mathrm{X}]p^{(i-1)},[\mathrm{X}]^{-T}[\mathrm{A}][\mathrm{X}]^{-1}[\mathrm{X}]p^{(i-1)}\right)}\\&=\frac{\left(\left([\mathrm{X}]^{-T}r^{(i-1)}\right)^T,[\mathrm{X}]^{-T}r^{(i-1)}\right)}{\left(\left([\mathrm{X}]p^{(i-1)}\right)^T,[\mathrm{X}]^{-T}[\mathrm{A}]p^{(i-1)}\right)}=\frac{\left(\left(r^{(i-1)}\right)^T[\mathrm{X}]^{-1},\left[\mathrm{X}^T\right]^{-1}r^{(i-1)}\right)}{\left(\left(p^{(i-1)}\right)^T[\mathrm{X}]^T,[\mathrm{X}]^{-T}[\mathrm{A}]p^{(i-1)}\right)}\\&=\frac{\left(r^{(i-1)},\left[\left[\mathrm{X}^T\right][\mathrm{X}]\right]^{-1}r^{(i-1)}\right)}{\left(p^{(i-1)},[\mathrm{A}]p^{(i-1)}\right)}=\frac{\left(r^{(i-1)},[\mathrm{M}]^{-1}r^{(i-1)}\right)}{\left(p^{(i-1)},[\mathrm{A}]p^{(i-1)}\right)}=\frac{\left(r^{(i-1)},z^{(i-1)}\right)}{\left(p^{(i-1)},[\mathrm{A}]p^{(i-1)}\right)}\end{aligned}$$

$$\beta'_{i-1} = \frac{\left(r'^{(i-1)}, r'^{(i-1)}\right)}{\left(r'^{(i-2)}, r'^{(i-2)}\right)} == \frac{\left([\mathrm{X}]^{-T} r^{(i-1)}, [\mathrm{X}]^{-T} r^{(i-1)}\right)}{\left([\mathrm{X}]^{-T} r^{(i-2)}, [\mathrm{X}]^{-T} r^{(i-2)}\right)}$$

$$= \frac{\left(\left([\mathrm{X}]^{-T} r^{(i-1)}\right)^T, [\mathrm{X}]^{-T} r^{(i-1)}\right)}{\left(\left([\mathrm{X}]^{-T} r^{(i-2)}\right)^T, [\mathrm{X}]^{-T} r^{(i-2)}\right)} = \frac{\left(\left(r^{(i-1)}\right)^T [\mathrm{X}]^{-1}, \left[\mathrm{X}^T\right]^{-1} r^{(i-1)}\right)}{\left(\left(r^{(i-2)}\right)^T [\mathrm{X}]^{-1}, \left[\mathrm{X}^T\right]^{-1} r^{(i-2)}\right)}$$

$$= \frac{\left(r^{(i-1)}, \left[\left[\mathrm{X}^T\right][\mathrm{X}]\right]^{-1} r^{(i-1)}\right)}{\left(r^{(i-2)}, \left[\left[\mathrm{X}^T\right][\mathrm{X}]\right]^{-1} r^{(i-2)}\right)} = \frac{\left(r^{(i-1)}, [\mathrm{M}]^{-1} r^{(i-1)}\right)}{\left(r^{(i-2)}, [\mathrm{M}]^{-1} r^{(i-2)}\right)} = \frac{\left(r^{(i-1)}, z^{(i-1)}\right)}{\left(r^{(i-2)}, z^{(i-2)}\right)}$$

前処理付共役勾配法
Preconditioned Conjugate Gradient Method (PCG)

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

実際にやるべき計算は:

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

「近似逆行列」の計算が必要:

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

究極の前処理:本当の逆行列

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

対角スケーリング:簡単=弱い

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

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。
- 簡単な問題では収束する。

ILU(0), IC(0)

- 最もよく使用されている前処理（疎行列用）
  - 不完全LU分解
    - Incomplete LU Factorization
  - 不完全コレスキー分解
    - Incomplete Cholesky Factorization（対称行列）

- 不完全な直接法
  - もとの行列が疎でも，逆行列は疎とは限らない。
  - fill-in
  - もとの行列と同じ非ゼロパターン（fill-in無し）を持っているのがILU（0），IC（0）

LU分解法：完全LU分解法

- 直接法の一種
  - 逆行列を直接求める手法
  - 「逆行列」に相当するものを保存しておけるので，右辺が変わったときに計算時間を節約できる
  - 逆行列を求める際にFill-in（もとの行列では0であったところに値が入る）が生じる
- LU factorization

「不」完全LU分解法

- ILU factorization
  - Incomplete LU factorization
- Fill-inの発生を制限して，前処理に使う手法
  - 不完全な逆行列，少し弱い直接法
  - Fill-inを許さないとき：ILU(0)

LU分解による連立一次方程式の解法

Aがn×n行列のとき、Aを次式のように表すことを（あるいは、そのようなLとUそのものを）AのLU分解という.

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

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

L : Lower triangular part of matrix A

U : Upper triangular part of matrix A

連立一次方程式の行列表現

n元の連立一次方程式の一般形

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

$$\Updownarrow$$

行列表現

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

LU分解を用いたAx=bの解法

1 $\mathbf{A} = \mathbf{LU}$ となるAのLU分解LとUを求める.

2 $\mathbf{Ly} = \mathbf{b}$ の解yを求める.（簡単！）

3 $\mathbf{Ux} = \mathbf{y}$ の解xを求める.（簡単！）

このxが $\mathbf{Ax} = \mathbf{b}$ の解となる

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

Ly=bの解法：前進代入

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

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

芋づる式に（one after another）解が求まる.

Ux=yの解法：後退代入

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

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

芋づる式に（one after another）解が求まる.

LU分解の求め方

①

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

③

② ④

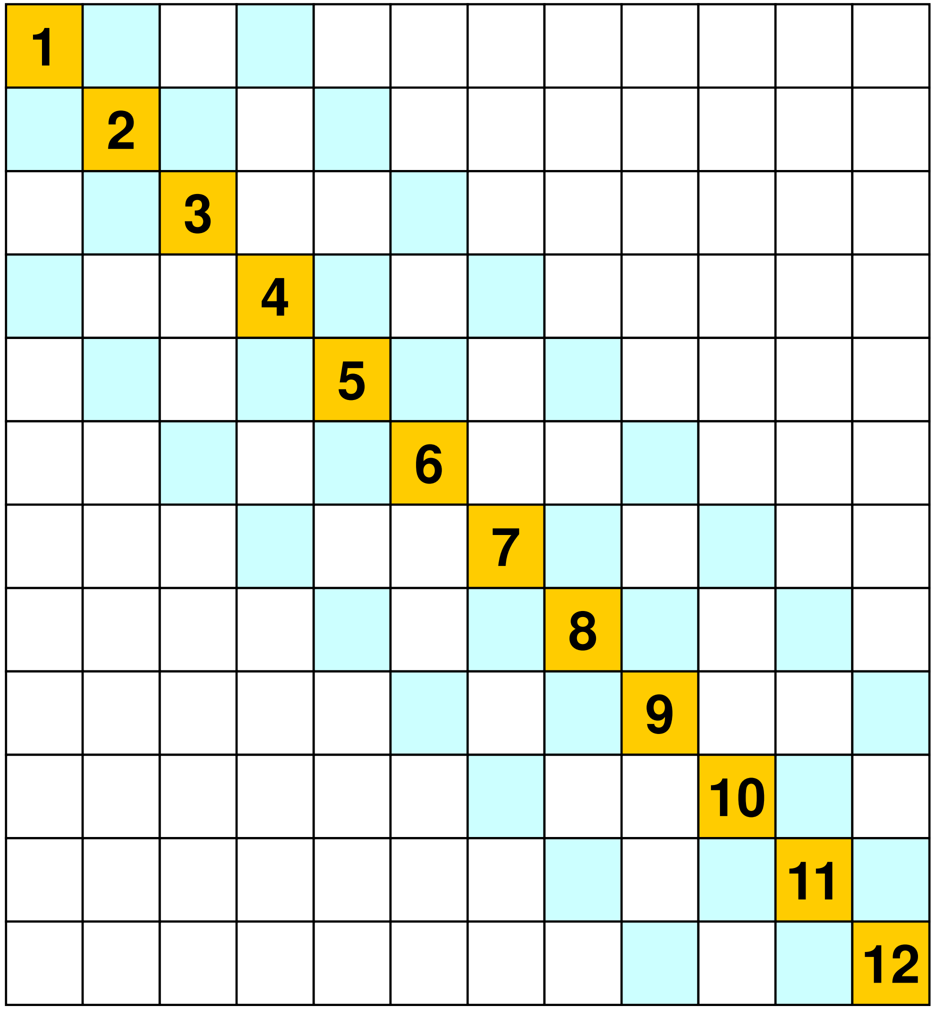
① ➡ $a_{11} = u_{11}, a_{12} = u_{12}, \cdots, a_{1n} = u_{1n} \Rightarrow u_{11}, u_{12}, \cdots, u_{1n}$

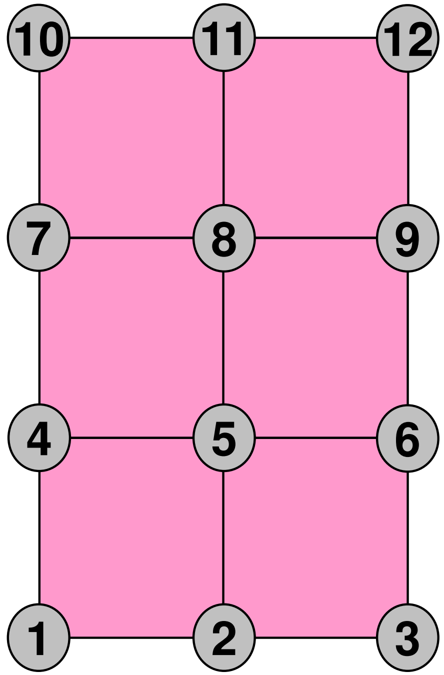
② ➡ $a_{21} = l_{21}u_{11}, a_{31} = l_{31}u_{11}, \cdots, a_{n1} = l_{n1}u_{11} \Rightarrow l_{21}, l_{31}, \cdots, l_{n1}$

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

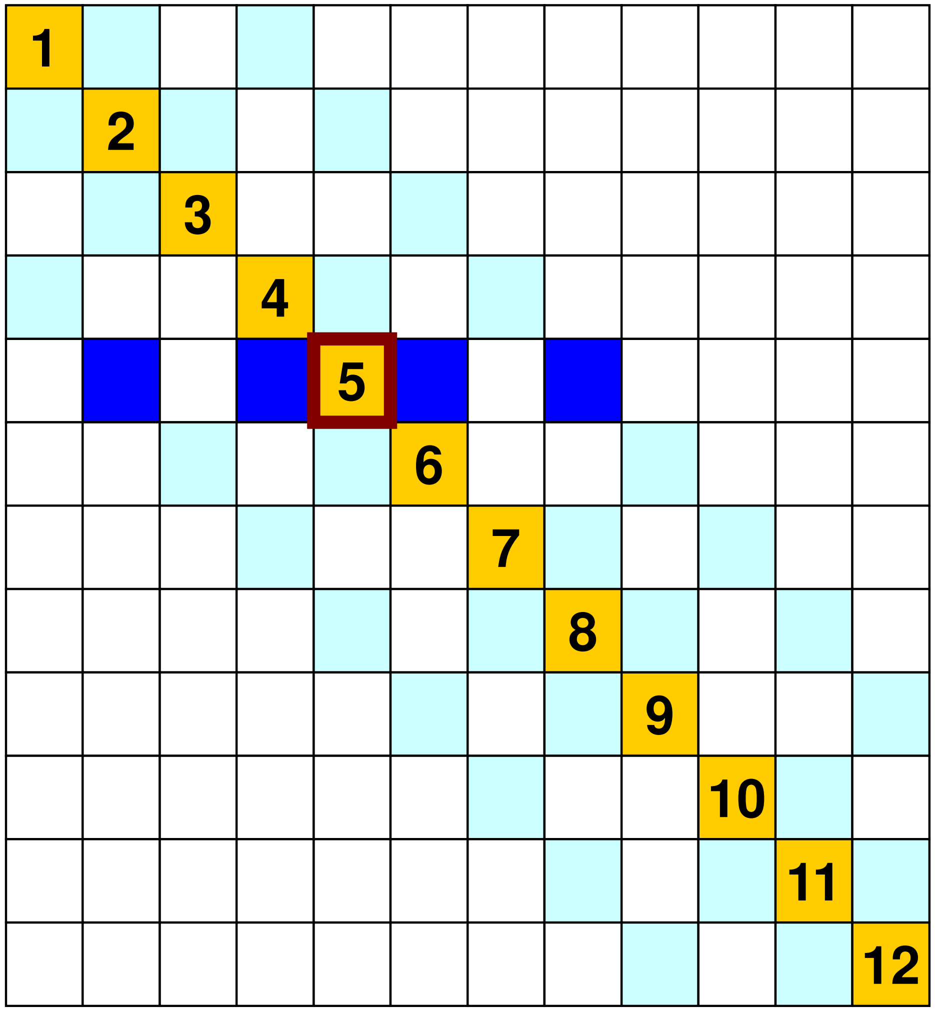
④ ➡ $a_{32} = l_{31}u_{12} + l_{32}u_{22}, \cdots \Rightarrow l_{32}, l_{42}, \cdots, l_{n2}$

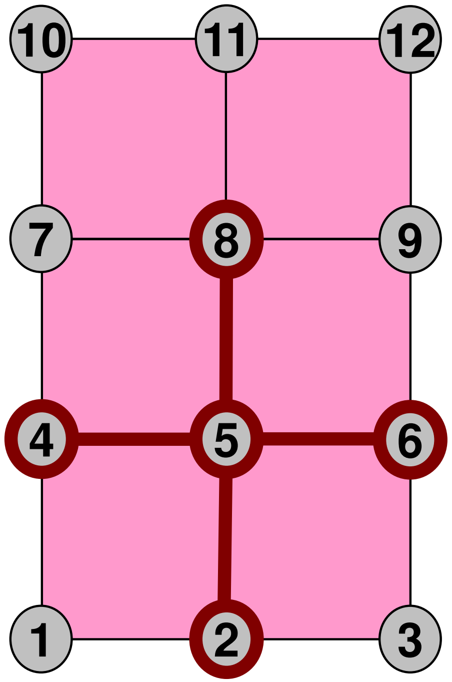
実例:5点差分





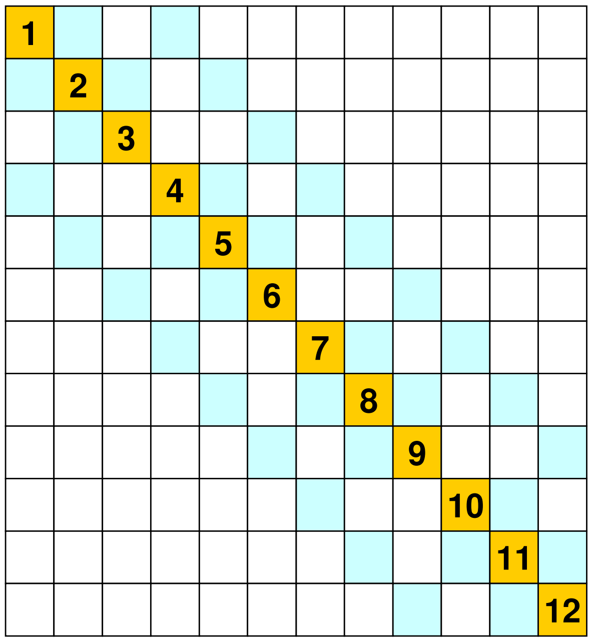
実例:5点差分

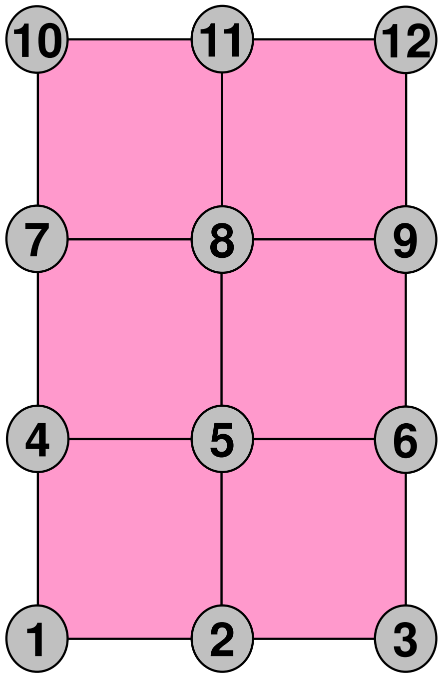




実例：係数マトリクス

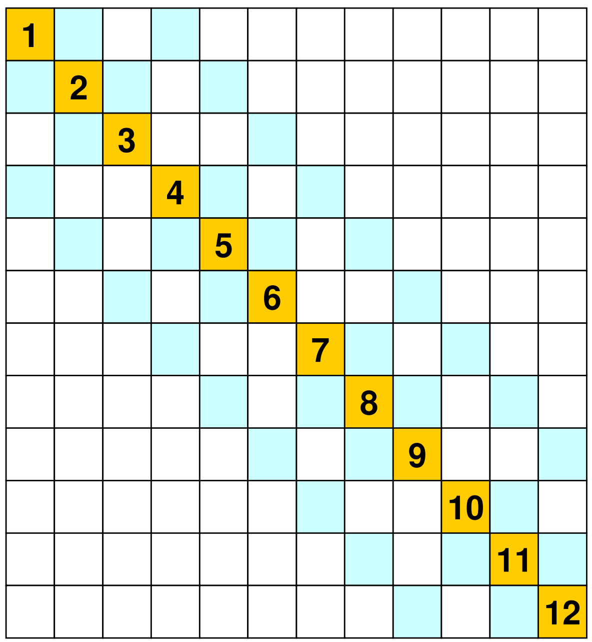
$$
\begin{pmatrix}
6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-1.00 & 6.00 & -1.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & -1.00 & 6.00 & 0.00 & 0.00 & -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-1.00 & 0.00 & 0.00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

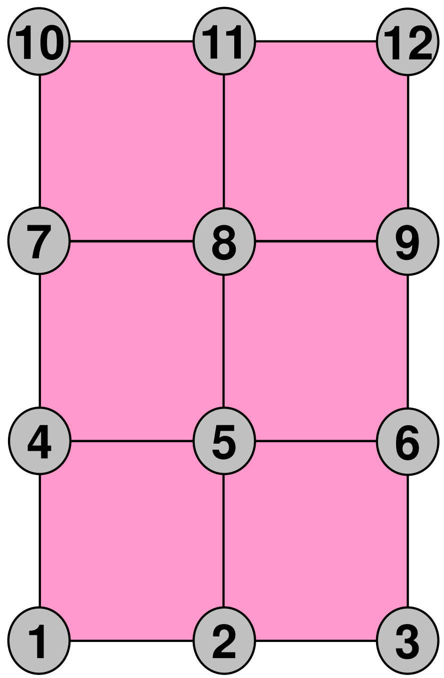




実例：解

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





完全LU分解したマトリクス

もとのマトリクス

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

LU分解したマトリクス
[L][U]同時に表示
[L]対角成分(=1)省略
(fill-inが生じている。もともと0だった成分が非ゼロになっている)

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

不完全LU分解したマトリクス（fill-in無し）

不完全LU分解した
マトリクス（fill-in無し）
[L][U]同時に表示
[L]対角成分（=1）省略

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

完全LU分解した
マトリクス
[L][U]同時に表示
[L]対角成分（=1）省略
（fill-inが生じている。もともと0だった成分が非ゼロになっている）

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

解の比較：ちょっと違う

不完全LU分解

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

0.92
1.75
2.76
3.79
4.46
5.57
6.66
7.25
8.46
9.66
10.54
11.83

完全LU分解

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

1.00
2.00
3.00
4.00
5.00
6.00
7.00
8.00
9.00
10.00
11.00
12.00

解の比較：ちょっと違う

不完全LU分解

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

0.92
1.75
2.76
3.79
4.46
5.57
6.66
7.25
8.46
9.66
10.54
11.83

対角スケーリング（PCG）

6.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	6.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	6.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	6.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	6.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	6.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	6.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.00

0.00
0.50
1.67
1.83
1.67
3.17
3.33
2.67
4.67
7.00
6.00
8.67

ILU(0), IC(0) 前処理

- Fill-inを全く考慮しない「不完全な」分解
  - 記憶容量，計算量削減
- これを解くと「不完全な」解が得られるが，本来の解とそれほどずれているわけではない
  - 問題に依存する

前処理の分類：Trade-off

Weak ⟷ **Strong**

Point Jacobi

Diagonal Blocking

ILU(0)

ILU(1)

ILU(2)

Gaussian Elimination

- Simple
- Easy to be Parallelized
- Cheap

- Complicated
- Global Dependency
- Expensive

- 背景
  - 有限体積法
  - 前処理付反復法
- **ICCG法によるポアソン方程式法ソルバーについて**
  - 実行方法
    - データ構造
  - プログラムの説明
    - 初期化
    - 係数マトリクス生成
    - ICCG法

対象とするアプリケーションの概要

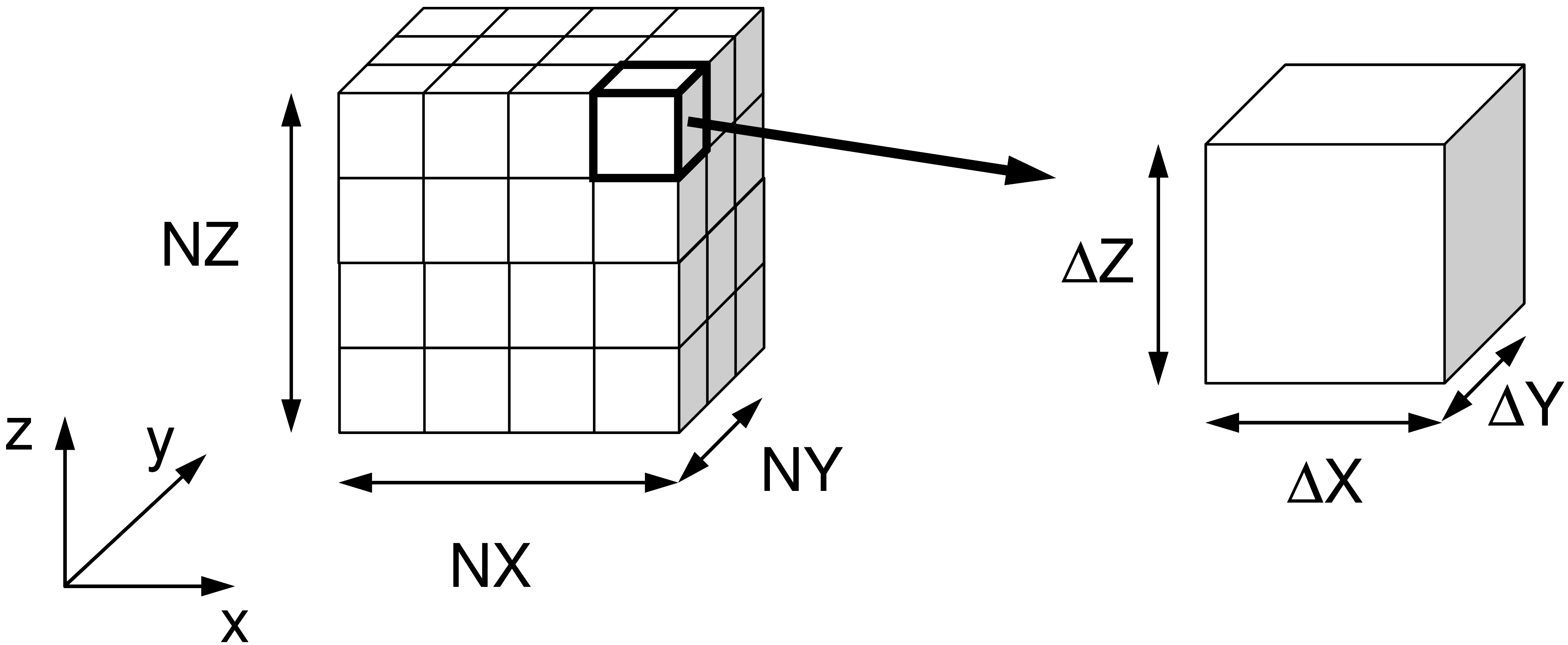
- 支配方程式：三次元ポアソン方程式

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} + f = 0$$

- 有限体積法（Finite Volume Method, **FVM**）による空間離散化
  - 任意形状の要素，要素中心で変数を定義。
  - 直接差分法（Direct Finite Difference Method）とも呼ばれる。
- 境界条件他
  - ディリクレ境界条件@Z=Zmax，体積フラックスf
- 反復法による連立一次方程式解法
  - 共役勾配法（CG）＋前処理

対象：規則正しい三次元差分格子

半非構造的に扱う



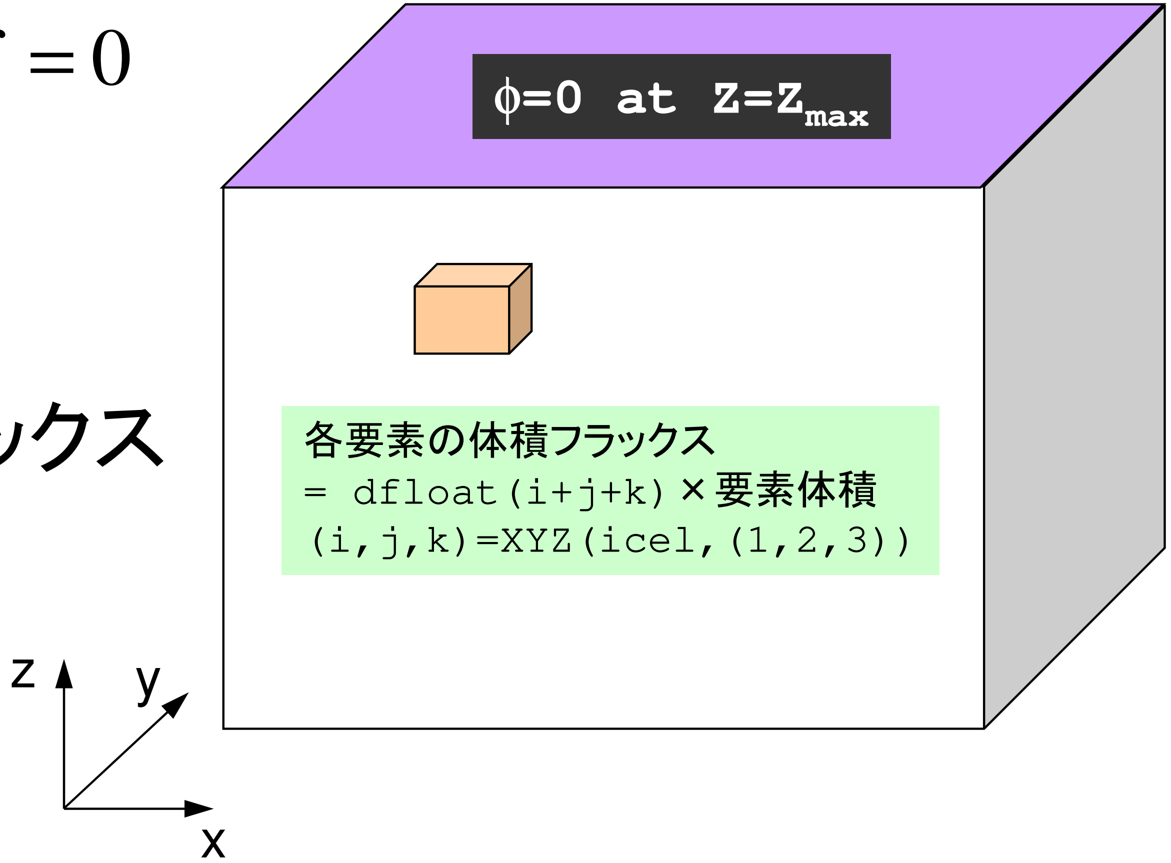
解いている問題：三次元ポアソン方程式 変数：要素中心で定義

ポアソン方程式

$$\frac{\partial^2 \phi}{\partial x^2}+\frac{\partial^2 \phi}{\partial y^2}+\frac{\partial^2 \phi}{\partial z^2}+f=0$$

<u>境界条件他</u>

- 各要素で体積フラックス
- $Z=Z_{max}$面で$\phi=0$

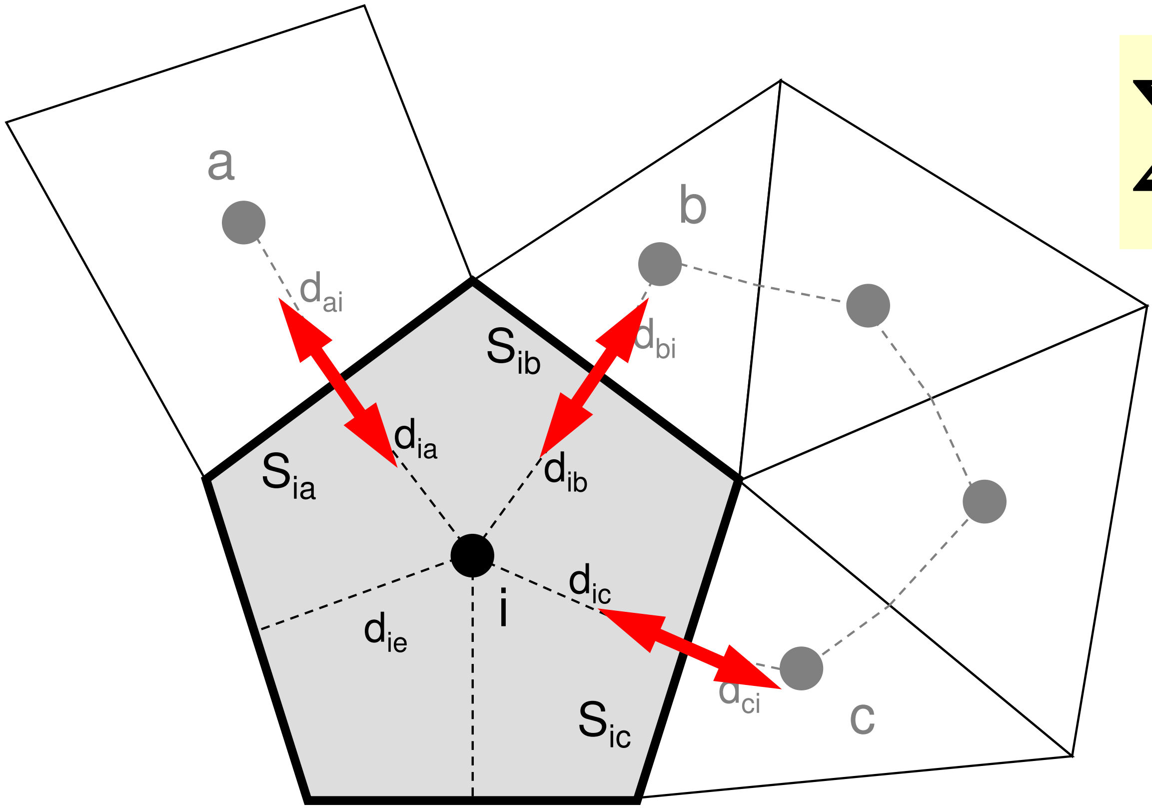


ポアソン方程式：有限体積法による離散化

$$\frac{\partial^2 \phi}{\partial x^2}+\frac{\partial^2 \phi}{\partial y^2}+\frac{\partial^2 \phi}{\partial z^2}+f=0$$

Poisson Eq. by Finite Volume Method（FVM）

面を通過するフラックス（flux，流束）の保存に着目



隣接要素との拡散

$$\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\left(\phi_k-\phi_i\right)+V_i\dot{Q}_i=0$$

体積フラックス

V_i ：要素体積

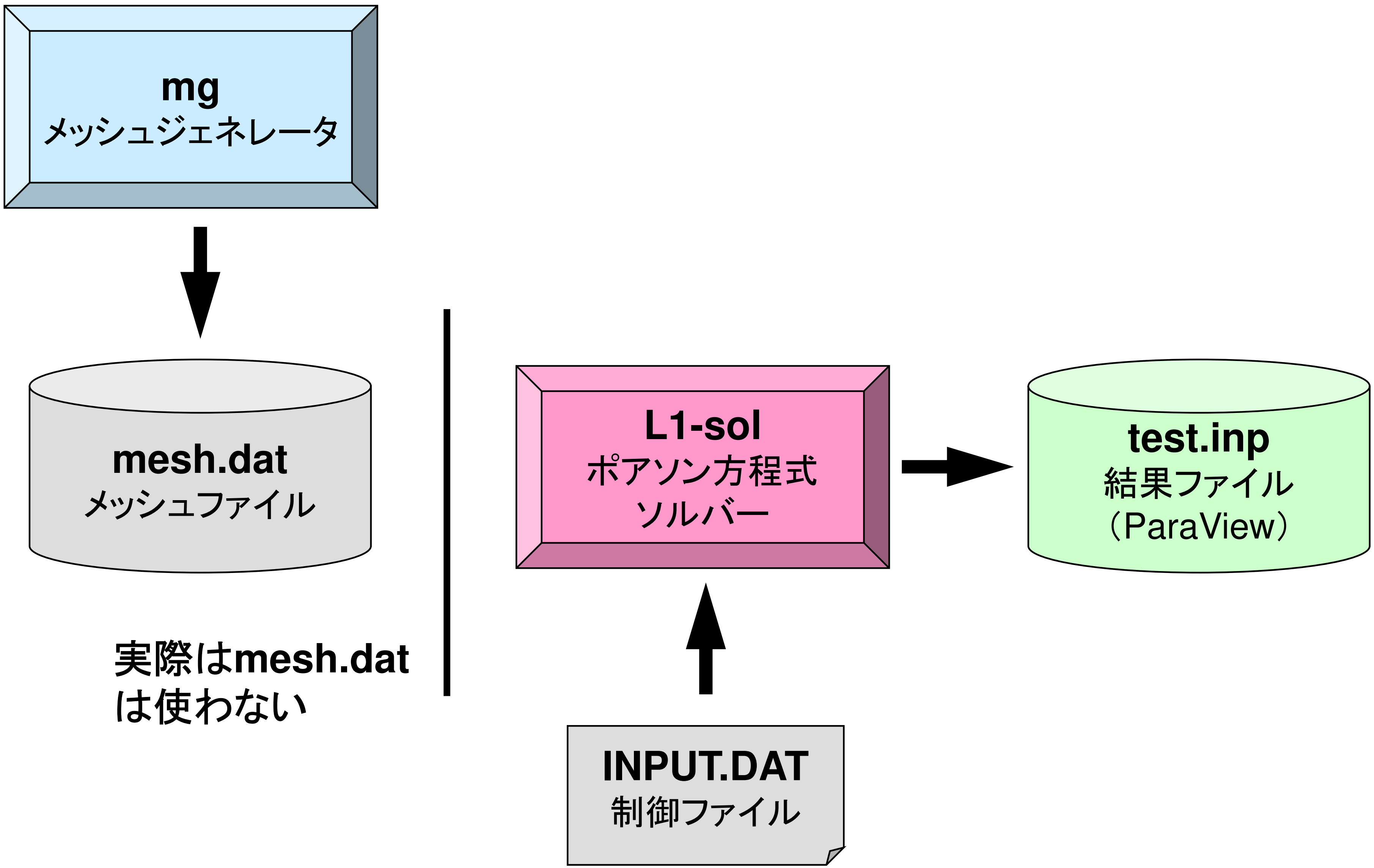
S ：表面面積

d_{ij} ：要素中心から表面までの距離

Q ：体積フラックス

プログラムの実行

プログラム，必要ファイル，実行ディレクトリ：<$P-L1>/run



プログラムの実行

コンパイル

```
$> cd <$P-L1>/run

$> gfortran -O mg.f -o mg (or cc -O mg.c -o mg)
$> ls mg
      mg

$> cd ../src
$> make
$> ls ../run/L1-sol
      L1-sol
```

Mesh Generator: **mg**

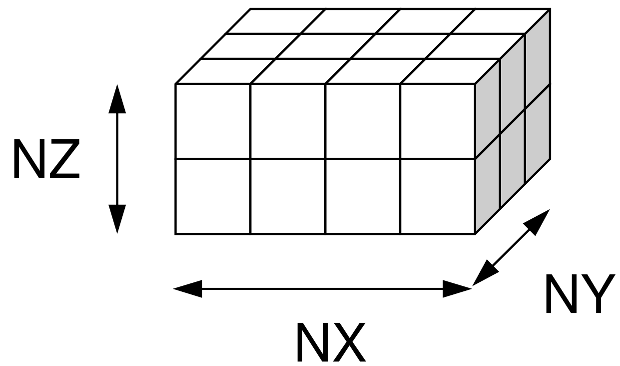
Poisson Solver (FVM): **L1-sol**

プログラムの実行

メッシュ生成

```
$> cd ../run
$> ./mg
   4 3 2
$> ls mesh.dat
     mesh.dat
```

下図のNX, NY, NZを入力すると,
「mesh.dat」が生成される



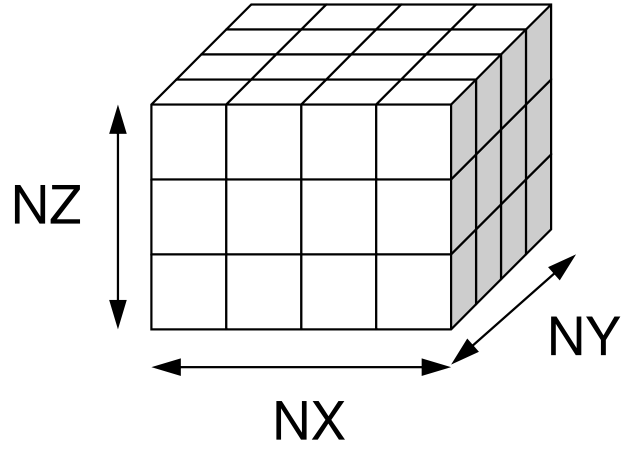
mesh.dat (1/5)

```
   4   3   2
  24
   1   0   2   0   5   0  13   1   1   1
   2   1   3   0   6   0  14   2   1   1
   3   2   4   0   7   0  15   3   1   1
   4   3   0   0   8   0  16   4   1   1
   5   0   6   1   9   0  17   1   2   1
   6   5   7   2  10   0  18   2   2   1
   7   6   8   3  11   0  19   3   2   1
   8   7   0   4  12   0  20   4   2   1
   9   0  10   5   0   0  21   1   3   1
  10   9  11   6   0   0  22   2   3   1
  11  10  12   7   0   0  23   3   3   1
  12  11   0   8   0   0  24   4   3   1
  13   0  14   0  17   1   0   1   1   2
  14  13  15   0  18   2   0   2   1   2
  15  14  16   0  19   3   0   3   1   2
  16  15   0   0  20   4   0   4   1   2
  17   0  18  13  21   5   0   1   2   2
  18  17  19  14  22   6   0   2   2   2
  19  18  20  15  23   7   0   3   2   2
  20  19   0  16  24   8   0   4   2   2
  21   0  22  17   0   9   0   1   3   2
  22  21  23  18   0  10   0   2   3   2
  23  22  24  19   0  11   0   3   3   2
  24  23   0  20   0  12   0   4   3   2
```

```
  read (21,'(10i10)') NX , NY , NZ
  read (21,'(10i10)') ICELTOT

  do i= 1, ICELTOT
    read (21, '(10i10)' ) ii, (NEIBcell(i,k), k= 1, 6), (XYZ(i,j), j= 1, 3)
  enddo
```

mesh.dat (2/5)

```
    4    3    2
   24
    1    0    2    0    5    0   13    1    1    1
    2    1    3    0    6    0   14    2    1    1
    3    2    4    0    7    0   15    3    1    1
    4    3    0    0    8    0   16    4    1    1
    5    0    6    1    9    0   17    1    2    1
    6    5    7    2   10    0   18    2    2    1
    7    6    8    3   11    0   19    3    2    1
    8    7    0    4   12    0   20    4    2    1
    9    0   10    5    0    0   21    1    3    1
   10    9   11    6    0    0   22    2    3    1
   11   10   12    7    0    0   23    3    3    1
   12   11    0    8    0    0   24    4    3    1
   13    0   14    0   17    1    0    1    1    2
   14   13   15    0   18    2    0    2    1    2
   15   14   16    0   19    3    0    3    1    2
   16   15    0    0   20    4    0    4    1    2
   17    0   18   13   21    5    0    1    2    2
   18   17   19   14   22    6    0    2    2    2
   19   18   20   15   23    7    0    3    2    2
   20   19    0   16   24    8    0    4    2    2
   21    0   22   17    0    9    0    1    3    2
   22   21   23   18    0   10    0    2    3    2
   23   22   24   19    0   11    0    3    3    2
   24   23    0   20    0   12    0    4    3    2
```



X,Y,Z方向の要素数

```
read (21,'(10i10)') NX , NY , NZ
read (21,'(10i10)') ICELTOT

do i= 1, ICELTOT
  read (21, '(10i10)' ) ii, (NEIBcell(i,k), k= 1, 6), (XYZ(i,j), j= 1, 3)
enddo
```

mesh.dat (3/5)

要素数:NX×NY×NZ

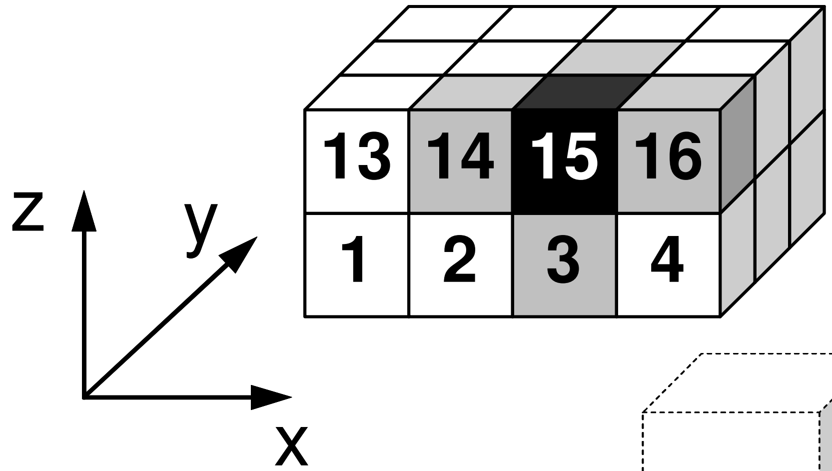
```
   4   3   2
  24
   1   0   2   0   5   0  13   1   1   1
   2   1   3   0   6   0  14   2   1   1
   3   2   4   0   7   0  15   3   1   1
   4   3   0   0   8   0  16   4   1   1
   5   0   6   1   9   0  17   1   2   1
   6   5   7   2  10   0  18   2   2   1
   7   6   8   3  11   0  19   3   2   1
   8   7   0   4  12   0  20   4   2   1
   9   0  10   5   0   0  21   1   3   1
  10   9  11   6   0   0  22   2   3   1
  11  10  12   7   0   0  23   3   3   1
  12  11   0   8   0   0  24   4   3   1
  13   0  14   0  17   1   0   1   1   2
  14  13  15   0  18   2   0   2   1   2
  15  14  16   0  19   3   0   3   1   2
  16  15   0   0  20   4   0   4   1   2
  17   0  18  13  21   5   0   1   2   2
  18  17  19  14  22   6   0   2   2   2
  19  18  20  15  23   7   0   3   2   2
  20  19   0  16  24   8   0   4   2   2
  21   0  22  17   0   9   0   1   3   2
  22  21  23  18   0  10   0   2   3   2
  23  22  24  19   0  11   0   3   3   2
  24  23   0  20   0  12   0   4   3   2
```

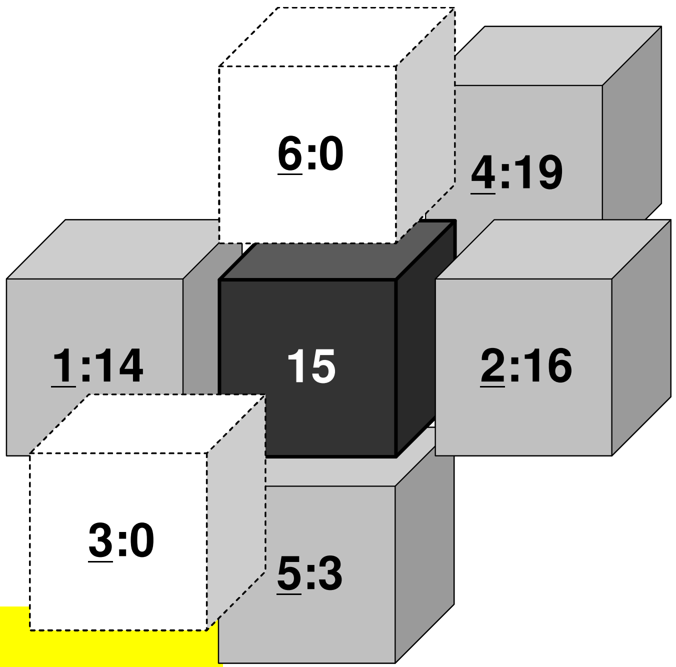
```
read (21,'(10i10)') NX , NY , NZ
read (21,'(10i10)') ICELTOT

do i= 1, ICELTOT
  read (21, '(10i10)' ) ii, (NEIBcell(i,k), k= 1, 6), (XYZ(i,j), j= 1, 3)
enddo
```

mesh.dat（4/5）

```
     4     3     2
    24
     1     0     2     0     5     0    13     1     1     1
     2     1     3     0     6     0    14     2     1     1
     3     2     4     0     7     0    15     3     1     1
     4     3     0     0     8     0    16     4     1     1
     5     0     6     1     9     0    17     1     2     1
     6     5     7     2    10     0    18     2     2     1
     7     6     8     3    11     0    19     3     2     1
     8     7     0     4    12     0    20     4     2     1
     9     0    10     5     0     0    21     1     3     1
    10     9    11     6     0     0    22     2     3     1
    11    10    12     7     0     0    23     3     3     1
    12    11     0     8     0     0    24     4     3     1
    13     0    14     0    17     1     0     1     1     2
    14    13    15     0    18     2     0     2     1     2
    15    14    16     0    19     3     0     3     1     2
    16    15     0     0    20     4     0     4     1     2
    17     0    18    13    21     5     0     1     2     2
    18    17    19    14    22     6     0     2     2     2
    19    18    20    15    23     7     0     3     2     2
    20    19     0    16    24     8     0     4     2     2
    21     0    22    17     0     9     0     1     3     2
    22    21    23    18     0    10     0     2     3     2
    23    22    24    19     0    11     0     3     3     2
    24    23     0    20     0    12     0     4     3     2
```

隣接要素：NEIBcell(i,k)

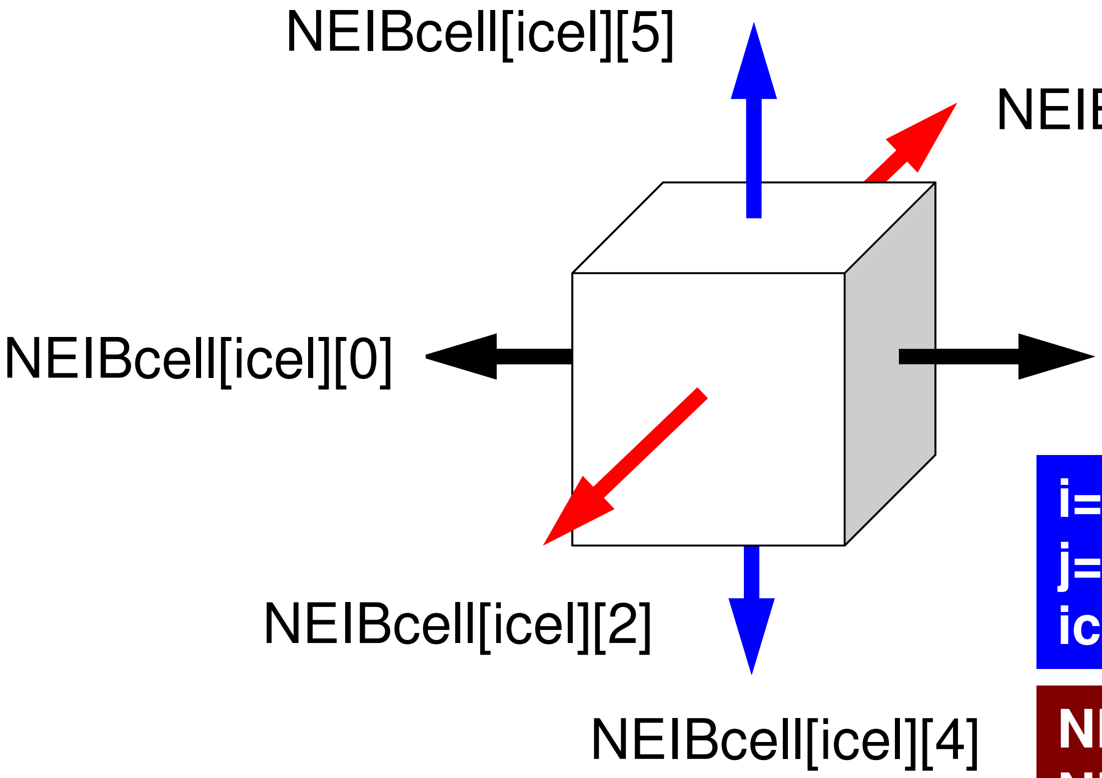





```
read (21,'(10i10)') NX , NY , NZ
read (21,'(10i10)') ICELTOT

do i= 1, ICELTOT
  read (21, '(10i10)' ) ii, (NEIBcell(i,k), k= 1, 6), (XYZ(i,j), j= 1, 3)
enddo
```

1項目目は通し番号です（読み飛ばし）

NEIBcell：隣接している要素番号 境界面の場合は0



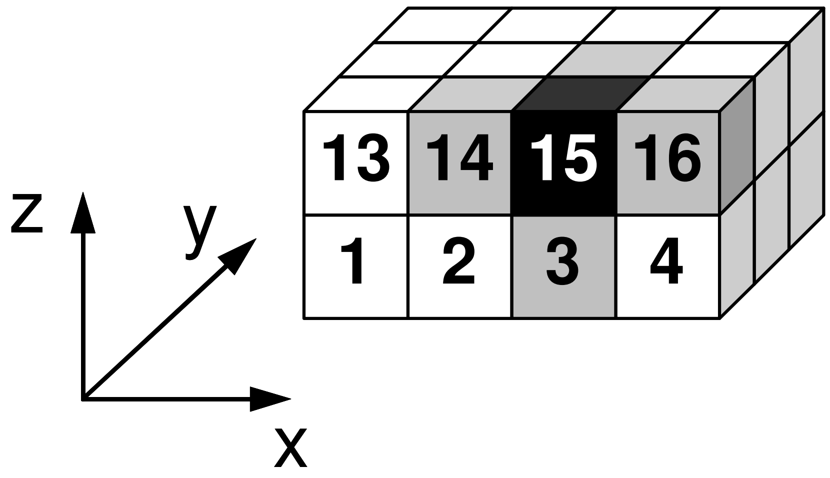
i= XYZ[icel][0]
j= XYZ[icel][1], k= XYZ[icel][2]
icel= k*NX*NY + j*NX + i

NEIBcell[icel][0]= icel – 1 + 1
NEIBcell[icel][1]= icel + 1 + 1
NEIBcell[icel][2]= icel – NX + 1
NEIBcell[icel][3]= icel + NX + 1
NEIBcell[icel][4]= icel – NX*NY + 1
NEIBcell[icel][5]= icel + NX*NY + 1

mesh.dat（5/5）

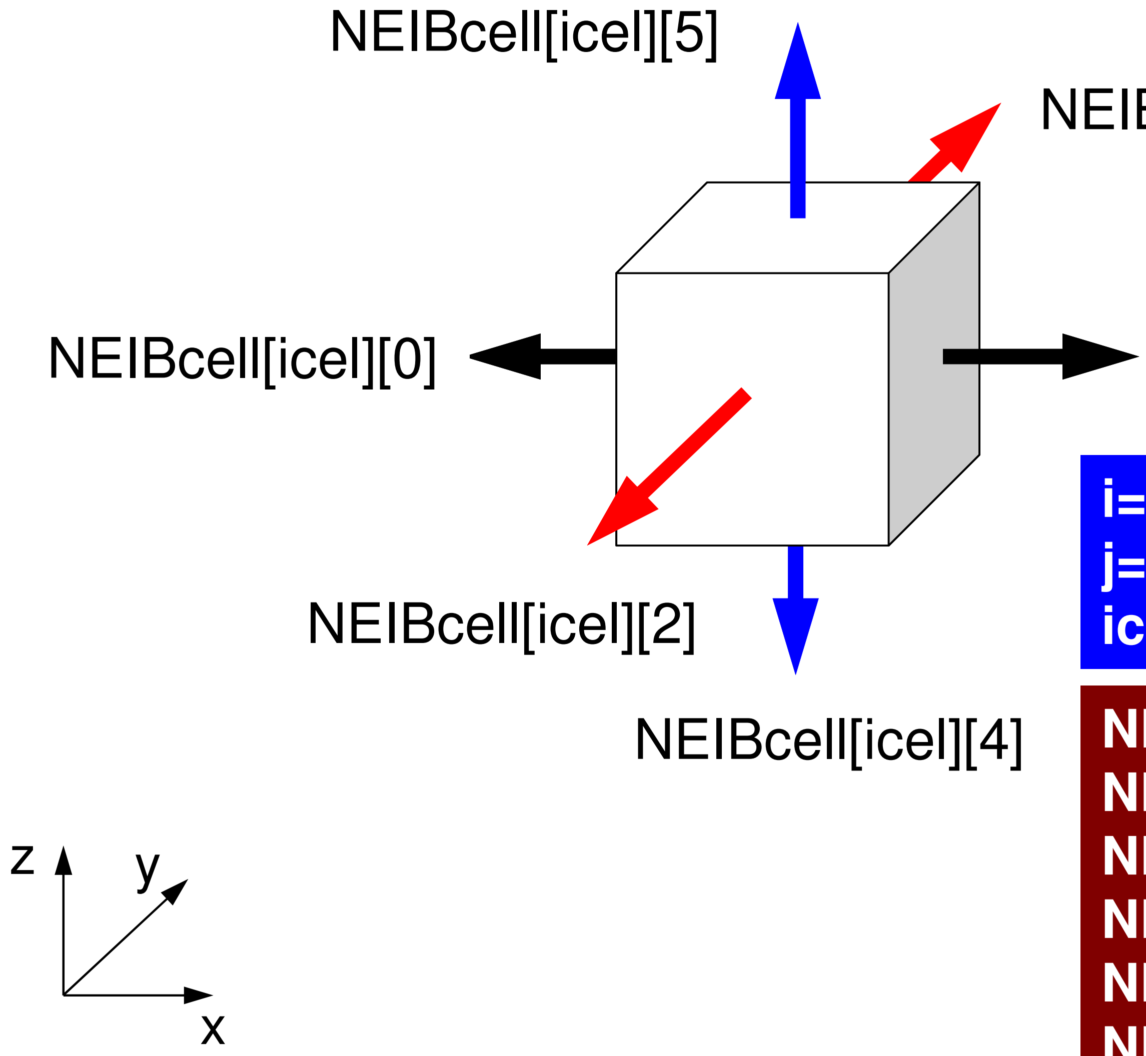
```
 4  3  2
24
 1  0  2  0  5  0 13  1  1  1
 2  1  3  0  6  0 14  2  1  1
 3  2  4  0  7  0 15  3  1  1
 4  3  0  0  8  0 16  4  1  1
 5  0  6  1  9  0 17  1  2  1
 6  5  7  2 10  0 18  2  2  1
 7  6  8  3 11  0 19  3  2  1
 8  7  0  4 12  0 20  4  2  1
 9  0 10  5  0  0 21  1  3  1
10  9 11  6  0  0 22  2  3  1
11 10 12  7  0  0 23  3  3  1
12 11  0  8  0  0 24  4  3  1
13  0 14  0 17  1  0  1  1  2
14 13 15  0 18  2  0  2  1  2
15 14 16  0 19  3  0  3  1  2
16 15  0  0 20  4  0  4  1  2
17  0 18 13 21  5  0  1  2  2
18 17 19 14 22  6  0  2  2  2
19 18 20 15 23  7  0  3  2  2
20 19  0 16 24  8  0  4  2  2
21  0 22 17  0  9  0  1  3  2
22 21 23 18  0 10  0  2  3  2
23 22 24 19  0 11  0  3  3  2
24 23  0 20  0 12  0  4  3  2
```

X,Y,Z方向の位置：XYZ(i,j)




```
read (21,'(10i10)') NX , NY , NZ
read (21,'(10i10)') ICELTOT

do i= 1, ICELTOT
  read (21, '(10i10)' ) ii, (NEIBcell(i,k), k= 1, 6), (XYZ(i,j), j= 1, 3)
enddo
```

NEIBcell：隣接している要素番号 境界面の場合は0



i= XYZ[icel][0]
j= XYZ[icel][1], k= XYZ[icel][2]
icel= k*NX*NY + j*NX + i

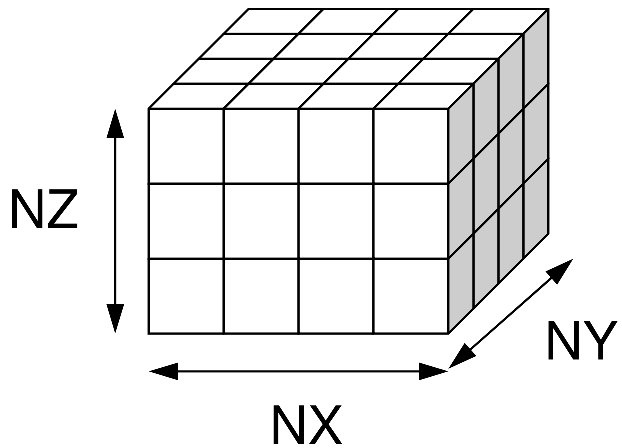
NEIBcell[icel][0]= icel – 1 + 1
NEIBcell[icel][1]= icel + 1 + 1
NEIBcell[icel][2]= icel – NX + 1
NEIBcell[icel][3]= icel + NX + 1
NEIBcell[icel][4]= icel – NX*NY + 1
NEIBcell[icel][5]= icel + NX*NY + 1

プログラムの実行

制御データ「<$P-L1>/run/INPUT.DAT」の作成

```
32 32 32                                   NX/NY/NZ
1                                          MEHOD 1:2:3
1.00e-00 1.00e-00 1.00e-00                 DX/DY/DZ
1.0e-08                                    EPSICCG
```

- NX, NY, NZ
  - 各方向のメッシュ数
- METHOD
  - 前処理行列の作成方法：次ページ
- DX, DY, DZ
  - 各要素のX,Y,Z方向辺長さ
- EPSICCG
  - ICCG法の収束判定値



前処理手法の選択

```
32 32 32                                NX/NY/NZ
1                                       MEHOD 1:2:3
1.00e-00 1.00e-00 1.00e-00              DX/DY/DZ
1.0e-08                                 EPSICCG
```

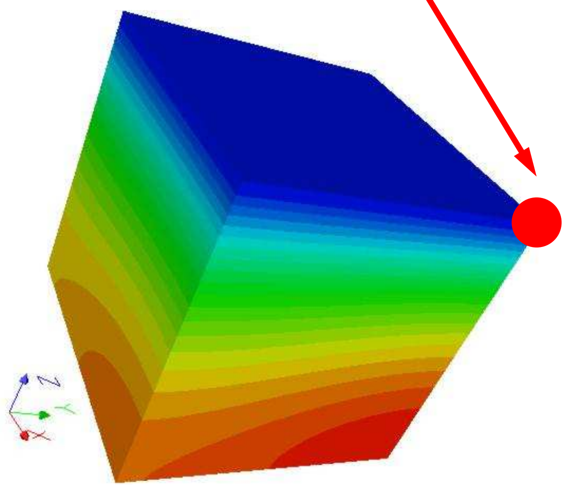
- METHOD=1　不完全修正コレスキー分解（非対角項保存）
- METHOD=2　不完全修正コレスキー分解（Fortranのみ）
- METHOD=3　対角スケーリング（点ヤコビ）

- METHOD=1,2,3について計算してみよ　！

プログラムの実行

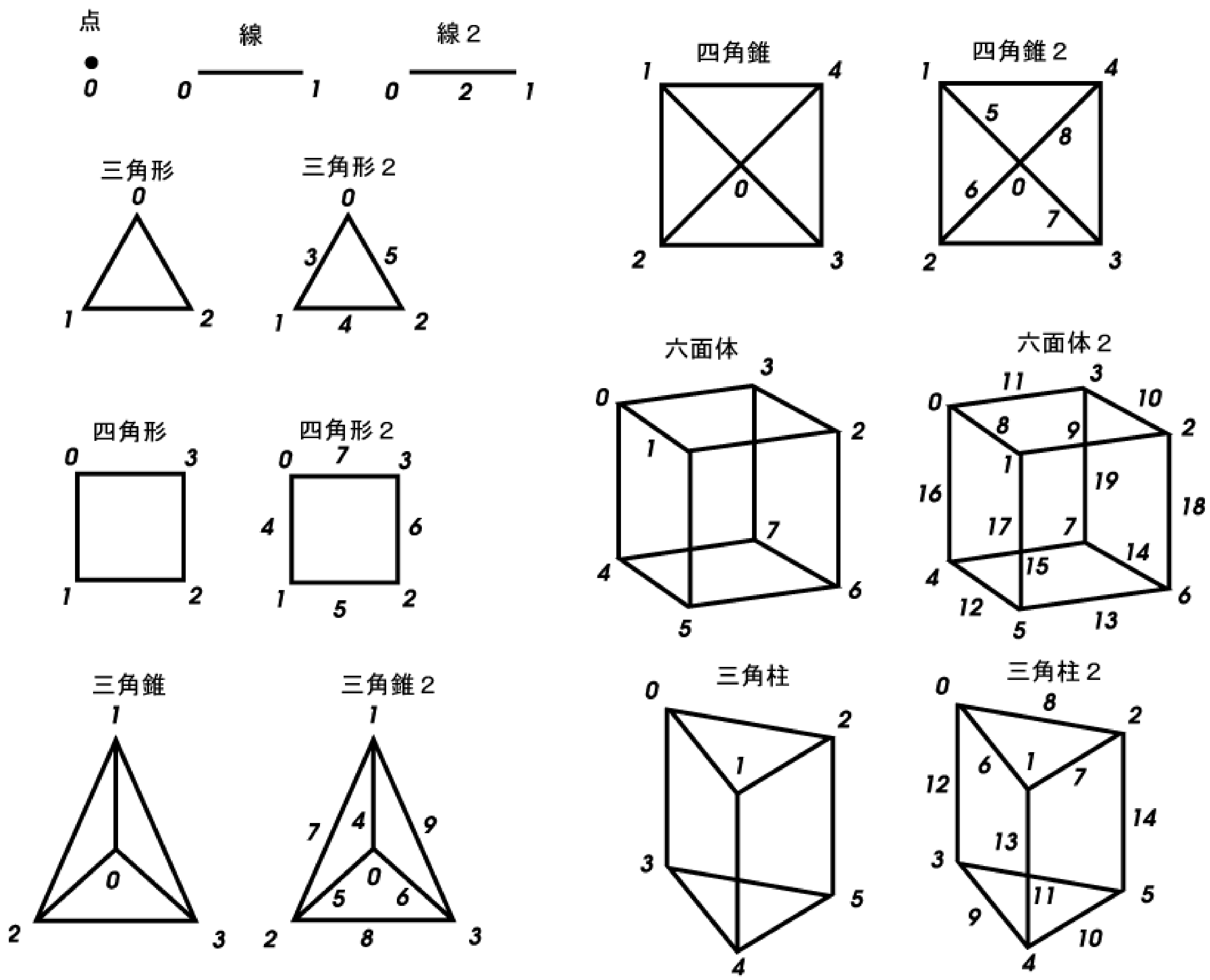
計算実行，ポスト処理

```
$> cd <$P-L1>/run
$> ./L1-sol
    1      4.504513e+00             1反復目の残差
   75      8.377861e-09             収束時の反復回数残差(<10^-8)

     32768     9.297409e+02         下図●点の答え

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



UCD Format (1/2)
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



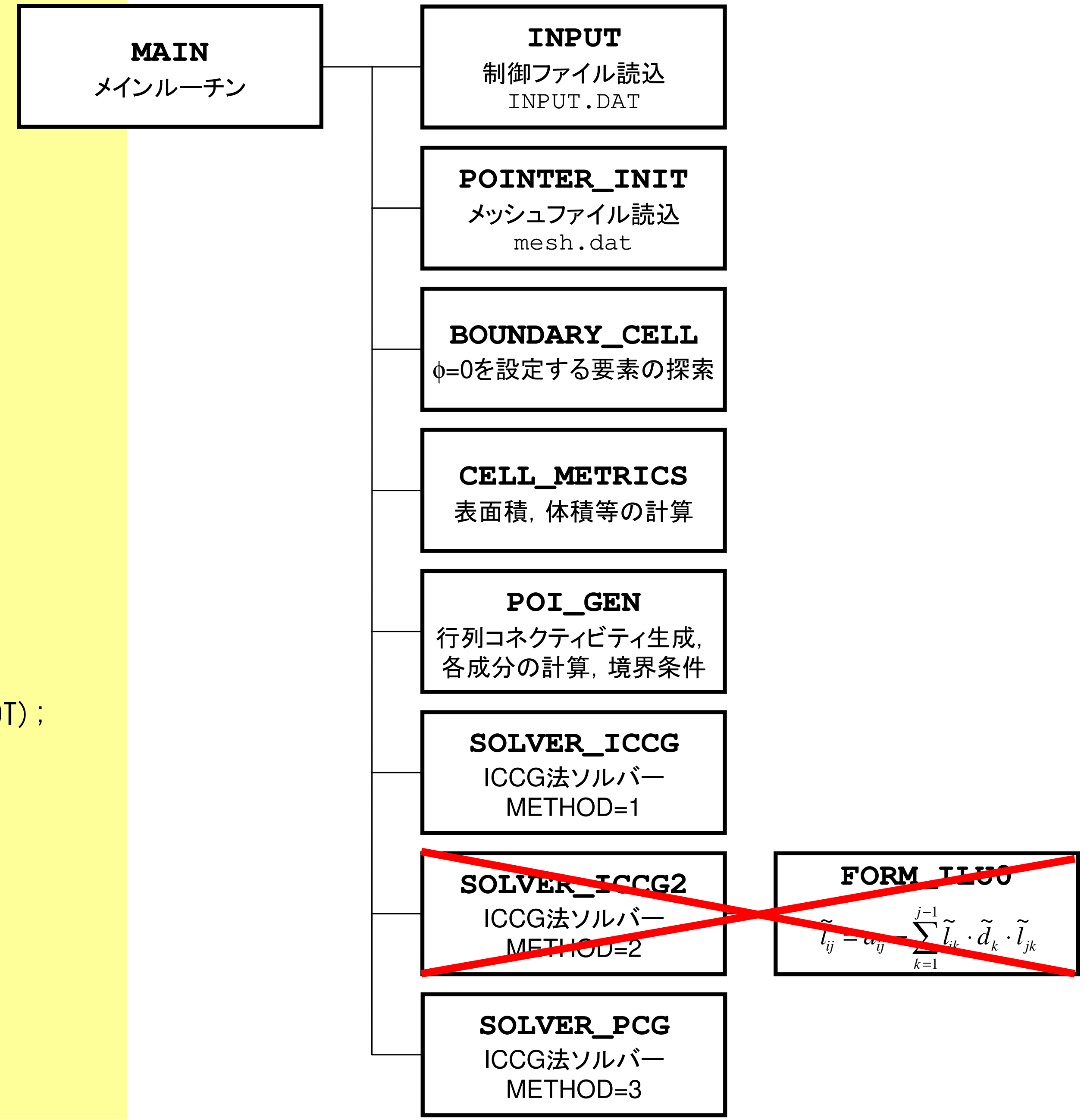
UCD Format (2/2)

- 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.

- 背景
  - 有限体積法
  - 前処理付反復法
- **ICCG法によるポアソン方程式法ソルバーについて**
  - 実行方法
    - データ構造
  - プログラムの説明
    - 初期化
    - 係数マトリクス生成
    - ICCG法

プログラムの構成

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

#include ”struct.h”
#include ”pcg.h”
#include ”input.h “ ...

int
main()
{
  double *WK;
  int NPL, NPU; ISET, ITR, IER; icel, ic0, i;
  double xN, xL, xU; Stime, Etime;

  if(INPUT()) goto error;
  if(POINTER_INIT()) goto error;
  if(BOUNDARY_CELL()) goto error;
  if(CELL_METRICS()) goto error;
  if(POI_GEN()) goto error;

  memset(PHI, 0.0, sizeof(double)*ICELTOT);
  ISET = 0;
  WK = (double *)malloc(sizeof(double)*ICELTOT);

  if(METHOD==1){
     if(solve_ICCG(...)) goto error;
     } else if(METHOD==3){
     if(solve_PCG(...)) goto error;}

  if(OUTUCD()) goto error;
  return 0;
error:
  return -1;
}
```



struct.h

```
#ifndef __H_STRUCT
#define __H_STRUCT

#include <omp.h>

int ICELTOT, ICELTOTp, N;
int NX, NY, NZ, NXP1, NYP1, NZP1, IBNODTOT;
int NXc, NYc, NZc;

double DX, DY, DZ, XAREA, YAREA, ZAREA;
double RDX, RDY, RDZ, RDX2, RDY2, RDZ2, R2DX, R2DY, R2DZ;
double *VOLCEL, *VOLNOD, *RVC, *RVN;

int **XYZ, **NEIBcell;

int ZmaxCELtot;
int *BC_INDEX, *BC_NOD;
int *ZmaxCEL;

int **IWKX;
double **FCV;

int my_rank, PETOT, PEsmpTOT;

#endif /* __H_STRUCT */
```

ICELTOT
要素数（NX×NY×NZ）

N
節点数

NX,NY,NZ
x,y,z方向要素数

NXP1,NYP1,NZP1
x,y,z方向節点数

IBNODTOT
NXP1×NYP1

XYZ[ICELTOT][3]
要素座標

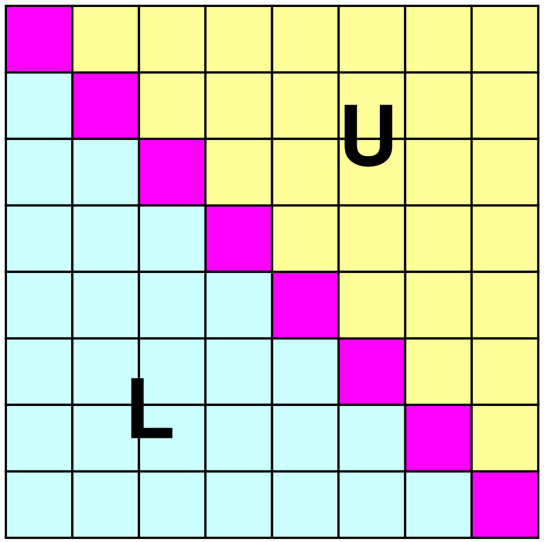
NEIBcell[ICELTOT][6]
隣接要素

pcg.h (1/5)

```
#ifndef __H_PCG
#define __H_PCG
        static int N2 = 256;
        int NUmax, NLmax, NCOLORtot, NCOLORk, NU, NL;
        int METHOD, ORDER_METHOD;
        double EPSICCG;

        double *D, *PHI, *BFORCE;
        double *AL, *AU;

        int *INL, *INU, *COLORindex;
        int *indexL, *indexU;
        int *OLDtoNEW, *NEWtoOLD;
        int **IAL, **IAU;
        int *itemL, *itemU;
        int NPL, NPU;
#endif /* __H_PCG */
```

扱う行列：疎行列
（自分の周辺のみと接続）
⇒ 非ゼロ成分のみを記憶する

上下三角成分を別々に記憶



pcg.h (2/5)

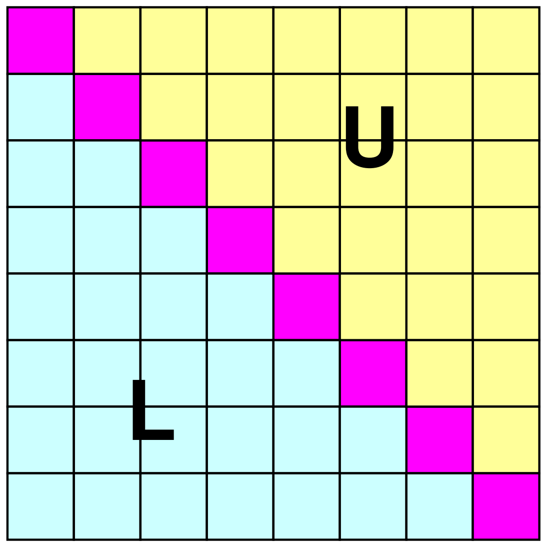
```
#ifndef __H_PCG
#define __H_PCG
        static int N2 = 256;
        int NUmax, NLmax, NCOLORtot, NCOLORk, NU, NL;
        int METHOD, ORDER_METHOD;
        double EPSICCG;

        double *D, *PHI, *BFORCE;
        double *AL, *AU;

        int *INL, *INU, *COLORindex;
        int *indexL, *indexU;
        int *OLDtoNEW, *NEWtoOLD;
        int **IAL, **IAU;
        int *itemL, *itemU;
        int NPL, NPU;
#endif /* __H_PCG */
```

補助配列

下三角成分(列番号):
非対角成分で自分より要素番号が小さい。
IAL[i][icou] < i

上三角成分(列番号):
非対角成分で自分より要素番号が大きい。
IAU[i[icou] > i

INL{ICELTOT}	非零下三角成分の数
IAL[ICELTOT][NL]	非零下三角成分（列番号）
INU[ICELTOT]	非零上三角成分の数
IAU[ICELTOT][NU]	非零上三角成分（列番号）
NU,NL	非零上下三角成分の最大数（ここでは6）
indexL[ICELTOT+1]	各行の非零下三角成分数（CRS）
indexU[ICELTOT+1]	各行の非零上三角成分数（CRS）
NPL, NPU	非零上下三角成分数の合計（CRS）
itemL[NPL],itemU[NPU]	比零上下三角成分（列番号）（CRS）



pcg.h (3/5)

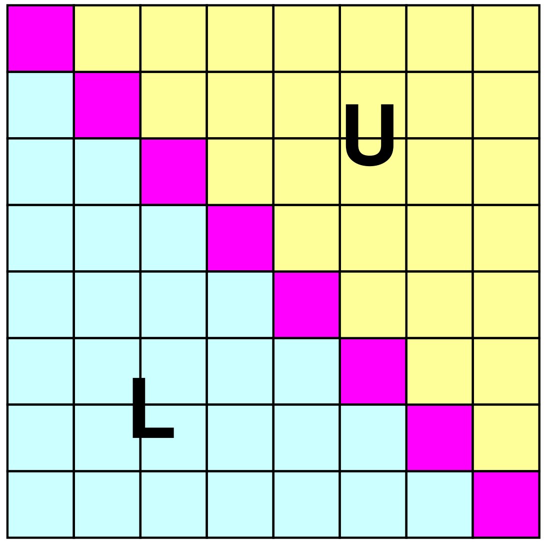
```
#ifndef __H_PCG
#define __H_PCG
        static int N2 = 256;
        int NUmax, NLmax, NCOLORtot, NCOLORk, NU, NL;
        int METHOD, ORDER_METHOD;
        double EPSICCG;

        double *D, *PHI, *BFORCE;
        double *AL, *AU;

        int *INL, *INU, *COLORindex;
        int *indexL, *indexU;
        int *OLDtoNEW, *NEWtoOLD;
        int **IAL, **IAU;
        int *itemL, *itemU;
        int NPL, NPU;
#endif /* __H_PCG */
```

補助配列

下三角成分(列番号):
IAL[i][icou] < i
その個数が INL[i]

上三角成分(列番号):
IAU[i][icou] > i
その個数がINU[i]

INL{ICELTOT}	**非零下三角成分の数**
IAL[ICELTOT][NL]	非零下三角成分（列番号）
INU[ICELTOT]	**非零上三角成分の数**
IAU[ICELTOT][NU]	非零上三角成分（列番号）
NU,NL	非零上下三角成分の最大数（ここでは6）
indexL[ICELTOT+1]	各行の非零下三角成分数（CRS）
indexU[ICELTOT+1]	各行の非零上三角成分数（CRS）
NPL, NPU	非零上下三角成分数の合計（CRS）
itemL[NPL],itemU[NPU]	比零上下三角成分（列番号）（CRS）



pcg.h (4/5)

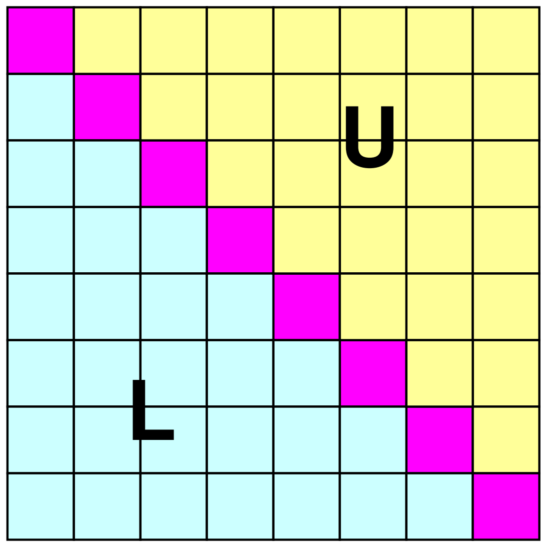
```
#ifndef __H_PCG
#define __H_PCG
        static int N2 = 256;
        int NUmax, NLmax, NCOLORtot, NCOLORk, NU, NL;
        int METHOD, ORDER_METHOD;
        double EPSICCG;

        double *D, *PHI, *BFORCE;
        double *AL, *AU;

        int *INL, *INU, *COLORindex;
        int *indexL, *indexU;
        int *OLDtoNEW, *NEWtoOLD;
        int **IAL, **IAU;
        int *itemL, *itemU;
        int NPL, NPU;
#endif /* __H_PCG */
```

<u>補助配列</u>

下三角成分（列番号）：
IAL[i][icou] < i
その個数が INL[i]

上三角成分（列番号）：
IAU[i][icou] > i
その個数がINU[i]

INL{ICELTOT}	非零下三角成分の数
IAL[ICELTOT][NL]	非零下三角成分（列番号）
INU[ICELTOT]	非零上三角成分の数
IAU[ICELTOT][NU]	非零上三角成分（列番号）
NU,NL	非零上下三角成分の最大数（ここでは6）
indexL[ICELTOT+1]	各行の非零下三角成分数（CRS）
indexU[ICELTOT+1]	各行の非零上三角成分数（CRS）
NPL, NPU	非零上下三角成分数の合計（CRS）
itemL[NPL],itemU[NPU]	比零上下三角成分（列番号）（CRS）



pcg.h (5/5)

```
#ifndef __H_PCG
#define __H_PCG
        static int N2 = 256;
        int NUmax, NLmax, NCOLORtot, NCOLORk, NU, NL;
        int METHOD, ORDER_METHOD;
        double EPSICCG;

        double *D, *PHI, *BFORCE;
        double *AL, *AU;

        int *INL, *INU, *COLORindex;
        int *indexL, *indexU;
        int *OLDtoNEW, *NEWtoOLD;
        int **IAL, **IAU;
        int *itemL, *itemU;
        int NPL, NPU;
#endif /* __H_PCG */
```

METHOD	前処理手法（=1, =2, =3）
EPSICCG	収束打切残差
D [ICELTOT]	係数行列の対角成分
PHI [ICLETOT]	従属変数
BFORCE[ICELTOT]	連立一次方程式の右辺ベクトル
AL[NPL], AU[NPU]	係数行列の非零上下三角成分（CRS）

行列関係変数：まとめ

配列・変数名	型	内　容
D[N]	**R**	対角成分，（N：全メッシュ数=ICELTOT）
BFORCE[N]	**R**	右辺ベクトル
PHI[N]	**R**	未知数ベクトル
indexL[N+1]	**I**	各行の非零下三角成分数（CRS）
indexU[N+1]	**I**	各行の非零上三角成分数（CRS）
NPL	**I**	非零下三角成分総数（CRS）
NPU	**I**	非零上三角成分総数（CRS）
itemL[NPL]	**I**	非零下三角成分（列番号）（CRS）
itemU[NPU]	**I**	非零下三角成分（列番号）（CRS）
AL[NPL]	**R**	非零下三角成分（係数）（CRS）
AU[NPU]	**R**	非零上三角成分（係数）（CRS）

行列関係変数：まとめ（補助配列）

配列・変数名	型	内　容
NL,NU	**I**	各行の非零上下三角成分の最大数（ここでは6）
INL[N]	**I**	各行の非零下三角成分数
INU[N]	**I**	各行の非零上三角成分数
IAL[N][NL]	**I**	各行の非零下三角成分に対応する列番号
IAU[N][NU]	**I**	各行の非零上三角成分に対応する列番号

補助配列を使う理由

① NPL，NPUの値が前以てわからない
② 後掲の並び替え（ordering，reordering）のとき CRS形式ではやりにくい

行列ベクトル積：{q}=[A]{p}

```
for (i=0; i<N; i++) {
    q[i]= D[i] * p[i];
    for (j=indexL[i]; j<indexL[i+1]; j++) {
        q[i] += AL[j] * p[itemL[j]-1];
    }
    for (j=indexU[i]; j<indexU[i+1]; j++) {
        q[i] += AU[j] * p[itemU[j]-1];
    }
}
```

itemL, itemUの中身（列番号）は「0」ではなく，「1」から始まっている

プログラムの構成

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

#include "struct.h"
#include "pcg.h"
#include "input.h " ...

int
main()
{
  double *WK;
  int NPL, NPU; ISET, ITR, IER; icel, ic0, i;
  double xN, xL, xU; Stime, Etime;

  if(INPUT()) goto error;
  if(POINTER_INIT()) goto error;
  if(BOUNDARY_CELL()) goto error;
  if(CELL_METRICS()) goto error;
  if(POI_GEN()) goto error;

  memset(PHI, 0.0, sizeof(double)*ICELTOT);
  ISET = 0;
  WK = (double *)malloc(sizeof(double)*ICELTOT);

  if(METHOD==1){
      if(solve_ICCG(...)) goto error;
      } else if(METHOD==3){
      if(solve_PCG(...)) goto error;}

  if(OUTUCD()) goto error;
  return 0;
error:
  return -1;
}
```

MAIN
メインルーチン

- **INPUT**
  制御ファイル読込
  INPUT.DAT
- **POINTER_INIT**
  メッシュファイル読込
  mesh.dat
- **BOUNDARY_CELL**
  ϕ=0を設定する要素の探索
- **CELL_METRICS**
  表面積，体積等の計算
- **POI_GEN**
  行列コネクティビティ生成，
  各成分の計算，境界条件
- **SOLVER_ICCG**
  ICCG法ソルバー
  METHOD=1
- ~~**SOLVER_ICCG2**~~
  ~~ICCG法ソルバー~~
  ~~METHOD=2~~
  - ~~**FORM_ILU0**~~
    $\cancel{\tilde{l}_{ij} = a_{ij} - \sum_{k=1}^{j-1} \tilde{l}_{ik} \cdot \tilde{d}_k \cdot \tilde{l}_{jk}}$
- **SOLVER_PCG**
  ICCG法ソルバー
  METHOD=3

input:「INPUT.DAT」の読み込み

```
#include <stdio.h>; <stdlib.h>; <string.h>; <errno.h>
#include "struct_ext.h "; "pcg_ext.h "; "input.h"

extern int
INPUT(void)
{
#define BUF_SIZE 1024
        char line[BUF_SIZE];
        char CNTFIL[81];
        double OMEGA;
        FILE *fp11;

        if((fp11 = fopen("INPUT.DAT", "r")) == NULL) {
            fprintf(stderr, "Error: %s¥n", strerror(errno));
            return -1;
        }
        fgets(line, BUF_SIZE, fp11); sscanf(line, "%d%d%d", &NX, &NY, &NZ);
        fgets(line, BUF_SIZE, fp11); sscanf(line, "%d", &METHOD);
        fgets(line, BUF_SIZE, fp11); sscanf(line, "%le%le%le", &DX, &DY, &DZ);
        fgets(line, BUF_SIZE, fp11); sscanf(line, "%le", &EPSICCG);
        fgets(line, BUF_SIZE, fp11);

        fclose(fp11);
        return 0;
}
```

```
32 32 32                                NX/NY/NZ
1                                       MEHOD 1-3
1.00e-00 1.00e-00 1.00e-00              DX/DY/DZ
1.0e-08                                 EPSICCG
```

pointer_init(1/3):「mesh.dat」の作成

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

#include "struct_ext.h"
#include "pcg_ext.h"
#include "pointer_init.h"
#include "allocate.h"

extern int
POINTER_INIT(void)
{

        int icel, ipe, i, j, k;

/*
 * INIT.
 */

  ICELTOT = NX * NY * NZ;

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


  NEIBcell =
    (int **)allocate_matrix(sizeof(int),ICELTOT,6);

  XYZ      =
    (int **)allocate_matrix(sizeof(int),ICELTOT,3);
```

ICELTOT
要素数（NX×NY×NZ）
N
節点数

NX,NY,NZ
x,y,z方向要素数
NXP1,NYP1,NZP1
x,y,z方向節点数
IBNODTOT
NXP1×NYP1

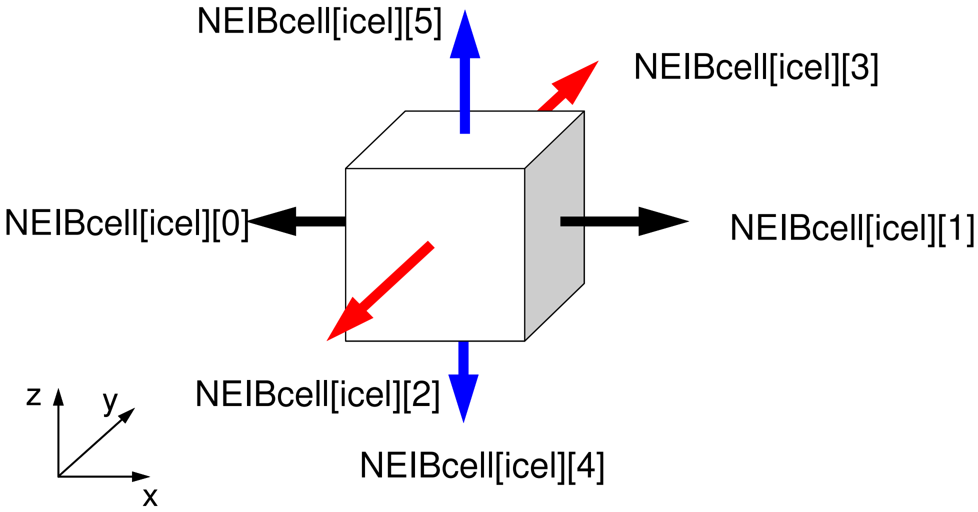
XYZ[ICELTOT][3]
要素座標

NEIBcell[ICELTOT][6]
隣接要素

allocate/deallocate

pointer_init(2/3):「mesh.dat」の作成

```
for(k=0; k<NZ; k++) {
    for(j=0; j<NY; j++) {
        for(i=0; i<NX; i++) {
            icel = k * NX * NY + j * NX + i;
            NEIBcell[icel][0] = icel - 1       + 1;
            NEIBcell[icel][1] = icel + 1       + 1;
            NEIBcell[icel][2] = icel - NX      + 1;
            NEIBcell[icel][3] = icel + NX      + 1;
            NEIBcell[icel][4] = icel - NX * NY + 1;
            NEIBcell[icel][5] = icel + NX * NY + 1;
            if(i == 0)    NEIBcell[icel][0] = 0;
            if(i == NX-1) NEIBcell[icel][1] = 0;
            if(j == 0)    NEIBcell[icel][2] = 0;
            if(j == NY-1) NEIBcell[icel][3] = 0;
            if(k == 0)    NEIBcell[icel][4] = 0;
            if(k == NZ-1) NEIBcell[icel][5] = 0;

            XYZ[icel][0] = i + 1;
            XYZ[icel][1] = j + 1;
            XYZ[icel][2] = k + 1;
        }
    }
}
```



i= XYZ[icel][0]
j= XYZ[icel][1], k= XYZ[icel][2]
icel= k*NX*NY + j*NX + i

"icel" starts at 0

12	13	14	15
8	9	10	11
4	5	6	7
0	1	2	3

"NEIBcell" starts at 1

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

NEIBcell[icel][0]= icel – 1 + 1
NEIBcell[icel][1]= icel + 1 + 1
NEIBcell[icel][2]= icel – NX + 1
NEIBcell[icel][3]= icel + NX + 1
NEIBcell[icel][4]= icel – NX*NY + 1
NEIBcell[icel][5]= icel + NX*NY + 1

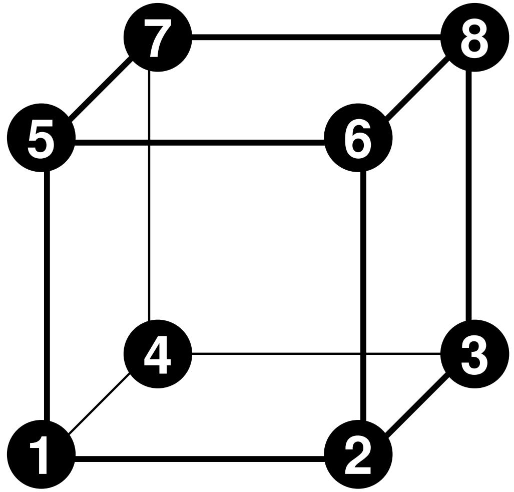
pointer_init (3/3): “mesh.dat”

```
        if(DX <= 0.0) {
                DX = 1.0 / (double)NX;
                DY = 1.0 / (double)NY;
                DZ = 1.0 / (double)NZ;
        }

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

        IBNODTOT = NXP1 * NYP1;
        N        = NXP1 * NYP1 * NZP1;

        return 0;
}
```

DX=0.0となっていた場合のみ，DX, DY, DZをこのように指定

pointer_init (3/3): “mesh.dat”

```
        if(DX <= 0.0) {
                DX = 1.0 / (double)NX;
                DY = 1.0 / (double)NY;
                DZ = 1.0 / (double)NZ;
        }

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

        IBNODTOT = NXP1 * NYP1;
        N        = NXP1 * NYP1 * NZP1;

        return 0;
}
```



NXP1, NYP1, NZP1 :
　x, y, z方向節点数

IBNODTOT :
　NXP1×NYP1

N :
　節点数（可視化に使用）

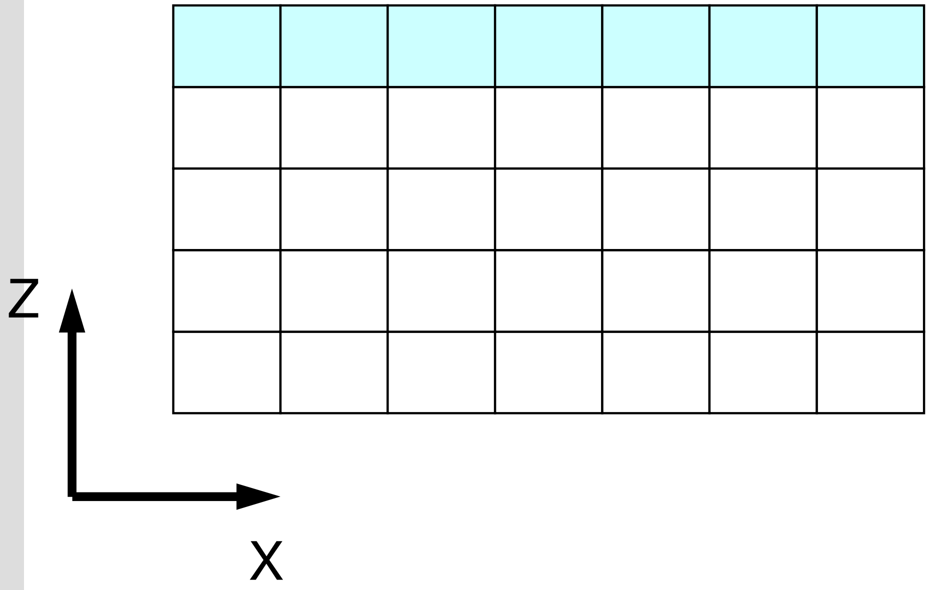
boundary_cell

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

#include "struct_ext.h"
#include "boundary_cell.h"
#include "allocate.h"

extern int
BOUNDARY_CELL(void)
{
 int IFACTOT;
 int icou, icel, i, j, k;

 IFACTOT = NX * NY;
 ZmaxCELtot = IFACTOT;

 ZmaxCEL =
    (int *)allocate_vector(sizeof(int),ZmaxCELtot);

 icou = 0;
 k    = NZ - 1;

 for(j=0; j<NY; j++) {
     for(i=0; i<NX; i++) {
         icel = k*IFACTOT + j*NX + i+1;
         ZmaxCEL[icou] = icel;
         icou++;
     }
 }

 return 0;
}
```

$Z=Z_{max}$の要素の定義

総数: ZmaxCELtot

要素番号: ZmaxCEL(:)

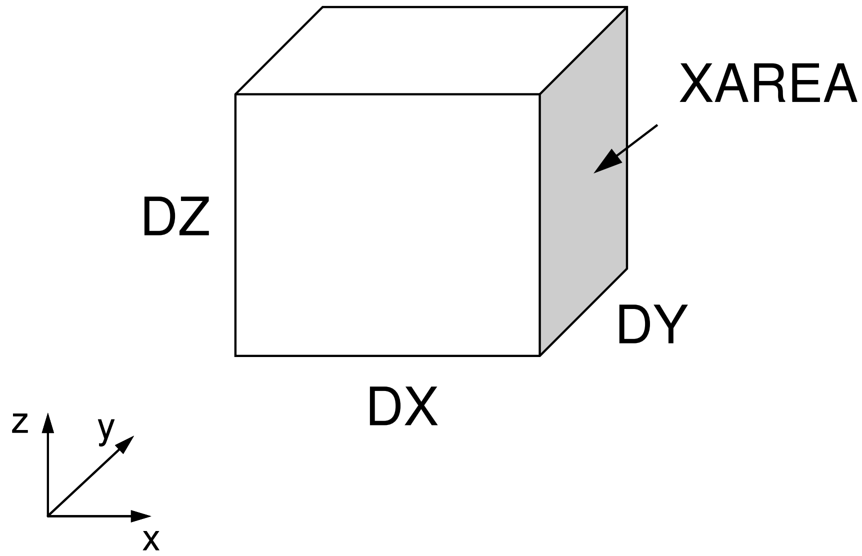



```
/***************************
  allocate vector                    allocate.c
***************************/

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;
}
```

```c
#include <stdio.h> ...

extern int
CELL_METRICS(void)
{       double V0, RV0;
        int i;
VOLCEL =
(double*)allocate_vector(sizeof(double), ICELTOT);
RVC    =
(double*)allocate_vector(sizeof(double), ICELTOT);

XAREA = DY * DZ;
YAREA = DZ * DX;
ZAREA = DX * DY;

RDX = 1.0 / DX;
RDY = 1.0 / DY;
RDZ = 1.0 / DZ;

RDX2 = 1.0 / (pow(DX, 2.0));
RDY2 = 1.0 / (pow(DY, 2.0));
RDZ2 = 1.0 / (pow(DZ, 2.0));
R2DX = 1.0 / (0.5 * DX);
R2DY = 1.0 / (0.5 * DY);
R2DZ = 1.0 / (0.5 * DZ);

V0  = DX * DY * DZ;
RV0 = 1.0 / V0;

for(i=0; i<ICELTOT; i++) {
    VOLCEL[i] = V0;
    RVC[i]    = RV0;
}
return 0; }
```

cell_metrics

計算に必要な諸パラメータ



$$XAREA = \Delta Y \times \Delta Z, \quad YAREA = \Delta Z \times \Delta X,$$

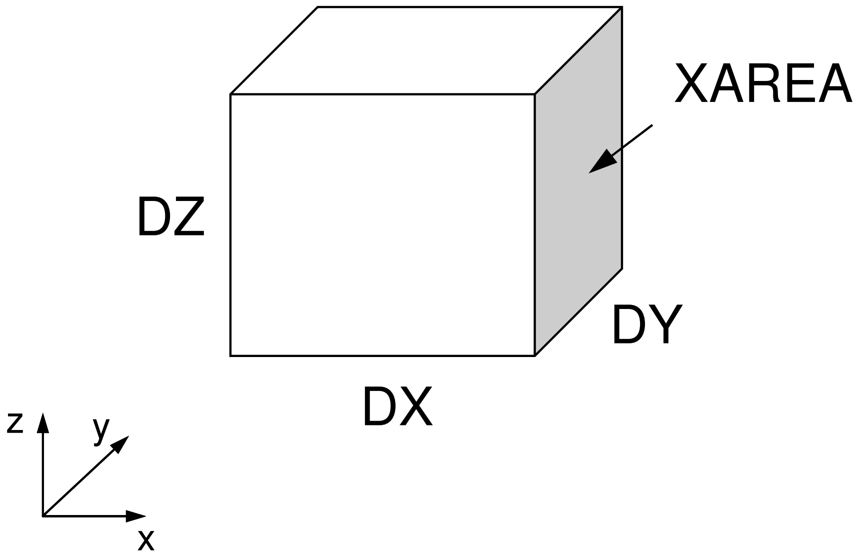
$$ZAREA = \Delta X \times \Delta Y$$

$$RDX = \frac{1}{\Delta X}, \quad RDY = \frac{1}{\Delta Y}, \quad RDZ = \frac{1}{\Delta Z}$$

cell_metrics

計算に必要な諸パラメータ

```
#include <stdio.h> ...

extern int
CELL_METRICS(void)
{       double V0, RV0;
        int i;
VOLCEL =
(double*)allocate_vector(sizeof(double),ICELTOT);
RVC    =
(double*)allocate_vector(sizeof(double),ICELTOT);

XAREA = DY * DZ;
YAREA = DZ * DX;
ZAREA = DX * DY;

RDX = 1.0 / DX;
RDY = 1.0 / DY;
RDZ = 1.0 / DZ;

RDX2 = 1.0 / (pow(DX, 2.0));
RDY2 = 1.0 / (pow(DY, 2.0));
RDZ2 = 1.0 / (pow(DZ, 2.0));
R2DX = 1.0 / (0.5 * DX);
R2DY = 1.0 / (0.5 * DY);
R2DZ = 1.0 / (0.5 * DZ);

V0  = DX * DY * DZ;
RV0 = 1.0 / V0;

for(i=0; i<ICELTOT; i++) {
    VOLCEL[i] = V0;
    RVC[i]    = RV0;
}
return 0; }
```



$$RDX2 = \frac{1}{\Delta X^2}, \quad RDY2 = \frac{1}{\Delta Y^2}, \quad RDZ2 = \frac{1}{\Delta Z^2}$$

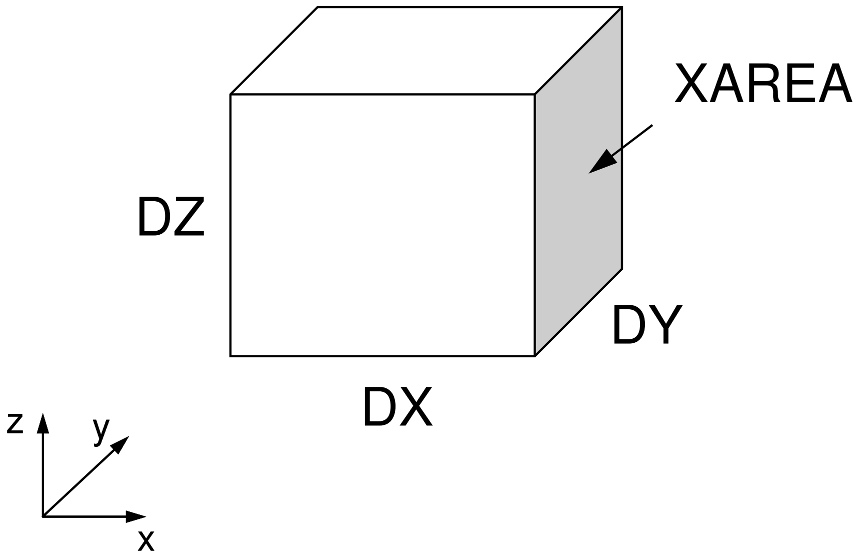
$$R2DX = \frac{1}{0.5 \times \Delta X}, \quad R2DY = \frac{1}{0.5 \times \Delta Y},$$

$$R2DZ = \frac{1}{0.5 \times \Delta Z}$$

cell_metrics

計算に必要な諸パラメータ

```
#include <stdio.h> ...

extern int
CELL_METRICS(void)
{       double V0, RV0;
        int i;
VOLCEL =
(double*)allocate_vector(sizeof(double),ICELTOT);
RVC    =
(double*)allocate_vector(sizeof(double),ICELTOT);

XAREA = DY * DZ;
YAREA = DZ * DX;
ZAREA = DX * DY;

RDX = 1.0 / DX;
RDY = 1.0 / DY;
RDZ = 1.0 / DZ;

RDX2 = 1.0 / (pow(DX, 2.0));
RDY2 = 1.0 / (pow(DY, 2.0));
RDZ2 = 1.0 / (pow(DZ, 2.0));
R2DX = 1.0 / (0.5 * DX);
R2DY = 1.0 / (0.5 * DY);
R2DZ = 1.0 / (0.5 * DZ);

V0  = DX * DY * DZ;
RV0 = 1.0 / V0;

for(i=0; i<ICELTOT; i++) {
    VOLCEL[i] = V0;
    RVC[i]    = RV0;
}
return 0; }
```

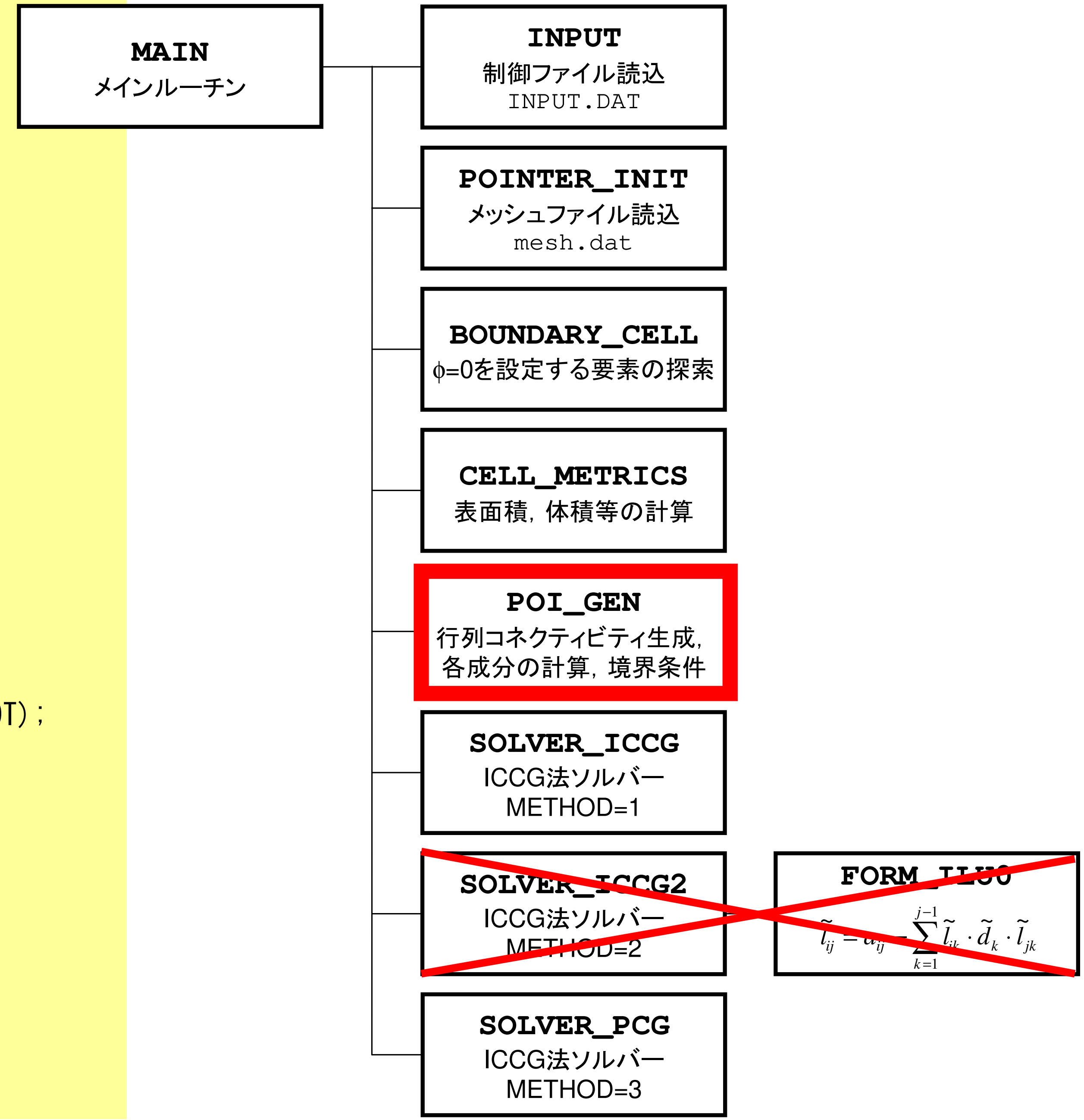


$$VOLCEL = V0 = \Delta X \times \Delta Y \times \Delta Z$$

$$RV0 = RVC = \frac{1}{VOLCEL}$$

プログラムの構成

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

#include "struct.h"
#include "pcg.h"
#include "input.h " ...

int
main()
{
  double *WK;
  int NPL, NPU; ISET, ITR, IER; icel, ic0, i;
  double xN, xL, xU; Stime, Etime;

  if(INPUT()) goto error;
  if(POINTER_INIT()) goto error;
  if(BOUNDARY_CELL()) goto error;
  if(CELL_METRICS()) goto error;
  if(POI_GEN()) goto error;

  memset(PHI, 0.0, sizeof(double)*ICELTOT);
  ISET = 0;
  WK = (double *)malloc(sizeof(double)*ICELTOT);

  if(METHOD==1){
     if(solve_ICCG(...)) goto error;
     } else if(METHOD==3){
     if(solve_PCG(...)) goto error;}

  if(OUTUCD()) goto error;
  return 0;
error:
  return -1;
}
```



poi_gen (1/7)

```
#include "allocate.h"
extern int
POI_GEN(void)
{ int nn;
  int ic0, icN1, icN2, icN3, icN4, icN5, icN6;
  int i, j, k, ib, ic, ip, icel, icou, icol, icouG;
  int ii, jj, kk, nn1, num, nr, j0, j1;
  double coef, VOL0, S1t, E1t;
  int isL, ieL, isU, ieU;
  NL=6; NU= 6;
  IAL = (int
**)allocate_matrix(sizeof(int),ICELTOT,NL);
  IAU = (int
**)allocate_matrix(sizeof(int),ICELTOT,NU);
  BFORCE = (double
*)allocate_vector(sizeof(double),ICELTOT);
  D      = (double
*)allocate_vector(sizeof(double),ICELTOT);
  PHI    = (double
*)allocate_vector(sizeof(double),ICELTOT);
  INL = (int *)allocate_vector(sizeof(int),ICELTOT);
  INU = (int *)allocate_vector(sizeof(int),ICELTOT);

  for (i = 0; i <ICELTOT ; i++) {
       BFORCE[i]=0.0;
       D[i]  =0.0; PHI[i]=0.0;
       INL[i] = 0; INU[i] = 0;
       for(j=0;j<6;j++){
           IAL[i][j]=0; IAU[i][j]=0;
       }
  }
  for (i = 0; i <=ICELTOT ; i++) {
       indexL[i] = 0; indexU[i] = 0;
  }
```

allocate.c

```
/*************************
  allocate matrix
*************************/
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;
}
```

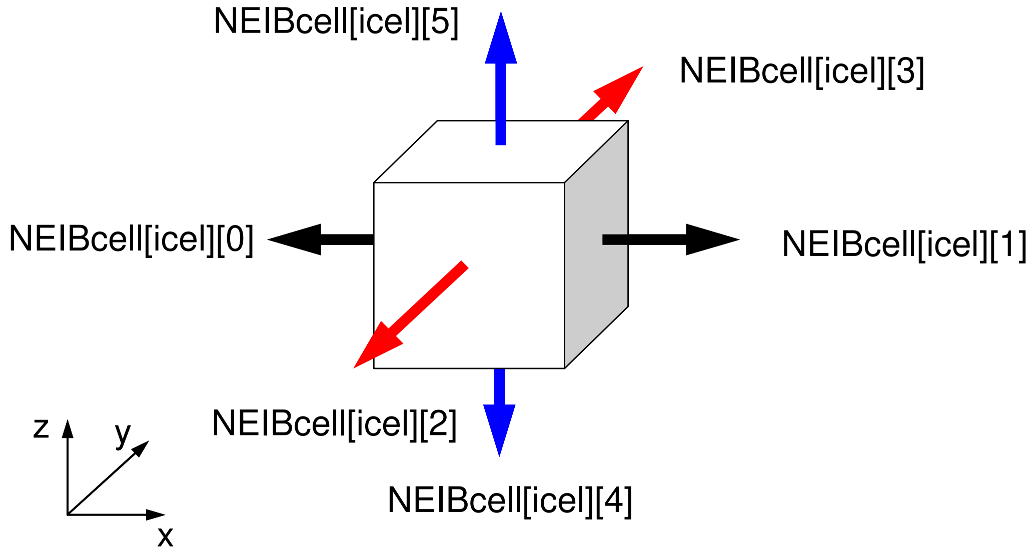
配列の宣言

配列・変数名	型	内 容
D[N]	**R**	対角成分,(N:全メッシュ数=ICELTOT)
BFORCE[N]	**R**	右辺ベクトル
PHI[N]	**R**	未知数ベクトル
indexL[N+1]	**I**	各行の非零下三角成分数(CRS)
indexU[N+1]	**I**	各行の非零上三角成分数(CRS)
NPL	**I**	非零下三角成分総数(CRS)
NPU	**I**	非零上三角成分総数(CRS)
itemL[NPL]	**I**	非零下三角成分(列番号)(CRS)
itemU[NPU]	**I**	非零下三角成分(列番号)(CRS)
AL[NPL]	**R**	非零下三角成分(係数)(CRS)
AU[NPU]	**R**	非零上三角成分(係数)(CRS)

配列・変数名	型	内 容
NL,NU	**I**	各行の非零上下三角成分の最大数(ここでは6)
INL[N]	**I**	各行の非零下三角成分数
INU[N]	**I**	各行の非零上三角成分数
IAL[N][NL]	**I**	各行の非零下三角成分に対応する列番号
IAU[N][NU]	**I**	各行の非零上三角成分に対応する列番号

poi_gen (2/7)

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

    icN1 = NEIBcell[icel][0];
    icN2 = NEIBcell[icel][1];
    icN3 = NEIBcell[icel][2];
    icN4 = NEIBcell[icel][3];
    icN5 = NEIBcell[icel][4];
    icN6 = NEIBcell[icel][5];

    if(icN5 != 0) {
       icou = INL[icel] + 1;
       IAL[icel][icou-1] = icN5;
       INL[icel]         = icou;
     }

     if(icN3 != 0) {
       icou = INL[icel] + 1;
       IAL[icel][icou-1] = icN3;
       INL[icel]         = icou;
     }

     if(icN1 != 0) {
       icou = INL[icel] + 1;
       IAL[icel][icou-1] = icN3;
       INL[icel]         = icou;
     }
```



下三角成分
NEIBcell[icel][4]= icel – NX*NY + 1
NEIBcell[icel][2]= icel – NX + 1
NEIBcell[icel][0]= icel – 1 + 1

“icel” starts at 0

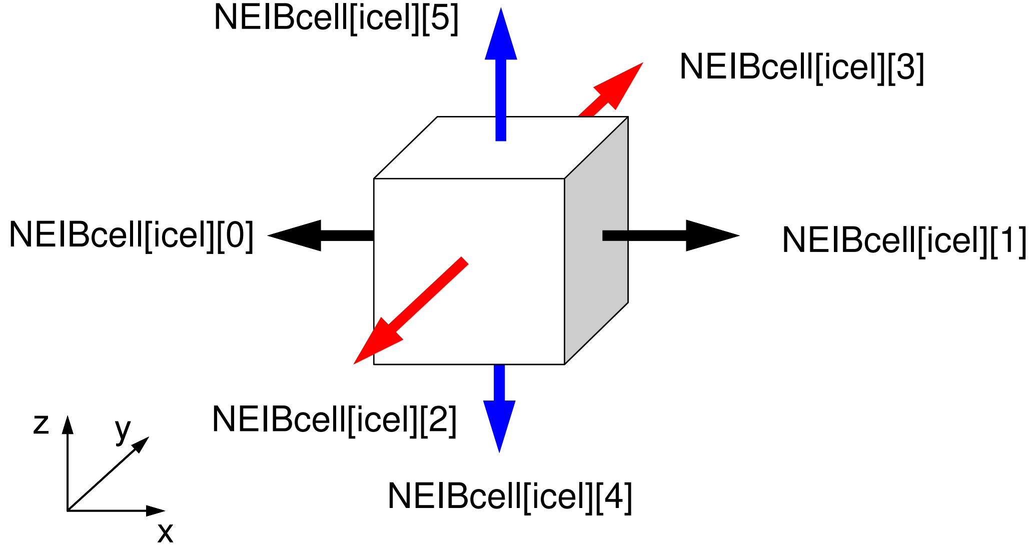
12	13	14	15
8	9	10	11
4	5	6	7
0	1	2	3

“IAL” starts at 1

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

poi_gen (3/7)

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

      icN1 = NEIBcell[icel][0];
      icN2 = NEIBcell[icel][1];
      icN3 = NEIBcell[icel][2];
      icN4 = NEIBcell[icel][3];
      icN5 = NEIBcell[icel][4];
      icN6 = NEIBcell[icel][5];
....
        if(icN2 != 0) {
          icou = INU[icel] + 1;
          IAU[icel][icou-1] = icN2;
          INU[icel]         = icou;
        }

        if(icN4 != 0) {
          icou = INU[icel] + 1;
          IAU[icel][icou-1] = icN4;
          INU[icel]         = icou;
        }

        if(icN6 != 0) {
          icou = INU[icel] + 1;
          IAU[icel][icou-1] = icN6;
          INU[icel]         = icou;
        }
 }
```



上三角成分
NEIBcell[icel][1]= icel + 1 + 1
NEIBcell[icel][3]= icel + NX + 1
NEIBcell[icel][5]= icel + NX*NY + 1

"icel" starts at 0

12	13	14	15
8	9	10	11
4	5	6	7
0	1	2	3

"IAU" starts at 1

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

poi_gen (4/7)

```
indexL =
    (int *)allocate_vector(sizeof(int),ICELTOT+1);
indexU =
    (int *)allocate_vector(sizeof(int),ICELTOT+1);

for(i=0; i<ICELTOT; i++){
     indexL[i+1]=indexL[i]+INL[i];
     indexU[i+1]=indexU[i]+INU[i];
}
NPL = indexL[ICELTOT];
NPU = indexU[ICELTOT];

itemL = (int *)allocate_vector(sizeof(int),NPL);
itemU = (int *)allocate_vector(sizeof(int),NPU);
AL =
(double *)allocate_vector(sizeof(double),NPL);
AU =
(double *)allocate_vector(sizeof(double),NPU);

memset(itemL, 0, sizeof(int)*NPL);
memset(itemU, 0, sizeof(int)*NPU);
memset(AL, 0.0, sizeof(double)*NPL);
memset(AU, 0.0, sizeof(double)*NPU);

for(i=0; i<ICELTOT; i++){
     for(k=0;k<INL[i];k++){
          kk= k + indexL[i];
          itemL[kk]= IAL[i][k];
     }
     for(k=0;k<INU[i];k++){
          kk= k + indexU[i];
          itemU[kk]= IAU[i][k];
     }
}

free(INL); free(INU);
free(IAL); free(IAU);
```

配列・変数名	型	内　容
D[N]	**R**	対角成分，(N:全メッシュ数=ICELTOT)
BFORCE[N]	**R**	右辺ベクトル
PHI[N]	**R**	未知数ベクトル
indexL[N+1]	**I**	各行の非零下三角成分数(CRS)
indexU[N+1]	**I**	各行の非零上三角成分数(CRS)
NPL	**I**	非零下三角成分総数(CRS)
NPU	**I**	非零上三角成分総数(CRS)
itemL[NPL]	**I**	非零下三角成分(列番号)(CRS)
itemU[NPU]	**I**	非零下三角成分(列番号)(CRS)
AL[NPL]	**R**	非零下三角成分(係数)(CRS)
AU[NPU]	**R**	非零上三角成分(係数)(CRS)

"itemL" /"itemU" start at 1

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

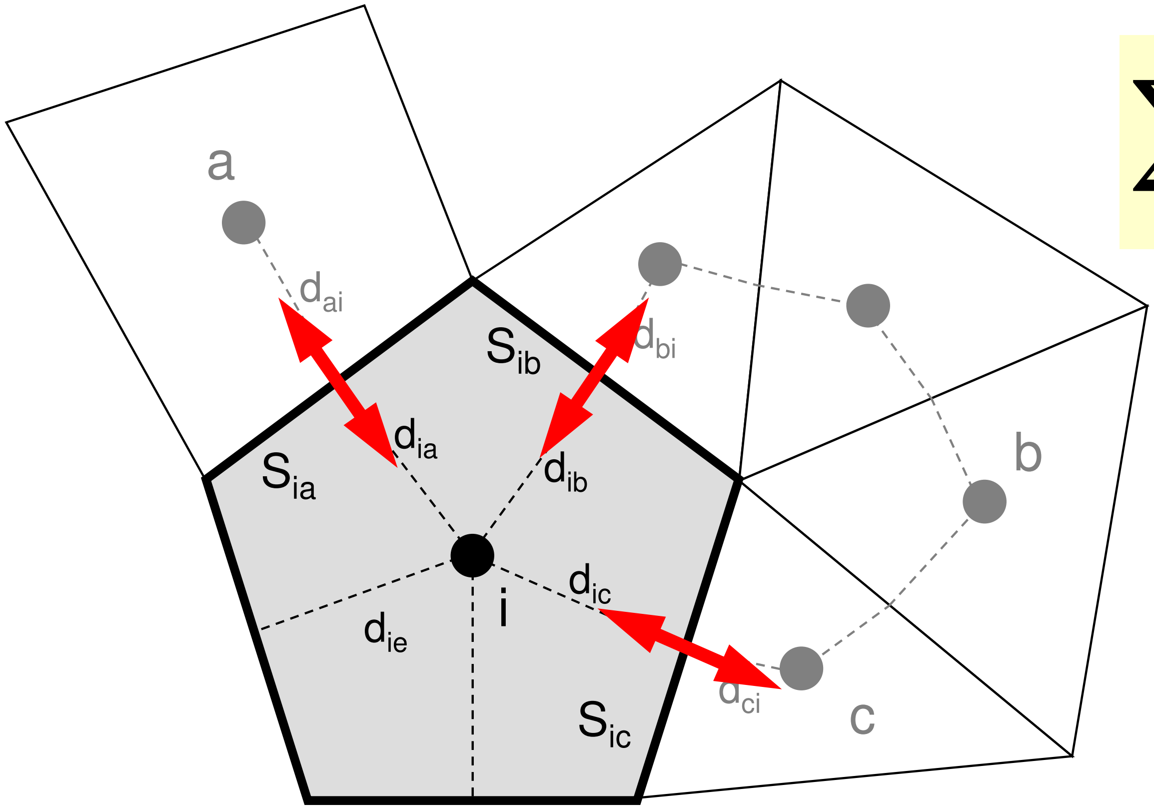
```
for (i=0; i<N; i++) {
    q[i]= D[i] * p[i];
    for (j=indexL[i]; j<indexL[i+1]; j++) {
        q[i] += AL[j] * p[itemL[j]-1];
    }
    for (j=indexU[i]; j<indexU[i+1]; j++) {
        q[i] += AU[j] * p[itemU[j]-1];
    }
}
```

ポアソン方程式：有限体積法による離散化

$$\frac{\partial^2 \phi}{\partial x^2}+\frac{\partial^2 \phi}{\partial y^2}+\frac{\partial^2 \phi}{\partial z^2}+f=0$$

Poisson Eq. by Finite Volume Method（FVM）

面を通過するフラックス（flux，流束）の保存に着目



隣接要素との拡散

$$\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\left(\phi_k-\phi_i\right)+V_i\dot{Q}_i=0$$

体積フラックス

- V_i ：要素体積
- S ：表面面積
- d_{ij} ：要素中心から表面までの距離
- Q ：体積フラックス

全体マトリクスの生成

要素iに関する釣り合い

$$\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\left(\phi_k-\phi_i\right)+V_i\dot{Q}_i=0$$

$$-\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k+\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_i=+V_i\dot{Q}_i$$

$$-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]=-V_i\dot{Q}_i$$

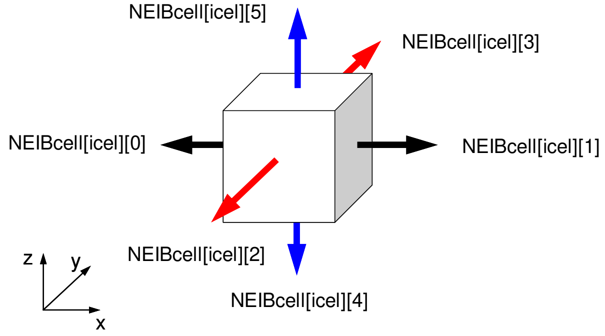
D（対角成分）

AL, AU（非対角成分）

BFORCE（右辺）

poi_gen (5/7)

```
for(icel=0; icel<ICELTOT; icel++) {
      icN1 = NEIBcell[icel][0];
      icN2 = NEIBcell[icel][1];
      icN3 = NEIBcell[icel][2];
      icN4 = NEIBcell[icel][3];
      icN5 = NEIBcell[icel][4];
      icN6 = NEIBcell[icel][5];
      VOL0 = VOLCEL[icel];
      isL =indexL[icel  ];     ieL =indexL[icel+1];
      isU =indexU[icel  ];     ieU =indexU[icel+1];

      if(icN5 != 0) {
          coef = RDZ * ZAREA;
          D[icel] -= coef;
          for(j=isL; j<ieL; j++) {
                if(itemL[j] == icN5) {
                    AL[j] = coef;
                    break;}
          }
        }
        if(icN3 != 0) {
          coef = RDY * YAREA;
          D[icel] -= coef;
          for(j=isL; j<ieL; j++) {
                if(itemL[j] == icN3) {
                    AL[j] = coef;
                    break;}
          }
        }
        if(icN1 != 0) {
          coef = RDX * XAREA;
          D[icel] -= coef;
          for(j=isL; j<ieL; j++) {
                if(itemL[j] == icN1) {
                    AL[j] = coef;
                    break;}
          }
        }
```

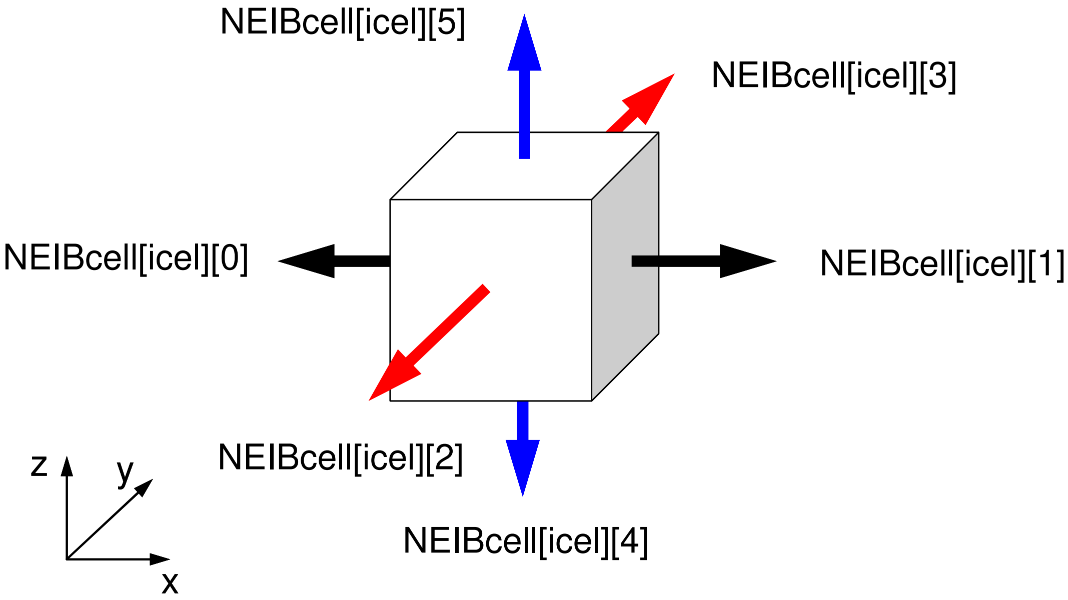


$$\frac{\phi_{neib[icel][0]}-\phi_{icel}}{\Delta x}\Delta y\Delta z+\frac{\phi_{neib[icel][1]}-\phi_{icel}}{\Delta x}\Delta y\Delta z+$$

$$\frac{\phi_{neib[icel][2]}-\phi_{icel}}{\Delta y}\Delta z\Delta x+\frac{\phi_{neib[icel][3]}-\phi_{icel}}{\Delta y}\Delta z\Delta x+$$

$$\boxed{\frac{\phi_{neib[icel][4]}-\phi_{icel}}{\Delta z}\Delta x\Delta y}+\frac{\phi_{neib[icel][5]}-\phi_{icel}}{\Delta z}\Delta x\Delta y+f_{icel}\Delta x\Delta y\Delta z=0$$

係数の計算



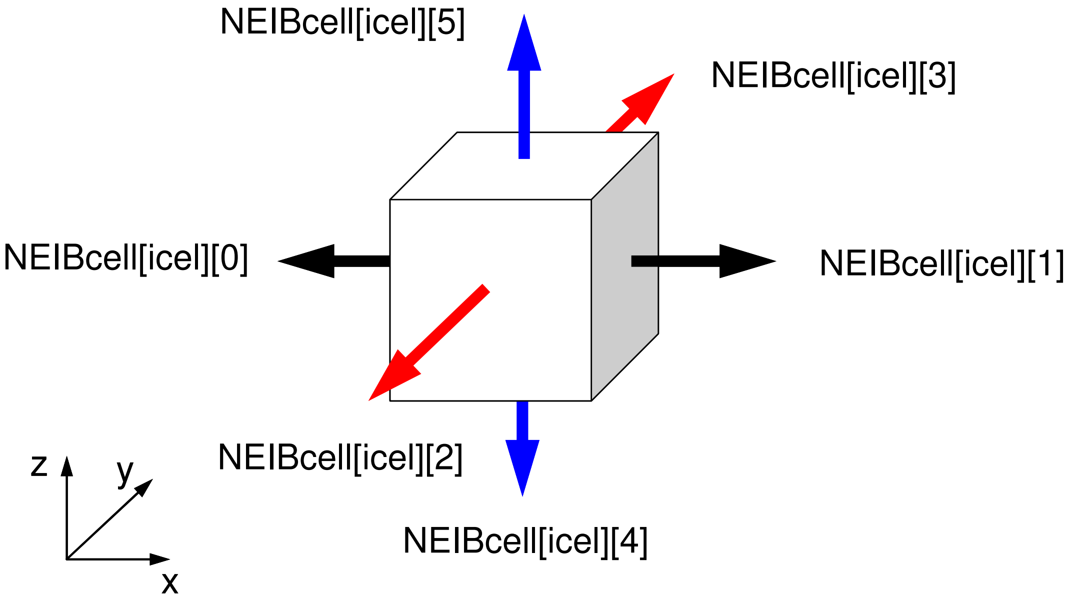

```
if(icN5 != 0) {
   coef = RDZ * ZAREA;
   D[icel] -= coef;
   for(j=isL; j<ieL; j++) {
        if(itemL[j] == icN5) {
           AL[j] = coef;
           break;
        }
     }
}
```

$$
\frac{\phi_{neib[icel][0]}-\phi_{icel}}{\Delta x}\Delta y\Delta z+\frac{\phi_{neib[icel][1]}-\phi_{icel}}{\Delta x}\Delta y\Delta z+
$$

$$
\frac{\phi_{neib[icel][2]}-\phi_{icel}}{\Delta y}\Delta z\Delta x+\frac{\phi_{neib[icel][3]}-\phi_{icel}}{\Delta y}\Delta z\Delta x+
$$

$$
\boxed{\frac{\phi_{neib[icel][4]}-\phi_{icel}}{\Delta z}\Delta x\Delta y}+\frac{\phi_{neib[icel][5]}-\phi_{icel}}{\Delta z}\Delta x\Delta y+f_{icel}\Delta x\Delta y\Delta z=0
$$

係数の計算




```
if(icN5 != 0) {
   coef = RDZ * ZAREA;
   D[icel] -= coef;
   for(j=isL; j<ieL; j++) {
        if(itemL[j] == icN5) {
          AL[j] = coef;
          break;
        }
     }
}
```

$$\frac{\Delta y \Delta z}{\Delta x}\left(\phi_{neib[icel][0]}-\phi_{icel}\right)+\frac{\Delta y \Delta z}{\Delta x}\left(\phi_{neib[icel][1]}-\phi_{icel}\right)+$$

$$\frac{\Delta z \Delta x}{\Delta y}\left(\phi_{neib[icel][2]}-\phi_{icel}\right)+\frac{\Delta z \Delta x}{\Delta y}\left(\phi_{neib[icel][3]}-\phi_{icel}\right)+$$

ZAREA

RDZ

$$\boxed{\frac{\Delta x \Delta y}{\Delta z}\left(\phi_{neib[icel][4]}-\phi_{icel}\right)}+\frac{\Delta x \Delta y}{\Delta z}\left(\phi_{neib[icel][5]}-\phi_{icel}\right)+f_{icel}\Delta x \Delta y \Delta z=0$$

poi_gen (6/7)

```
      if(icN2 != 0) {
        coef = RDX * XAREA;
        D[icel] -= coef;
        for(j=isU; j<ieU; j++) {
            if(itemU[j] == icN2) {
               AU[j] = coef;
               break;}
        }
      }

      if(icN4 != 0) {
        coef = RDY * YAREA;
        D[icel] -= coef;
        for(j=isU; j<ieU; j++) {
            if(itemU[j] == icN4) {
               AU[j] = coef;
               break;}
        }
      }

      if(icN6 != 0) {
        coef = RDZ * ZAREA;
        D[icel] -= coef;
        for(j=isU; j<ieU; j++) {
            if(itemU[j] == icN6) {
               AU[j] = coef;
               break;}
        }
      }

      ii = XYZ[icel][0];
      jj = XYZ[icel][1];
      kk = XYZ[icel][2];

      BFORCE[icel]= -(double)(ii+jj+kk) *
                          VOLCEL[icel];
}
```

poi_gen (6/7)

```
      if(icN2 != 0) {
        coef = RDX * XAREA;
        D[icel] -= coef;
        for(j=isU; j<ieU; j++) {
            if(itemU[j] == icN2) {
               AU[j] = coef;
               break;}
        }
      }

      if(icN4 != 0) {
        coef = RDY * YAREA;
        D[icel] -= coef;
        for(j=isU; j<ieU; j++) {
            if(itemU[j] == icN4) {
               AU[j] = coef;
               break;}
        }
      }

      if(icN6 != 0) {
        coef = RDZ * ZAREA;
        D[icel] -= coef;
        for(j=isU; j<ieU; j++) {
            if(itemU[j] == icN6) {
               AU[j] = coef;
               break;}
        }
      }

      ii = XYZ[icel][0];
      jj = XYZ[icel][1];
      kk = XYZ[icel][2];

      BFORCE[icel]= -(double)(ii+jj+kk) *
                              VOLCEL[icel];
}
```

Volume Flux

$$f = dfloat(i_0 + j_0 + k_0)$$

$$i_0 = XYZ[icel][0],$$

$$j_0 = XYZ[icel][1],$$

$$k_0 = XYZ[icel][2]$$

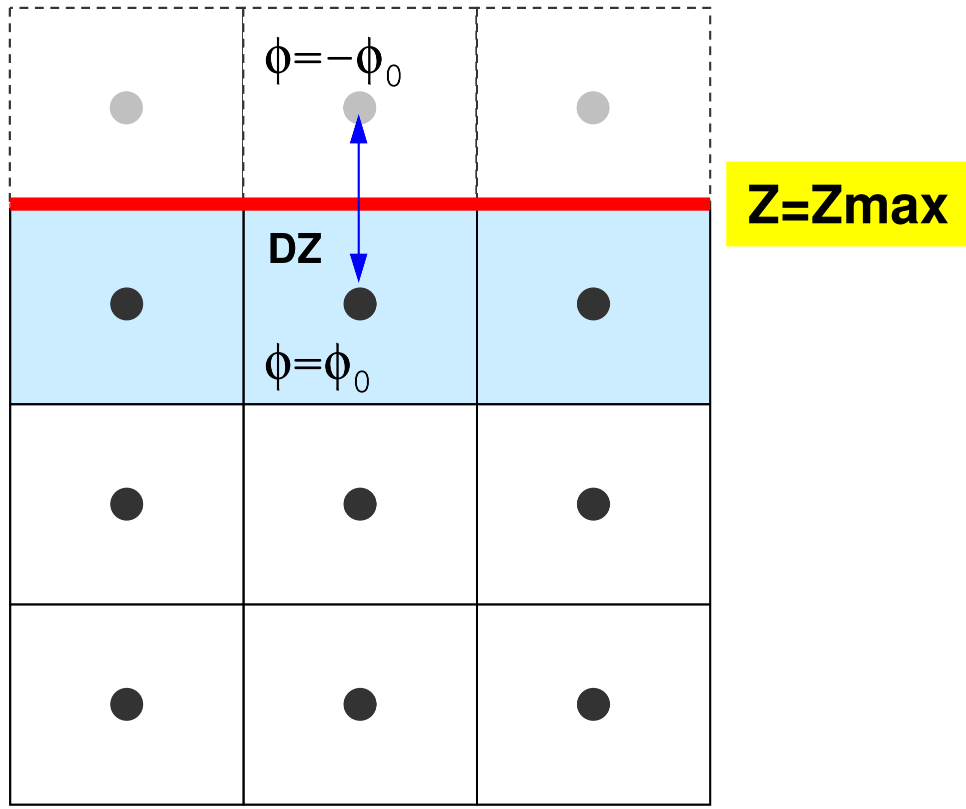
XYZ[icel][k](k=0,1,2)はX，Y，Z方向の差分格子のインデックス

各メッシュがX，Y，Z方向の何番目にあるかを示している。

poi_gen (7/7)

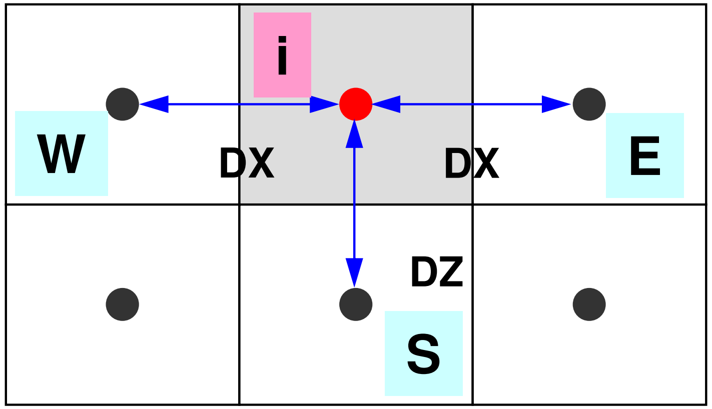
```
/* TOP SURFACE */

        for(ib=0; ib<ZmaxCELtot; ib++) {
            icel   = ZmaxCEL[ib];
            coef = 2.0 * RDZ * ZAREA;
            D[icel-1] -= coef;
        }


        return 0;
}
```

係数の計算（境界面）



境界面の外側に，大きさが同じで，値が $\phi=-\phi_0$ となるような要素があると仮定（境界面で丁度 $\phi=0$ となる）：一次近似

ディリクレ条件



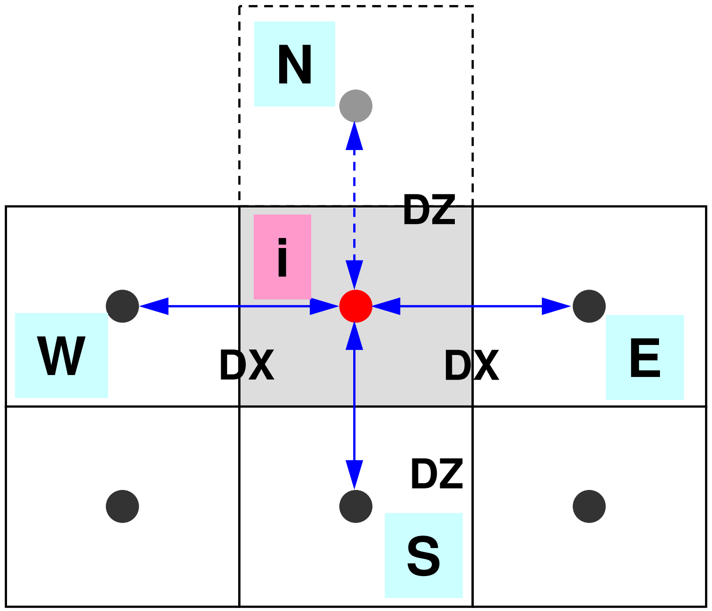
$$-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i + \left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right] = -V_i\dot{Q}_i$$

D（対角成分）

AL, AU（非対角成分）

BFORCE（右辺）

Dirichlet B.C.



$$\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}(\phi_k-\phi_i)+V_i\dot{Q}_i=0$$

$$-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]=-V_i\dot{Q}_i$$

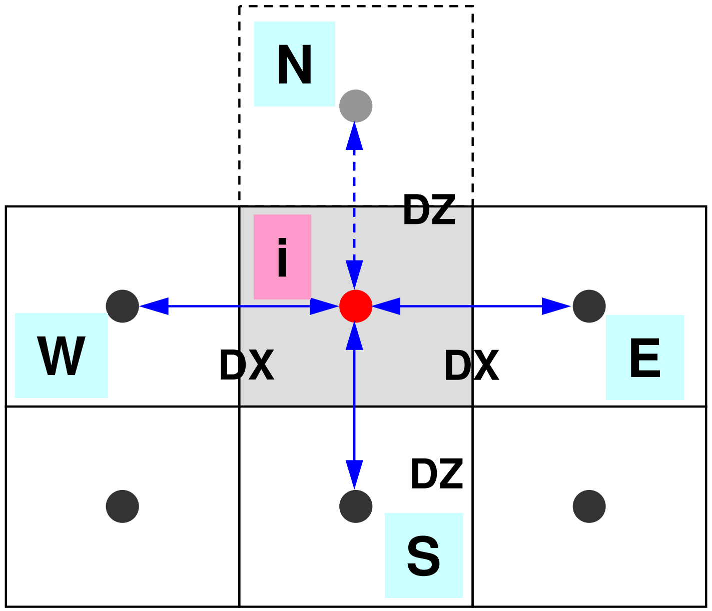
D (diagonal)

AL, AU (off-diag.)

BFORCE (RHS)

$$-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]+\frac{\phi_N-\phi_i}{\Delta z}\Delta x\Delta y=-V_i\dot{Q}_i, \quad \phi_N=-\phi_i$$

Dirichlet B.C.



$$\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}(\phi_k-\phi_i)+V_i\dot{Q}_i=0$$

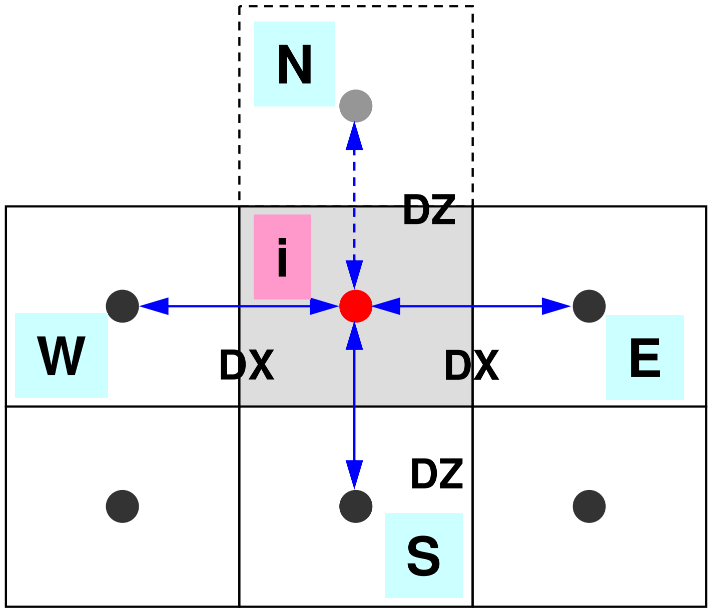
$$-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]=-V_i\dot{Q}_i$$

D (diagonal) **AL, AU (off-diag.)** **BFORCE (RHS)**

$$-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]+\frac{\phi_N-\phi_i}{\Delta z}\Delta x\Delta y=-V_i\dot{Q}_i, \quad \phi_N=-\phi_i$$

$$-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]+\frac{-\phi_i-\phi_i}{\Delta z}\Delta x\Delta y=-V_i\dot{Q}_i$$

Dirichlet B.C.



$$\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}(\phi_k-\phi_i)+V_i\dot{Q}_i=0$$

$$-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]=-V_i\dot{Q}_i$$

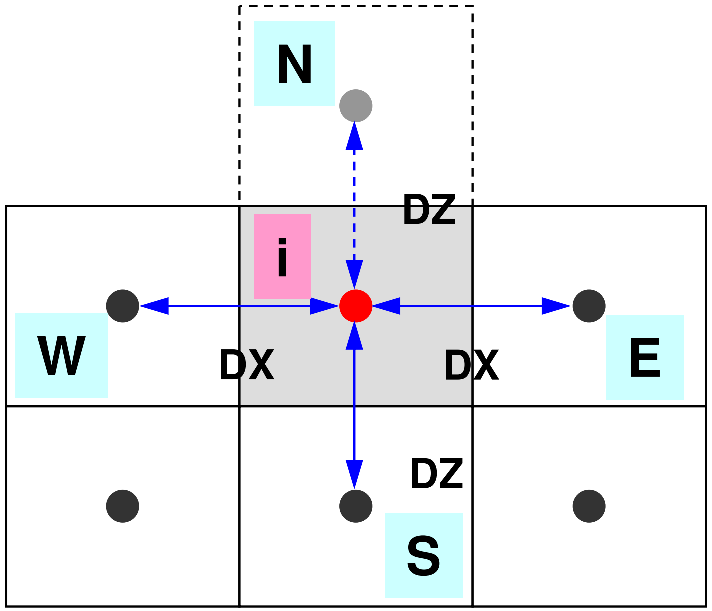
D (diagonal)

AL, AU (off-diag.)

BFORCE (RHS)

$$-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]+\frac{-2\phi_i}{\Delta z}\Delta x\Delta y=+V_i\dot{Q}_i$$

Dirichlet B.C.



$$\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}(\phi_k-\phi_i)+V_i\dot{Q}_i=0$$

$$-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]=-V_i\dot{Q}_i$$

D (diagonal)

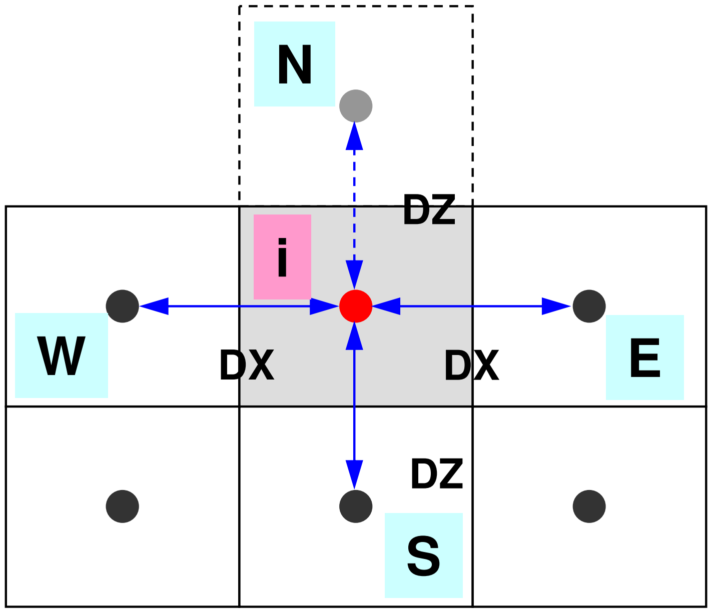
AL, AU (off-diag.)

BFORCE (RHS)

$$-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]+\frac{-2\phi_i}{\Delta z}\Delta x\Delta y=+V_i\dot{Q}_i$$

$$\left[-\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}-\frac{2}{\Delta z}\Delta x\Delta y\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]+=-V_i\dot{Q}_i$$

Dirichlet B.C.



$$\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}(\phi_k-\phi_i)+V_i\dot{Q}_i=0$$

$$-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]=-V_i\dot{Q}_i$$

D (diagonal)

AL, AU (off-diag.)

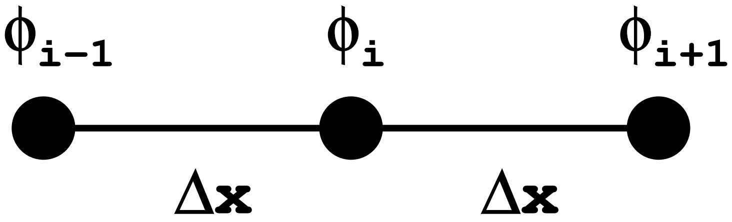
BFORCE (RHS)

$$-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right.$$

```
for(ib=0; ib<ZmaxCELtot; ib++) {
      icel  = ZmaxCEL[ib];
      coef = 2.0 * RDZ * ZAREA;
      D[icel-1] -= coef;
}
```

$$\left[-\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}-\frac{2}{\Delta z}\Delta x\Delta y\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]+=-V_i\dot{Q}_i$$

Taylor Series Expansion



$$\phi_{i+1} = \phi_i + \Delta x \left(\frac{\partial \phi}{\partial x}\right)_i + \frac{(\Delta x)^2}{2!}\left(\frac{\partial^2 \phi}{\partial x^2}\right)_i + \frac{(\Delta x)^3}{3!}\left(\frac{\partial^3 \phi}{\partial x^3}\right)_i \cdots$$

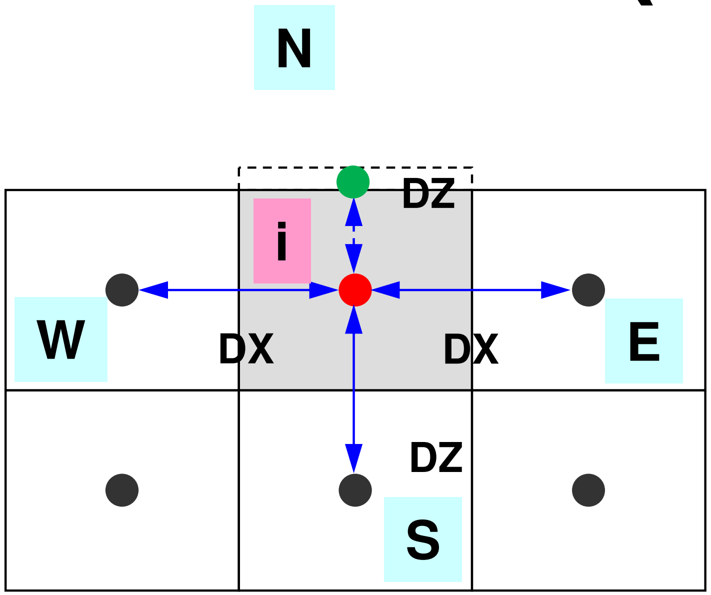
$$\phi_{i-1} = \phi_i - \Delta x \left(\frac{\partial \phi}{\partial x}\right)_i + \frac{(\Delta x)^2}{2!}\left(\frac{\partial^2 \phi}{\partial x^2}\right)_i - \frac{(\Delta x)^3}{3!}\left(\frac{\partial^3 \phi}{\partial x^3}\right)_i \cdots$$

$$\phi_{i-1} + \phi_{i+1} = 2\phi_i + 2 \times \frac{(\Delta x)^2}{2!}\left(\frac{\partial^2 \phi}{\partial x^2}\right)_i + 2 \times \frac{(\Delta x)^4}{4!}\left(\frac{\partial^4 \phi}{\partial x^4}\right)_i \cdots$$

$$\frac{\phi_{i-1} - 2\phi_i + \phi_{i+1}}{(\Delta x)^2} = \left(\frac{\partial^2 \phi}{\partial x^2}\right)_i + \boxed{\frac{(\Delta x)^2}{12}\left(\frac{\partial^4 \phi}{\partial x^4}\right)_i} \cdots$$

Truncation Err.: 2nd Order
2nd Order Accuracy
If Δx is not uniform: 1st or Lower Order Accuracy

Dirichlet B.C. “N” is very thin (=ε) 1st order (or lower) Accuracy



$$\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}(\phi_k-\phi_i)+V_i\dot{Q}_i=0$$

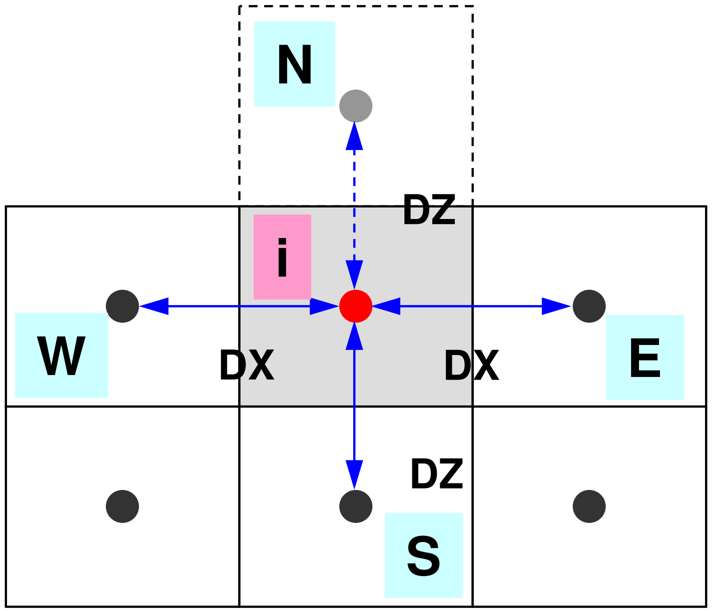
$$-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]=-V_i\dot{Q}_i$$

D (diagonal) | **AL, AU (off-diag.)** | **BFORCE (RHS)**

$$-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]+\frac{\phi_N-\phi_i}{\left(\frac{\Delta z}{2}+\frac{\varepsilon}{2}\right)}\Delta x\Delta y=-V_i\dot{Q}_i, \quad \phi_N=0, \quad \varepsilon\sim 0$$

$$\boxed{-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]-\frac{2\phi_i}{\Delta z}\Delta x\Delta y=-V_i\dot{Q}_i}$$

Dirichlet B.C. using Mirror Image



$$\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}(\phi_k-\phi_i)+V_i\dot{Q}_i=0$$

$$-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]=-V_i\dot{Q}_i$$

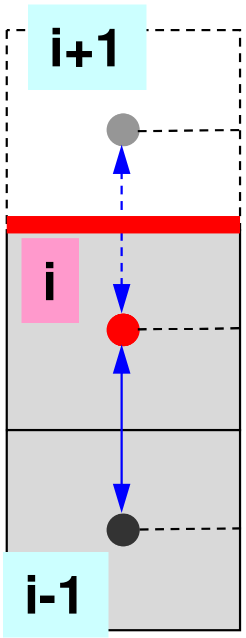
D (diagonal) **AL, AU (off-diag.)** **BFORCE (RHS)**

$$\boxed{-\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]+\frac{-2\phi_i}{\Delta z}\Delta x\Delta y=+V_i\dot{Q}_i}$$

$$\left[-\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}-\frac{2}{\Delta z}\Delta x\Delta y\right]\phi_i+\left[\sum_k \frac{S_{ik}}{d_{ik}+d_{ki}}\phi_k\right]+=-V_i\dot{Q}_i$$

Higher Order Approximation for Dirichlet B.C. in 1D Problem

more complicated in 2D/3D cases



$$z = +\frac{\Delta z}{2} = +l$$

$$\phi = 0 \ @ \ z = 0$$

$$z = -\frac{\Delta z}{2} = -l$$

$$z = -\frac{3}{2}\Delta z = -3l$$

$$\phi = az^2 + bz + c$$

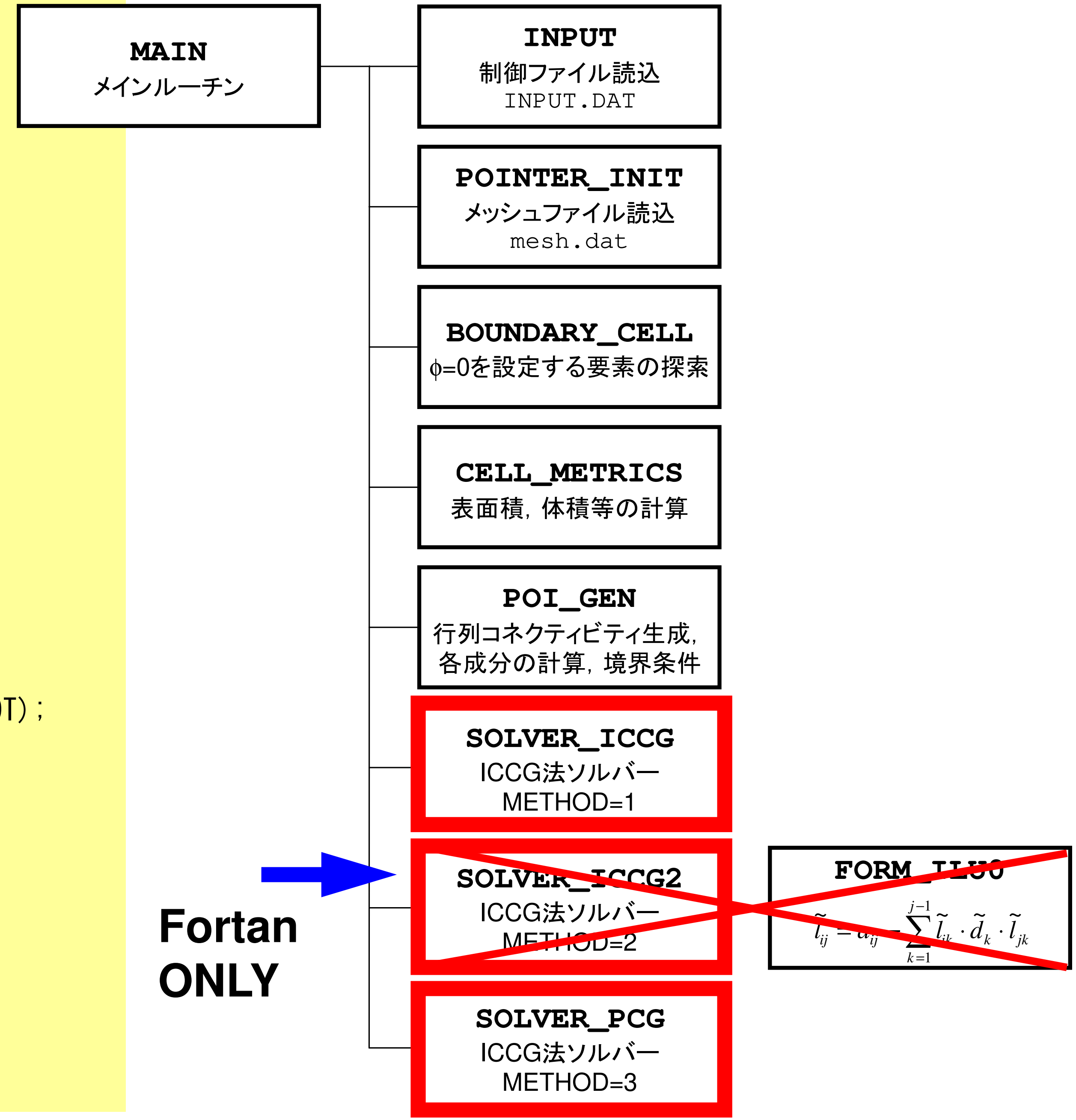
$$\phi(z=0) = c = 0$$

$$\phi_i = al^2 - bl + c = al^2 - bl, \quad \phi_{i-1} = 9al^2 - 3bl + c = 9al^2 - 3bl$$

$$a = \frac{\phi_{i-1} - 3\phi_i}{6l^2}, \quad b = \frac{\phi_{i-1} - 9\phi_i}{6l} \Rightarrow \boxed{\phi_{i+1} = al^2 + bl = \frac{1}{3}\phi_{i-1} - 2\phi_i}$$

Structure of the Program

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

#include "struct.h"
#include "pcg.h"
#include "input.h " ...

int
main()
{
  double *WK;
  int NPL, NPU; ISET, ITR, IER; icel, ic0, i;
  double xN, xL, xU; Stime, Etime;

  if(INPUT()) goto error;
  if(POINTER_INIT()) goto error;
  if(BOUNDARY_CELL()) goto error;
  if(CELL_METRICS()) goto error;
  if(POI_GEN()) goto error;

  memset(PHI, 0.0, sizeof(double)*ICELTOT);
  ISET = 0;
  WK = (double *)malloc(sizeof(double)*ICELTOT);

  if(METHOD==1){
     if(solve_ICCG(...)) goto error;
     } else if(METHOD==3){
     if(solve_PCG(...)) goto error;}

  if(OUTUCD()) goto error;
  return 0;
error:
  return -1;
}
```



- 背景
  - 有限体積法
  - 前処理付反復法
- **ICCG法によるポアソン方程式法ソルバーについて**
  - 実行方法
    - データ構造
  - **プログラムの説明**
    - 初期化
    - 係数マトリクス生成
    - **ICCG法**

あとは線形方程式を解けば良い

- 共役勾配法（Conjugate Gradient, CG）
- 前処理
  - 不完全コレスキー分解（Incomplete Cholesky Factorization, IC）
  - 実は不完全「修正」コレスキー分解
- ICCG

修正コレスキー分解

- 対称行列AのLU分解
- 対称行列Aは，対角行列Dを利用して，$[A] = [L][D][L]^T$ のような形に分解することができる。
  - この分解をLDL^T分解または修正コレスキー分解（modified Cholesky decomposition）と呼ぶ。
  - $[A] = [L][L]^T$とするような分解法もある（コレスキー分解）

N=5の場合の例

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} & 0 \\ l_{51} & l_{52} & l_{53} & l_{54} & l_{55} \end{bmatrix} \begin{bmatrix} d_1 & 0 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 & 0 \\ 0 & 0 & d_3 & 0 & 0 \\ 0 & 0 & 0 & d_4 & 0 \\ 0 & 0 & 0 & 0 & d_5 \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} & l_{41} & l_{51} \\ 0 & l_{22} & l_{32} & l_{42} & l_{52} \\ 0 & 0 & l_{33} & l_{43} & l_{53} \\ 0 & 0 & 0 & l_{44} & l_{54} \\ 0 & 0 & 0 & 0 & l_{55} \end{bmatrix}$$

修正コレスキー分解

$$\sum_{k=1}^{j} l_{ik} \cdot d_k \cdot l_{jk} = a_{ij} \quad (j = 1,2,\cdots,i-1)$$

$$\sum_{k=1}^{i} l_{ik} \cdot d_k \cdot l_{ik} = a_{ii}$$

ここで $l_{ii} \cdot d_i = 1$ とすると以下が導かれる

$$l_{ii} \cdot d_i = 1$$

$$l_{ij} \cdot d_j \cdot l_{jj} + \sum_{k=1}^{j-1} l_{ik} \cdot d_k \cdot l_{jk}$$

$$= l_{ij} + \sum_{k=1}^{j-1} l_{ik} \cdot d_k \cdot l_{jk} = a_{ij}$$

$$i = 1,2,\cdots,n$$

$$j = 1,2,\cdots,i-1$$

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} \cdot d_k \cdot l_{jk}$$

$$d_i = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}{}^2 \cdot d_k \right)^{-1} = l_{ii}{}^{-1}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} & 0 \\ l_{51} & l_{52} & l_{53} & l_{54} & l_{55} \end{bmatrix} \begin{bmatrix} d_1 & 0 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 & 0 \\ 0 & 0 & d_3 & 0 & 0 \\ 0 & 0 & 0 & d_4 & 0 \\ 0 & 0 & 0 & 0 & d_5 \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} & l_{41} & l_{51} \\ 0 & l_{22} & l_{32} & l_{42} & l_{52} \\ 0 & 0 & l_{33} & l_{43} & l_{53} \\ 0 & 0 & 0 & l_{44} & l_{54} \\ 0 & 0 & 0 & 0 & l_{55} \end{bmatrix}$$

$$= \begin{bmatrix} l_{11} & 0 & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} & 0 \\ l_{51} & l_{52} & l_{53} & l_{54} & l_{55} \end{bmatrix} \begin{bmatrix} d_1 \cdot l_{11} & d_1 \cdot l_{21} & d_1 \cdot l_{31} & d_1 \cdot l_{41} & d_1 \cdot l_{51} \\ 0 & d_2 \cdot l_{22} & d_2 \cdot l_{32} & d_2 \cdot l_{42} & d_2 \cdot l_{52} \\ 0 & 0 & d_3 \cdot l_{33} & d_3 \cdot l_{43} & d_3 \cdot l_{53} \\ 0 & 0 & 0 & d_4 \cdot l_{44} & d_4 \cdot l_{54} \\ 0 & 0 & 0 & 0 & d_5 \cdot l_{55} \end{bmatrix}$$

$$= \begin{bmatrix} l_{11} \cdot d_1 \cdot l_{11} & l_{11} \cdot d_1 \cdot l_{21} & l_{11} \cdot d_1 \cdot l_{31} & l_{11} \cdot d_1 \cdot l_{41} & l_{11} \cdot d_1 \cdot l_{51} \\ l_{21} \cdot d_1 \cdot l_{11} & l_{21} \cdot d_1 \cdot l_{21} + l_{22} \cdot d_2 \cdot l_{22} & l_{21} \cdot d_1 \cdot l_{31} + l_{22} \cdot d_2 \cdot l_{32} & l_{21} \cdot d_1 \cdot l_{41} + l_{22} \cdot d_2 \cdot l_{42} & l_{21} \cdot d_1 \cdot l_{51} + l_{22} \cdot d_2 \cdot l_{52} \\ l_{31} \cdot d_1 \cdot l_{11} & l_{31} \cdot d_1 \cdot l_{21} + l_{32} \cdot d_2 \cdot l_{22} & l_{31} \cdot d_1 \cdot l_{31} + l_{32} \cdot d_2 \cdot l_{32} + l_{33} \cdot d_3 \cdot l_{33} & l_{31} \cdot d_1 \cdot l_{41} + l_{32} \cdot d_2 \cdot l_{42} + l_{33} \cdot d_3 \cdot l_{43} & l_{31} \cdot d_1 \cdot l_{51} + l_{32} \cdot d_2 \cdot l_{52} + l_{33} \cdot d_3 \cdot l_{53} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix}$$

$$\sum_{k=1}^{j} l_{ik} \cdot d_k \cdot l_{jk} = a_{ij} \quad (j = 1, 2, \cdots, i-1)$$

$$\sum_{k=1}^{i} l_{ik} \cdot d_k \cdot l_{ik} = a_{ii}$$

$$a_{ij} - \sum_{k=1}^{j-1} l_{ik} \cdot d_k \cdot l_{jk}$$

$$= l_{ij} \cdot \underline{d_j \cdot l_{jj}} = l_{ij}$$

不完全修正コレスキー分解

$$\sum_{k=1}^{j} l_{ik} \cdot d_k \cdot l_{jk} = a_{ij} \quad (j = 1,2,\cdots,i-1)$$

$$\sum_{k=1}^{i} l_{ik} \cdot d_k \cdot l_{ik} = a_{ii}$$

ここで $l_{ii} \cdot d_i = 1$ とすると以下が導かれる

$$i = 1,2,\cdots,n$$

$$j = 1,2,\cdots,i-1$$

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} \cdot d_k \cdot l_{jk} \rightarrow \boxed{l_{ij} = a_{ij}}$$

$$d_i = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}{}^2 \cdot d_k \right)^{-1} = l_{ii}{}^{-1}$$

実際には，「不完全」な分解を実施し，このような形を用いることが多い

プログラムの実行

制御データ「<$P-L1>/run/INPUT.DAT」の作成

```
32 32 32                           NX/NY/NZ
1                                  MEHOD 1:2:3
1.00e-00 1.00e-00 1.00e-00         DX/DY/DZ
0.10      1.0e-08                  OMEGA, EPSICCG
```

- METHOD
  - 前処理行列の作成方法：不完全修正コレスキー分解
    - METHOD=1　対角成分のみ変更
    - METHOD=2　非対角成分変更（Fill-inは無し：$a_{ij}\neq 0$の場合のみ）
    - METHOD=3　対角スケーリング（点ヤコビ）

$$
\begin{cases}
i = 1,2,\cdots,n \\
\begin{cases}
j = 1,2,\cdots,i-1 \\
\boxed{l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}\cdot d_k \cdot l_{jk}}
\end{cases} \\
d_i = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}{}^2 \cdot d_k \right)^{-1} = l_{ii}{}^{-1}
\end{cases}
$$

不完全修正コレスキー分解

$$\sum_{k=1}^{j} l_{ik} \cdot d_k \cdot l_{jk} = a_{ij} \quad (j = 1,2,\cdots,i-1)$$

$$\sum_{k=1}^{i} l_{ik} \cdot d_k \cdot l_{ik} = a_{ii}$$

ここで $l_{ii} \cdot d_i = 1$ とすると以下が導かれる

$$i = 1,2,\cdots,n$$

$$j = 1,2,\cdots,i-1$$

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} \cdot d_k \cdot l_{jk} \rightarrow \boxed{l_{ij} = a_{ij}}$$

$$\boxed{d_i = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}{}^2 \cdot d_k \right)^{-1} = l_{ii}{}^{-1}}$$

対角成分のみがもとの
行列と変わる

不完全修正コレスキー分解を使用した前進後退代入

$$(M)\{z\}=\left(LDL^T\right)\{z\}=\{r\}$$

$$\{z\}=\left(LDL^T\right)^{-1}\{r\} \longrightarrow$$

$$(L)\{y\}=\{r\}$$
$$\left(DL^T\right)\{z\}=\{y\}$$

$$\begin{bmatrix} d_1 & 0 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 & 0 \\ 0 & 0 & d_3 & 0 & 0 \\ 0 & 0 & 0 & d_4 & 0 \\ 0 & 0 & 0 & 0 & d_5 \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} & l_{41} & l_{51} \\ 0 & l_{22} & l_{32} & l_{42} & l_{52} \\ 0 & 0 & l_{33} & l_{43} & l_{53} \\ 0 & 0 & 0 & l_{44} & l_{54} \\ 0 & 0 & 0 & 0 & l_{55} \end{bmatrix} = \begin{bmatrix} 1 & l_{21}/l_{11} & l_{31}/l_{11} & l_{41}/l_{11} & l_{51}/l_{11} \\ 0 & 1 & l_{32}/l_{22} & l_{42}/l_{22} & l_{52}/l_{22} \\ 0 & 0 & 1 & l_{43}/l_{33} & l_{53}/l_{33} \\ 0 & 0 & 0 & 1 & l_{54}/l_{44} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

不完全修正コレスキー分解を使用した前進後退代入.

$$(L)\{y\}=\{r\}$$

$$(DL^T)\{z\}=\{y\}$$

```
for(i=0; i<N; i++) {
    W[Y][i] = W[R][i];
}

for(i=0; i<N; i++) {
    WVAL = W[Y][i];
    for(j=indexL[i]; j<indexL[i+1]; j++) {
        WVAL -= AL[j] * W[Y][itemL[j]-1];
    }
    W[Y][i] = WVAL * W[DD][i];
}

for(i=N-1; i>=0; i--) {
    SW = 0.0;
    for(j=indexU[i]; j<indexU[i+1]; j++) {
    SW += AU[j] * W[Z][itemU[j]-1];
    }
    W[Z][i]= W[Y][i]  - W[DD][i] * SW;
}
```

$$W[DD][i]=1/l_{ii}=d_i$$

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} & 0 \\ l_{51} & l_{52} & l_{53} & l_{54} & l_{55} \end{bmatrix}$$

$$\begin{bmatrix} 1 & l_{21}/l_{11} & l_{31}/l_{11} & l_{41}/l_{11} & l_{51}/l_{11} \\ 0 & 1 & l_{32}/l_{22} & l_{42}/l_{22} & l_{52}/l_{22} \\ 0 & 0 & 1 & l_{43}/l_{33} & l_{53}/l_{33} \\ 0 & 0 & 0 & 1 & l_{54}/l_{44} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

不完全修正コレスキー分解:前進代入

```
for(i=0; i<N; i++) {
    W[Y][i] = W[R][i];
}

for(i=0; i<N; i++) {
    WVAL = W[Y][i];
    for(j=indexL[i]; j<indexL[i+1]; j++) {
        WVAL -= AL[j] * W[Y][itemL[j]-1];
    }
    W[Y][i] = WVAL * W[DD][i];
}
```

$$(L)\{y\}=\{r\}$$

$$W(i,DD)=1/l_{ii}=d_i$$

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} & 0 \\ l_{51} & l_{52} & l_{53} & l_{54} & l_{55} \end{bmatrix}$$

$$l_{11}y_1 = r_1$$

$$l_{21}y_1 + l_{22}y_2 = r_2$$

$$\vdots$$

$$l_{n1}y_1 + l_{n2}y_2 + \cdots + l_{nn}y_n = r_n$$

$$\Leftrightarrow$$

$$y_1 = r_1 / l_{11}$$

$$y_2 = \left(r_2 - l_{21}y_1\right) / l_{22}$$

$$\vdots$$

$$y_n = \left(r_n - l_{n1}y_1 - l_{n2}y_2 \ldots = r_n - \sum_{j=1}^{n-1} l_{nj}y_j \right) / l_{nn}$$

不完全修正コレスキー分解：前進代入

```
for(i=0; i<N; i++) {
    W[Y][i] = W[R][i];
}

for(i=0; i<N; i++) {
    WVAL = W[Y][i];
    for(j=indexL[i]; j<indexL[i+1]; j++) {
        WVAL -= AL[j] * W[Y][itemL[j]-1];
    }
    W[Y][i] = WVAL * W[DD][i];
}
```

$$(L)\{y\}=\{r\}$$

$$W(i,DD)=1/l_{ii}=d_i$$

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} & 0 \\ l_{51} & l_{52} & l_{53} & l_{54} & l_{55} \end{bmatrix}$$

$$y_i = \left(r_i - \sum_{j=1}^{i-1} l_{ij} y_j \right) / l_{ii}$$

WVAL

不完全修正コレスキー分解：後退代入

```
for(i=N-1; i>=0; i--) {
    SW = 0.0;
    for(j=indexU[i]; j<indexU[i+1]; j++) {
    SW += AU[j] * W[Z][itemU[j]-1];
    }
    W[Z][i]= W[Y][i]  - W[DD][i] * SW;
}
```

$$\left(DL^T\right)\{z\}=\{y\}$$

$$W(i,DD)=1/l_{ii}=d_i$$

$$\begin{bmatrix} 1 & l_{21}/l_{11} & l_{31}/l_{11} & l_{41}/l_{11} & l_{51}/l_{11} \\ 0 & 1 & l_{32}/l_{22} & l_{42}/l_{22} & l_{52}/l_{22} \\ 0 & 0 & 1 & l_{43}/l_{33} & l_{53}/l_{33} \\ 0 & 0 & 0 & 1 & l_{54}/l_{44} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} & z_n = y_n \\ & z_{n-1} + \left(l_{n-1,n}/l_{n-1,n-1}\right) z_n = y_{n-1} \\ & \vdots \\ & z_1 + \left(l_{21}/l_{11}\right) z_2 + \cdots + \left(l_{n1}/l_{11}\right) z_n = y_1 \end{aligned} \quad \Longleftrightarrow \quad \begin{aligned} & z_n = y_n \\ & z_{n-1} = y_{n-1} - \left(l_{n-1,n} z_n\right)/l_{n-1,n-1} \\ & \vdots \\ & z_1 = y_1 - \left(\sum_{j=2}^{n} l_{j1} z_j\right)/l_{11} \end{aligned}$$

不完全修正コレスキー分解：後退代入

```
for(i=N-1; i>=0; i--) {
    SW = 0.0;
    for(j=indexU[i]; j<indexU[i+1]; j++) {
    SW += AU[j] * W[Z][itemU[j]-1];
    }
    W[Z][i]= W[Y][i]  - W[DD][i] * SW;
}
```

$$(DL^T)\{z\}=\{y\}$$

$$W(i,DD)=1/l_{ii}=d_i$$

$$\begin{bmatrix} 1 & l_{21}/l_{11} & l_{31}/l_{11} & l_{41}/l_{11} & l_{51}/l_{11} \\ 0 & 1 & l_{32}/l_{22} & l_{42}/l_{22} & l_{52}/l_{22} \\ 0 & 0 & 1 & l_{43}/l_{33} & l_{53}/l_{33} \\ 0 & 0 & 0 & 1 & l_{54}/l_{44} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$z_i = y_i - \underbrace{\left(\sum_{j=i+1}^{n} l_{ij} z_j\right)}_{\text{SW}} / l_{ii}$$

不完全修正コレスキー分解を使用した前進後退代入:計算順序考慮

$$(L)\{z\}=\{z\}$$

$$(DL^T)\{z\}=\{z\}$$

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

for(i=0; i<N; i++) {
    WVAL = W[Z][i];
    for(j=indexL[i]; j<indexL[i+1]; j++) {
        WVAL -= AL[j] * W[Z][itemL[j]-1];
    }
    W[Z][i] = WVAL * W[DD][i];
}

for(i=N-1; i>=0; i--) {
    SW = 0.0;
    for(j=indexU[i]; j<indexU[i+1]; j++) {
    SW += AU[j] * W[Z][itemU[j]-1];
    }
    W[Z][i]= W[Z][i]  - W[DD][i] * SW;
}
```

$$W[DD][i]=1/l_{ii}=d_i$$

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} & 0 \\ l_{51} & l_{52} & l_{53} & l_{54} & l_{55} \end{bmatrix}$$

$$\begin{bmatrix} 1 & l_{21}/l_{11} & l_{31}/l_{11} & l_{41}/l_{11} & l_{51}/l_{11} \\ 0 & 1 & l_{32}/l_{22} & l_{42}/l_{22} & l_{52}/l_{22} \\ 0 & 0 & 1 & l_{43}/l_{33} & l_{53}/l_{33} \\ 0 & 0 & 0 & 1 & l_{54}/l_{44} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

solve_ICCG (1/7): METHOD= 1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <math.h> etc.

#include "solver_ICCG.h"

extern int
solve_ICCG (int N, int NL, int NU, int *indexL, int *itemL,
            int *indexU, int *itemU,
            double *D, double *B, double *X, double *AL,
            double *AU,
            double EPS, int *ITR, int *IER)
{
  double **W;
  double VAL, BNRM2, WVAL, SW, RHO, BETA, RHO1, C1, DNRM2;
  doble  ALPHA, ERR;

        int i, j, ic, ip, L, ip1;
        int R  = 0;
        int Z  = 1;
        int Q  = 1;
        int P  = 2;
        int DD = 3;
```

ICELTOT → N
BFORCE → B
PHI → X
EPSICCG → EPS

W[0][i]= W[R][i} ⇒ {r}
W[1][i]= W[Z][i} ⇒ {z}
W[1][i]= W[Q][i} ⇒ {q}
W[2][i]= W[P][i} ⇒ {p}

W[3][i]= W[DD][i} ⇒ {d}

solve_ICCG (2/7): METHOD= 1

```
W = (double **)malloc(sizeof(double *)*4);
if(W == NULL) {
   fprintf(stderr, "Error: %s¥n", strerror(errno));
   return -1;
}

for(i=0; i<4; i++) {
    W[i] = (double *)malloc(sizeof(double)*N);
    if(W[i] == NULL) {
       fprintf(stderr, "Error: %s¥n",
strerror(errno));
       return -1;
    }
}

for(i=0; i<N; i++) {
    X[i] = 0.0;
    W[1][i] = 0.0;
    W[2][i] = 0.0;
    W[3][i] = 0.0;
}


for(i=0; i<N; i++) {
    VAL = D[i];
    for(j=indexL[i]; j<indexL[i+1]; j++) {
        VAL = VAL - AL[j]*AL[j]*W[DD][itemL[j] - 1];
    }
    W[DD][i] = 1.0 / VAL;
}
```

不完全修正コレスキー分解

W[DD][i]= d_i

$$i = 1,2,\cdots,n$$

$$j = 1,2,\cdots,i-1$$

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} \cdot d_k \cdot l_{jk} \rightarrow \boxed{l_{ij} = a_{ij}}$$

$$\boxed{d_i = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}{}^2 \cdot d_k \right)^{-1} = l_{ii}{}^{-1}}$$

Only diagonal components are changed

W[DD][i]:	d_i
D[i]:	a_{ii}
itemL[j]:	k
AL[j]:	a_{ik}

solve_ICCG (3/7): METHOD= 1

```
for(i=0; i<N; i++) {
    VAL = D[i] * X[i];
    for(j=indexL[i]; j<indexL[i+1]; j++) {
        VAL += AL[j] * X[itemL[j]-1];
    }
    for(j=indexU[i]; j<indexU[i+1]; j++) {
        VAL += AU[j] * X[itemU[j]-1];
    }
    W[R][i] = B[i] - VAL;
}

BNRM2 = 0.0;
for(i=0; i<N; i++) {
    BNRM2 += B[i]*B[i];
}
```

BNRM2=|b|^2
Convergence criteria

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

solve_ICCG (4/7): METHOD= 1

```
*ITR = N;
for(L=0; L<(*ITR); L++) {

    for(i=0; i<N; i++) {
        W[Z][i] = W[R][i];
    }

    for(i=0; i<N; i++) {
        WVAL = W[Z][i];
        for(j=indexL[i]; j<indexL[i+1]; j++) {
            WVAL -= AL[j] * W[Z][itemL[j]-1];
        }
        W[Z][i] = WVAL * W[DD][i];
    }

    for(i=N-1; i>=0; i--) {
        SW = 0.0;
        for(j=indexU[i]; j<indexU[i+1]; j++) {
        SW += AU[j] * W[Z][itemU[j]-1];
        }
        W[Z][i]= W[Z][i]  - W[DD][i] * SW;
    }
```

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

solve_ICCG (4/7): METHOD= 1

$$(M)\{z\} = (LDL^T)\{z\} = \{r\}$$

$$(L)\{z\} = \{r\}$$

```
for(i=0; i<N; i++) {
    W[Z][i] = W[R][i];
}
for(i=0; i<N; i++) {
    WVAL = W[Z][i];
    for(j=indexL[i]; j<indexL[i+1]; j++) {
        WVAL -= AL[j] * W[Z][itemL[j]-1];
    }
    W[Z][i] = WVAL * W[DD][i];
}
```

前進代入
Forward Substitution

$$(DL^T)\{z\} = \{z\}$$

```
for(i=N-1; i>=0; i--) {
    SW = 0.0;
    for(j=indexU[i]; j<indexU[i+1]; j++) {
    SW += AU[j] * W[Z][itemU[j]-1];
    }
    W[Z][i]= W[Z][i]  - W[DD][i] * SW;
}
```

後退代入
Backward Substitution

solve_ICCG (5/7): METHOD= 1

```
/*****************
 * RHO = {r}{z} *
 *****************/

RHO = 0.0;
for(i=0; i<N; i++) {
    RHO += W[R][i] * W[Z][i];
}
```

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

solve_ICCG (6/7): METHOD= 1

```
/*******************************
 * {p}  = {z} if      ITER=0    *
 * BETA = RHO / RHO1  otherwise *
 *******************************/

if(L == 0) {
   for(i=0; i<N; i++) {
       W[P][i] = W[Z][i];
   }
   } else {
   BETA = RHO / RHO1;
   for(i=0; i<N; i++) {
       W[P][i] = W[Z][i] + BETA * W[P][i];
   }
}

/***************
 * {q} = [A]{p} *
 ***************/

for(i=0; i<N; i++) {
    VAL = D[i] * W[P][i];
    for(j=indexL[i]; j<indexL[i+1]; j++) {
        VAL += AL[j] * W[P][itemL[j]-1];
    }
    for(j=indexU[i]; j<indexU[i+1]; j++) {
        VAL += AU[j] * W[P][itemU[j]-1];
    }
    W[Q][i] = VAL;
}
```

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

solve_ICCG (7/7): METHOD= 1

```
/************************
 * ALPHA = RHO / {p}{q} *
 ************************/
C1 = 0.0;
for(i=0; i<N; i++) {
    C1 += W[P][i] * W[Q][i];
}
ALPHA = RHO / C1;

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

DNRM2 = 0.0;
for(i=0; i<N; i++) {
    DNRM2 += W[R][i]*W[R][i];
}

ERR = sqrt(DNRM2/BNRM2);
if((L+1)%100 ==1) {
   fprintf(stderr, "%5d%16.6e¥n", L+1, ERR);
}
if(ERR < EPS) {
   *IER = 0; goto N900;
  } else {
   RHO1 = RHO;
  }
}
*IER = 1;
```

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

solve_ICCG (7/7): METHOD= 1

```
/*************************
 * ALPHA = RHO / {p}{q} *
 *************************/
C1 = 0.0;
for(i=0; i<N; i++) {
    C1 += W[P][i] * W[Q][i];
}
ALPHA = RHO / C1;

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

DNRM2 = 0.0;
for(i=0; i<N; i++) {
    DNRM2 += W[R][i]*W[R][i];
}

ERR = sqrt(DNRM2/BNRM2);
if((L+1)%100 ==1) {
   fprintf(stderr, "%5d%16.6e¥n", L+1, ERR);
}
if(ERR < EPS) {
   *IER = 0; goto N900;
  } else {
   RHO1 = RHO;
  }
}
*IER = 1;
```

$$|r|, |b| : 2 / L2 / Euclidean-norm \quad (\|r\|_2, \|b\|_2)$$

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

r= b-[A]x
DNRM2=|r|²
BNRM2=|b|²

ERR= |r|/|b|

$$ERR = \sqrt{\frac{\mathrm{DNorm2}}{\mathrm{BNorm2}}} = \frac{|r|}{|b|} = \frac{|b-Ax|}{|b|} \le \mathrm{Eps}$$

$$Ax = b \Rightarrow \alpha Ax = \alpha b$$

$$r = b - Ax \Rightarrow R = \alpha b - \alpha Ax = \alpha r$$

solve_ICCG2 (1/3): METHOD= 2

Fortran ONLY

```
!C
!C***
!C*** module solver_ICCG2
!C***
!
      module solver_ICCG2
      contains
!C
!C*** solve_ICCG2
!C
      subroutine solve_ICCG2                                            &
     &  ( N, NPL, NPU, indexL, itemL, indexU, itemU, D, B, X,          &
     &    AL, AU, EPS, ITR, IER)

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

      real(kind=8), dimension(N)   :: D
      real(kind=8), dimension(N)   :: B
      real(kind=8), dimension(N)   :: X
      real(kind=8), dimension(NPL) :: AL
      real(kind=8), dimension(NPU) :: AU

      integer, dimension(0:N)   :: indexL, indexU
      integer, dimension(NPL):: itemL
      integer, dimension(NPU):: itemU

      real(kind=8), dimension(:,:), allocatable :: W

      integer, parameter ::  R= 1
      integer, parameter ::  Z= 2
      integer, parameter ::  Q= 2
      integer, parameter ::  P= 3
      integer, parameter :: DD= 4

      real(kind=8), dimension(:), allocatable :: ALlu0, AUlu0
      real(kind=8), dimension(:), allocatable :: Dlu0
```

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

solve_ICCG2（2/3）: METHOD=2

```
!C
!C +-------+
!C | INIT |
!C +-------+
!C===
      allocate (W(N,4))

      do i= 1, N
        X(i)   = 0.d0
        W(i,1)= 0.0D0
        W(i,2)= 0.0D0
        W(i,3)= 0.0D0
        W(i,4)= 0.0D0
      enddo

      call FORM_ILU0
!C===
```

Dlu0, ALlu0，AUlu0にはILU(0)分解された対角，下三角，上三角成分が入る（行列[M]）。

FORM_ILU0(1/2)

不完全修正コレスキー分解：正確には不完全修正LU分解 solver_ICCG2.fに附属

```
 contains

!C
!C***
!C*** FORM_ILU0
!C***
!C
!C    form ILU(0) matrix
!C
      subroutine FORM_ILU0
      implicit REAL*8 (A-H,O-Z)
      integer, dimension(:), allocatable :: IW1 , IW2
      integer, dimension(:), allocatable :: IWsL, IWsU
      real (kind=8):: RHS_Aij, DkINV, Aik, Akj

!C
!C +--------+
!C | INIT. |
!C +--------+
!C===
      allocate (ALlu0(NPL), AUlu0(NPU))
      allocate (Dlu0(N))

      do i= 1, N
        Dlu0(i)= D(i)
        do k= 1, INL(i)
          ALlu0(k,i)= AL(k,i)
        enddo

        do k= 1, INU(i)
          AUlu0(k,i)= AU(k,i)
        enddo
      enddo
!C===
```

$$
\begin{cases}
i = 1,2,\cdots,n \\
\quad \begin{cases}
j = 1,2,\cdots,i-1 \\
\quad l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}\cdot d_k \cdot l_{jk} \\
\quad d_i = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}{}^2 \cdot d_k \right)^{-1} = l_{ii}{}^{-1}
\end{cases}
\end{cases}
$$

Dlu0, ALlu0, AUlu0にはILU(0)分解された対角，下三角，上三角成分が入る（行列[M]）。

「Dlu0,ALlu0,AUlu0」初期値として，「D,AL,AU」の値を代入する。

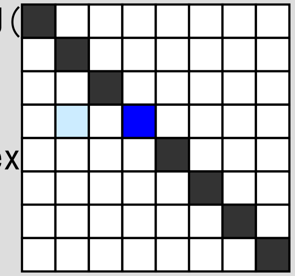
FORM_ILU0 (2/2)

```
!C
!C +----------------------+
!C | ILU(0) factorization |
!C +----------------------+
!C===
      allocate (IW1(N) , IW2(N))
      IW1=0
      IW2= 0

      do i= 1, N
        do k0= indexL(i-1)+1, indexL(i)
          IW1(itemL(k0))= k0
        enddo

        do k0= indexU(i-1)+1, indexU(i)
          IW2(itemU(k0))= k0
        enddo

        do icon= indexL(i-1)+1, indexL(i)
            k= itemL(icon)
          D11= Dlu0(k)

          DkINV= 1.d0/D11
            Aik= ALlu0(icon)

          do kcon= indexU(k-1)+1, indexU(k)
            j= itemU(kcon)

            if (j.eq.i) then
                   Akj= AUlu0(kcon)
              RHS_Aij= Aik * DkINV * Akj
              Dlu0(i)= Dlu0(i) - RHS_Aij
            endif

            if (j.lt.i .and. IW1(j).ne.0) then
                   Akj= AUlu0(kcon)
              RHS_Aij= Aik * DkINV * Akj

                       ij0 = IW1(j)
              ALlu0(ij0)= ALlu0(ij0) - RHS_Aij
            endif

            if (j.gt.i .and. IW2(j).ne.0) then
                   Akj= AUlu0(kcon)
              RHS_Aij= Aik * DkINV * Akj

                       ij0 = IW2(j)
              AUlu0(ij0)= AUlu0(ij0) - RHS_Aij
            endif

          enddo
        enddo

        do k0= indexL(i-1)+1, indexL(i)
          IW1(itemL(k0))= 0
        enddo

        do k0= indexU(i-1)+1, indexU(i)
          IW2(itemU(k0))= 0
        enddo
      enddo

      do i= 1, N
        Dlu0(i)= 1.d0 / Dlu0(i)
      enddo
      deallocate (IW1, IW2)
!C===
      end subroutine FORM_ILU0
```

```
do i= 1, N
  do k= 1, i-1
    if (A(i,k) is non-zero) then
      do j= k+1, N
        if (A(i,j) is non-zero) then
          A(i,j)=  A(i,j)                                  &
                  -A(i,k)*(A(k,k))^-1*A(k,j)
        endif
      enddo
    endif
  enddo
enddo
```

FORM_ILU0 (2/2)

```
!C
!C +----------------------+
!C | ILU(0) factorization |
!C +----------------------+
!C===
      allocate (IW1(N) , IW2(N))
      IW1=0
      IW2= 0

      do i= 1, N
        do k0= indexL(i-1)+1, indexL(i)
          IW1(itemL(k0))= k0
        enddo

        do k0= indexU(i-1)+1, indexU(
          IW2(itemU(k0))= k0
        enddo

        do icon= indexL(i-1)+1, index
            k= itemL(icon)
          D11= Dlu0(k)

          DkINV= 1.d0/D11
            Aik= ALlu0(icon)

          do kcon= indexU(k-1)+1, indexU(k)
            j= itemU(kcon)

            if (j.eq.i) then
                  Akj= AUlu0(kcon)
              RHS_Aij= Aik * DkINV * Akj
              Dlu0(i)= Dlu0(i) - RHS_Aij
            endif

            if (j.lt.i .and. IW1(j).ne.0) then
                  Akj= AUlu0(kcon)
              RHS_Aij= Aik * DkINV * Akj

                      ij0 = IW1(j)
                ALlu0(ij0)= ALlu0(ij0) - RHS_Aij
            endif
```

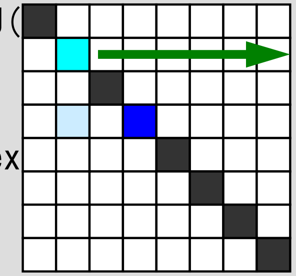


```
            if (j.gt.i .and. IW2(j).ne.0) then
                  Akj= AUlu0(kcon)
              RHS_Aij= Aik * DkINV * Akj

                      ij0 = IW2(j)
              AUlu0(ij0)= AUlu0(ij0) - RHS_Aij
            endif

          enddo
        enddo

        do k0= indexL(i-1)+1, indexL(i)
          IW1(itemL(k0))= 0
        enddo

        do k0= indexU(i-1)+1, indexU(i)
          IW2(itemU(k0))= 0
        enddo
      enddo

      do i= 1, N
        Dlu0(i)= 1.d0 / Dlu0(i)
      enddo
      deallocate (IW1, IW2)
!C===
      end subroutine FORM_ILU0
```

```
do i= 1, N
  do k= 1, i-1
    if (A(i,k) is non-zero) then
      do j= k+1, N
        if (A(i,j) is non-zero) then
          A(i,j)=  A(i,j)                                   &
                 -A(i,k)*(A(k,k))^-1*A(k,j)
        endif
      enddo
    endif
  enddo
enddo
```

FORM_ILU0 (2/2)

```
!C
!C +----------------------+
!C | ILU(0) factorization |
!C +----------------------+
!C===
      allocate (IW1(N) , IW2(N))
      IW1=0
      IW2= 0

      do i= 1, N
        do k0= indexL(i-1)+1, indexL(i)
          IW1(itemL(k0))= k0
        enddo

        do k0= indexU(i-1)+1, indexU(
          IW2(itemU(k0))= k0
        enddo

        do icon= indexL(i-1)+1, index
            k= itemL(icon)
          D11= Dlu0(k)

          DkINV= 1.d0/D11
            Aik= ALlu0(icon)

          do kcon= indexU(k-1)+1, indexU(k)
            j= itemU(kcon)

            if (j.eq.i) then
                  Akj= AUlu0(kcon)
              RHS_Aij= Aik * DkINV * Akj
              Dlu0(i)= Dlu0(i) - RHS_Aij
            endif

            if (j.lt.i .and. IW1(j).ne.0) then
                  Akj= AUlu0(kcon)
              RHS_Aij= Aik * DkINV * Akj

                     ij0 = IW1(j)
              ALlu0(ij0)= ALlu0(ij0) - RHS_Aij
            endif
```



```
            if (j.gt.i .and. IW2(j).ne.0) then
                  Akj= AUlu0(kcon)
              RHS_Aij= Aik * DkINV * Akj

                     ij0 = IW2(j)
              AUlu0(ij0)= AUlu0(ij0) - RHS_Aij
            endif

          enddo
        enddo

        do k0= indexL(i-1)+1, indexL(i)
          IW1(itemL(k0))= 0
        enddo

        do k0= indexU(i-1)+1, indexU(i)
          IW2(itemU(k0))= 0
        enddo
      enddo

      do i= 1, N
        Dlu0(i)= 1.d0 / Dlu0(i)
      enddo
      deallocate (IW1, IW2)
!C===
      end subroutine FORM_ILU0
```

```
do i= 1, N
  do k= 1, i-1
    if (A(i,k) is non-zero) then
      do j= k+1, N
        if (A(i,j) is non-zero) then
          A(i,j)=  A(i,j)                                &
                  -A(i,k)*(A(k,k))^-1*A(k,j)
        endif
      enddo
    endif
  enddo
enddo
```

FORM_ILU0 (2/2)

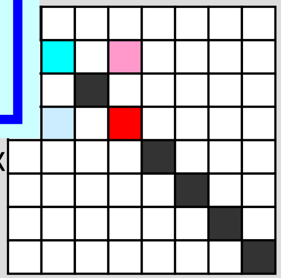
```
!C
!C +----------------------+
!C | ILU(0) factorization |
!C +----------------------+
!C===
```

$$i = 1,2,\cdots,n$$

$$j = 1,2,\cdots,i-1$$

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}\cdot d_k \cdot l_{jk}$$

$$\boxed{d_i = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}{}^2 \cdot d_k\right)^{-1} = l_{ii}{}^{-1}}$$



```
        do icon= indexL(i-1)+1, index
          k= itemL(icon)
          D11= Dlu0(k)

          DkINV= 1.d0/D11
            Aik= ALlu0(icon)

          do kcon= indexU(k-1)+1, indexU(k)
            j= itemU(kcon)

            if (j.eq.i) then
                   Akj= AUlu0(kcon)
              RHS_Aij= Aik * DkINV * Akj
              Dlu0(i)= Dlu0(i) - RHS_Aij
            endif

            if (j.lt.i .and. IW1(j).ne.0) then
                   Akj= AUlu0(kcon)
              RHS_Aij= Aik * DkINV * Akj

                      ij0 = IW1(j)
              ALlu0(ij0)= ALlu0(ij0) - RHS_Aij
            endif
```

j=i

```
            if (j.gt.i .and. IW2(j).ne.0) then
                   Akj= AUlu0(kcon)
              RHS_Aij= Aik * DkINV * Akj

                      ij0 = IW2(j)
              AUlu0(ij0)= AUlu0(ij0) - RHS_Aij
            endif

          enddo
        enddo

        do k0= indexL(i-1)+1, indexL(i)
          IW1(itemL(k0))= 0
        enddo

        do k0= indexU(i-1)+1, indexU(i)
          IW2(itemU(k0))= 0
        enddo
      enddo

      do i= 1, N
        Dlu0(i)= 1.d0 / Dlu0(i)
      enddo
      deallocate (IW1, IW2)
!C===
      end subroutine FORM_ILU0
```

```
do i= 1, N
  do k= 1, i-1
    if (A(i,k) is non-zero) then
      do j= k+1, N
        if (A(i,j) is non-zero) then
          A(i,j)=  A(i,j)                        &
                 -A(i,k)*(A(k,k))^-1*A(k,j)
        endif
      enddo
    endif
  enddo
enddo
```

FORM_ILU0 (2/2)

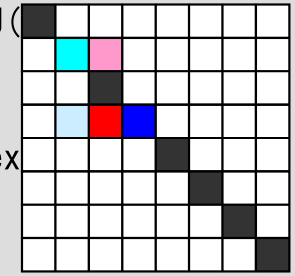
$$
i = 1,2,\cdots,n
$$

$$
j = 1,2,\cdots,i-1
$$

$$
l_{ij} = a_{ij} - \boxed{\sum_{k=1}^{j-1} l_{ik} \cdot d_k \cdot l_{jk}}
$$

$$
d_i = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}{}^2 \cdot d_k \right)^{-1} = l_{ii}{}^{-1}
$$

```
      do k0= indexU(i-1)+1, indexU(
        IW2(itemU(k0))= k0
      enddo

      do icon= indexL(i-1)+1, index
          k= itemL(icon)
        D11= Dlu0(k)

        DkINV= 1.d0/D11
          Aik= ALlu0(icon)

        do kcon= indexU(k-1)+1, indexU(k)
          j= itemU(kcon)

          if (j.eq.i) then
                 Akj= AUlu0(kcon)
            RHS_Aij= Aik * DkINV * Akj
            Dlu0(i)= Dlu0(i) - RHS_Aij
          endif

          if (j.lt.i .and. IW1(j).ne.0) then
                 Akj= AUlu0(kcon)
            RHS_Aij= Aik * DkINV * Akj

                      ij0 = IW1(j)
              ALlu0(ij0)= ALlu0(ij0) - RHS_Aij
          endif
```



j<i

```
          if (j.gt.i .and. IW2(j).ne.0) then
                 Akj= AUlu0(kcon)
            RHS_Aij= Aik * DkINV * Akj

                      ij0 = IW2(j)
            AUlu0(ij0)= AUlu0(ij0) - RHS_Aij
          endif

        enddo
      enddo

      do k0= indexL(i-1)+1, indexL(i)
        IW1(itemL(k0))= 0
      enddo

      do k0= indexU(i-1)+1, indexU(i)
        IW2(itemU(k0))= 0
      enddo
    enddo

    do i= 1, N
      Dlu0(i)= 1.d0 / Dlu0(i)
    enddo
    deallocate (IW1, IW2)
!C===
    end subroutine FORM_ILU0
```

```
do i= 1, N
  do k= 1, i-1
    if (A(i,k) is non-zero) then
      do j= k+1, N
        if (A(i,j) is non-zero) then
          A(i,j)=  A(i,j)                              &
                -A(i,k)*(A(k,k))-1*A(k,j)
        endif
      enddo
    endif
  enddo
enddo
```

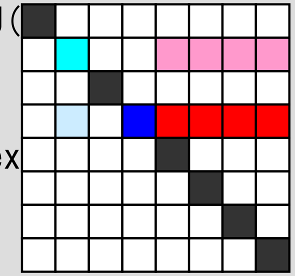
FORM_ILU0 (2/2)

$$i = 1,2,\cdots,n$$

$$j = 1,2,\cdots,i-1$$

$$l_{ij} = a_{ij} - \boxed{\sum_{k=1}^{j-1} l_{ik}\cdot d_k \cdot l_{jk}}$$

$$d_i = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}^{\,2}\cdot d_k\right)^{-1} = l_{ii}^{\,-1}$$



```
do k0= indexU(i-1)+1, indexU(
  IW2(itemU(k0))= k0
enddo

do icon= indexL(i-1)+1, index
    k= itemL(icon)
  D11= Dlu0(k)

  DkINV= 1.d0/D11
    Aik= ALlu0(icon)

  do kcon= indexU(k-1)+1, indexU(k)
    j= itemU(kcon)

    if (j.eq.i) then
          Akj= AUlu0(kcon)
      RHS_Aij= Aik * DkINV * Akj
      Dlu0(i)= Dlu0(i) - RHS_Aij
    endif

    if (j.lt.i .and. IW1(j).ne.0) then
          Akj= AUlu0(kcon)
      RHS_Aij= Aik * DkINV * Akj

              ij0 = IW1(j)
      ALlu0(ij0)= ALlu0(ij0) - RHS_Aij
    endif

    if (j.gt.i .and. IW2(j).ne.0) then
          Akj= AUlu0(kcon)
      RHS_Aij= Aik * DkINV * Akj

              ij0 = IW2(j)
      AUlu0(ij0)= AUlu0(ij0) - RHS_Aij
    endif

      enddo
    enddo

    do k0= indexL(i-1)+1, indexL(i)
      IW1(itemL(k0))= 0
    enddo

    do k0= indexU(i-1)+1, indexU(i)
      IW2(itemU(k0))= 0
    enddo
  enddo

  do i= 1, N
    Dlu0(i)= 1.d0 / Dlu0(i)
  enddo
  deallocate (IW1, IW2)
!C===
  end subroutine FORM_ILU0
```

j>i

```
do i= 1, N
  do k= 1, i-1
    if (A(i,k) is non-zero) then
      do j= k+1, N
        if (A(i,j) is non-zero) then
          A(i,j)=  A(i,j)                          &
                -A(i,k)*(A(k,k))^-1*A(k,j)
        endif
      enddo
    endif
  enddo
enddo
```

FORM_ILU0 (2/2)

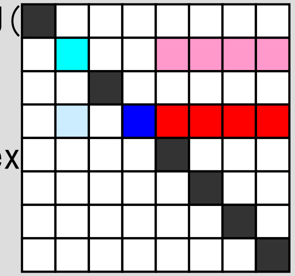
$$i = 1,2,\cdots,n$$

$$j = 1,2,\cdots,i-1$$

$$l_{ij} = a_{ij} - \boxed{\sum_{k=1}^{j-1} l_{ik} \cdot d_k \cdot l_{jk}}$$

$$d_i = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}^{\ 2} \cdot d_k \right)^{-1} = l_{ii}^{\ -1}$$

```
do k0= indexU(i-1)+1, indexU(
  IW2(itemU(k0))= k0
enddo

do icon= indexL(i-1)+1, index
    k= itemL(icon)
  D11= Dlu0(k)

  DkINV= 1.d0/D11
    Aik= ALlu0(icon)

  do kcon= indexU(k-1)+1, indexU(k)
    j= itemU(kcon)

    if (j.eq.i) then
          Akj= AUlu0(kcon)
      RHS_Aij= Aik * DkINV * Akj
      Dlu0(i)= Dlu0(i) - RHS_Aij
    endif

    if (j.lt.i .and. IW1(j).ne.0) then
          Akj= AUlu0(kcon)
      RHS_Aij= Aik * DkINV * Akj

              ij0 = IW1(j)
        ALlu0(ij0)= ALlu0(ij0) - RHS_Aij
    endif
```



j<i

```
    if (j.gt.i .and. IW2(j).ne.0) then
          Akj= AUlu0(kcon)
      RHS_Aij= Aik * DkINV * Akj

              ij0 = IW2(j)
        AUlu0(ij0)= AUlu0(ij0) - RHS_Aij
    endif

    enddo
  enddo

  do k0= indexL(i-1)+1, indexL(i)
    IW1(itemL(k0))= 0
  enddo

  do k0= indexU(i-1)+1, indexU(i)
    IW2(itemU(k0))= 0
  enddo
enddo

do i= 1, N
  Dlu0(i)= 1.d0 / Dlu0(i)
enddo
deallocate (IW1, IW2)
!C===
end subroutine FORM_ILU0
```

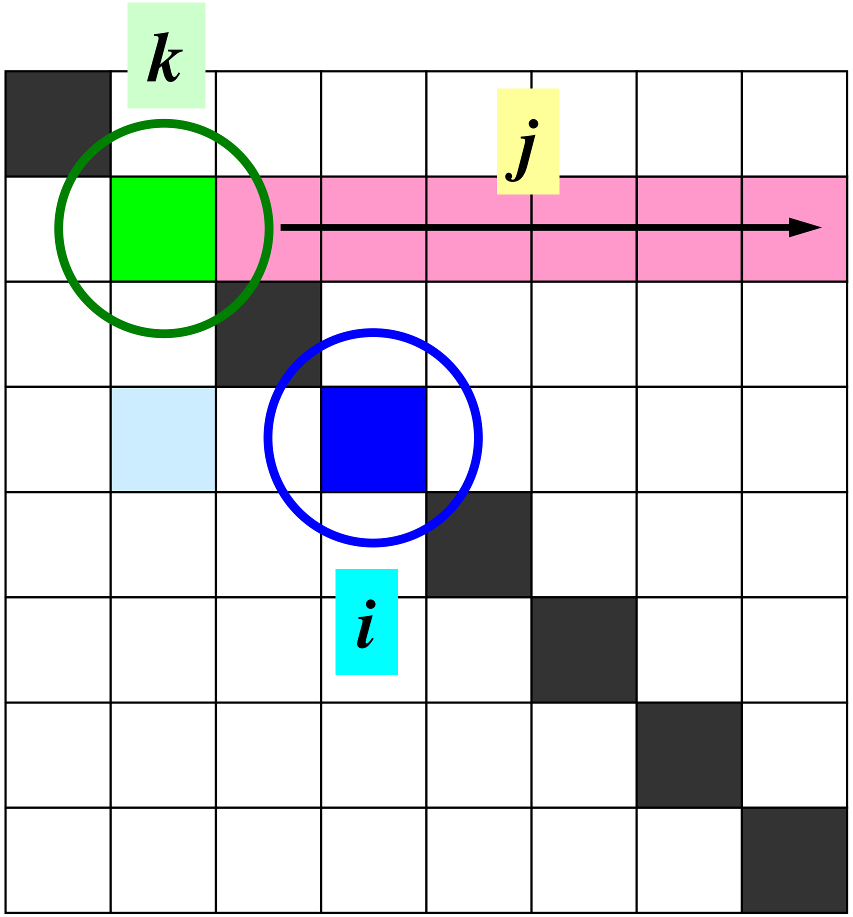
j>i

実はこのIF文の中は通らない（理由は後述）したがって，

ALlu0= AL

AUlu0= AU

$j=i$, $j<i$, $j>i$ （1/3）



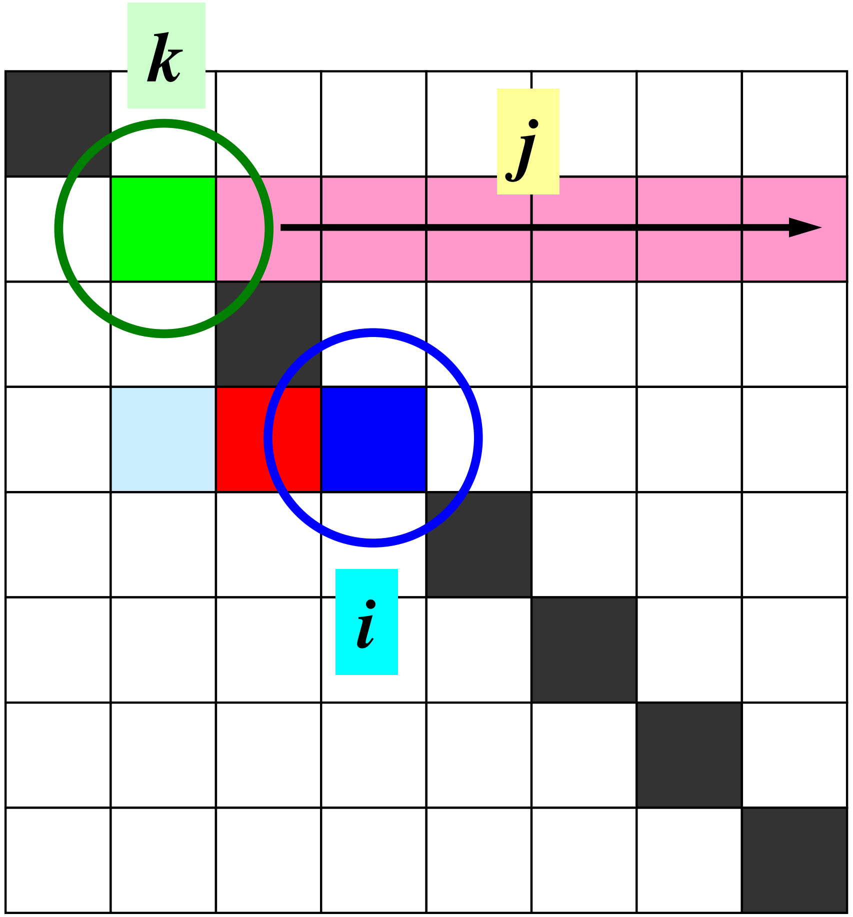
ある要素「i（○）」に接続する下三角成分「k（■○）」の上三角成分「j（■）」が：

（1） j=i

「i」自身である場合，Dlu（■）更新

$$
\begin{cases}
i = 1,2,\cdots,n \\
\quad \begin{cases}
j = 1,2,\cdots,i-1 \\
\quad l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} \cdot d_k \cdot l_{jk} \\
\quad \boxed{d_i = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}{}^2 \cdot d_k \right)^{-1} = l_{ii}{}^{-1}}
\end{cases}
\end{cases}
$$

j=*i*, *j*<*i*, *j*>*i*（2/3）



ある要素「i（○）」に接続する下三角成分「k（■ ○）」の上三角成分「j（■）」が：

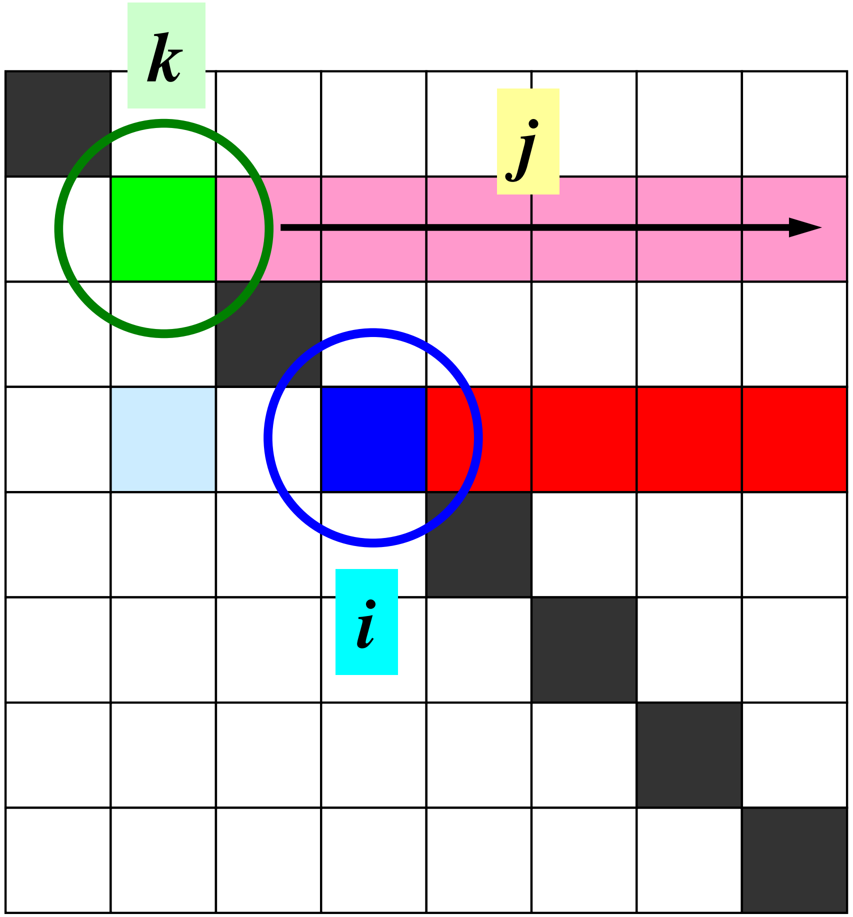
（2） **j<i**

「i」の下三角成分である場合

ALlu0(i-j)（■）更新

$$
\left\{
\begin{array}{l}
i = 1,2,\cdots,n \\
\left\{
\begin{array}{l}
j = 1,2,\cdots,i-1 \\
\quad l_{ij} = a_{ij} - \boxed{\sum_{k=1}^{j-1} l_{ik} \cdot d_k \cdot l_{jk}} \\
d_i = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}^{\,2} \cdot d_k \right)^{-1} = l_{ii}^{-1}
\end{array}
\right.
\end{array}
\right.
$$

j=i, j<i, j>i（3/3）



ある要素「i（○）」に接続する下三角成分「k（■○）」の上三角成分「j（■）」が：

（3）**j>i**

「i」の上三角成分である場合

AUlu0(i-j)（■）更新

実際は（2），（3）に該当する場合は無し

$$
\begin{cases}
i = 1,2,\cdots,n \\
\begin{cases}
j = 1,2,\cdots,i-1 \\
l_{ij} = a_{ij} - \boxed{\sum_{k=1}^{j-1} l_{ik}\cdot d_k \cdot l_{jk}}
\end{cases} \\
d_i = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}{}^2 \cdot d_k\right)^{-1} = l_{ii}{}^{-1}
\end{cases}
$$

solve_ICCG2（3/3）: METHOD=2

```
!C
!C*************************************************** ITERATION
      ITR= N

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

      do i= 1, N
        WVAL= W(i,Z)
        do k= indexL(i-1)+1, indexL(i)
          WVAL= WVAL -  ALlu0(k) * W(itemL(k),Z)
        enddo
        W(i,Z)= WVAL * Dlu0(i)
      enddo

      do i= N, 1, -1
        SW  = 0.0d0
        do k= indexU(i-1)+1, indexU(i)
          SW= SW + AUlu0(k) * W(itemU(k),Z)
        enddo
        W(i,Z)= W(i,Z) - Dlu0(i)*SW
      enddo
!C===
```

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

<u>for</u> i= 1, 2, …

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

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

  <u>if</u> i=1

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

  <u>else</u>

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

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

  <u>endif</u>

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

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

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

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

  check convergence $|r|$

<u>end</u>

これ以下の処理は「solve_ICCG」と全く同じ

solve_ICCG2（3/3）: METHOD=2

```
!C
!C***************************************************  ITERATION
      ITR= N

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

       do i= 1, N
        WVAL= W(i,Z)
        do k= indexL(i-1)+1, indexL(i)
          WVAL= WVAL -  ALlu0(k) * W(itemL(k),Z)
        enddo
        W(i,Z)= WVAL * Dlu0(i)
      enddo

      do i= N, 1, -1
        SW  = 0.0d0
        do k= indexU(i-1)+1, indexU(i)
          SW= SW + AUlu0(k) * W(itemU(k),Z)
        enddo
        W(i,Z)= W(i,Z) - Dlu0(i)*SW
      enddo
!C===
```

$$(M)\{z\}=\left(LDL^T\right)\{z\}=\{r\}$$

$$(L)\{z\}=\{r\}$$

前進代入
Forward Substitution

$$\left(DL^T\right)\{z\}=\{z\}$$

後退代入
Backward Substitution

実は，ALlu0，AUlu0の値はAL，AUと全く同じである。
METHOD=1，METHOD=2の答え（反復回数）は同じ

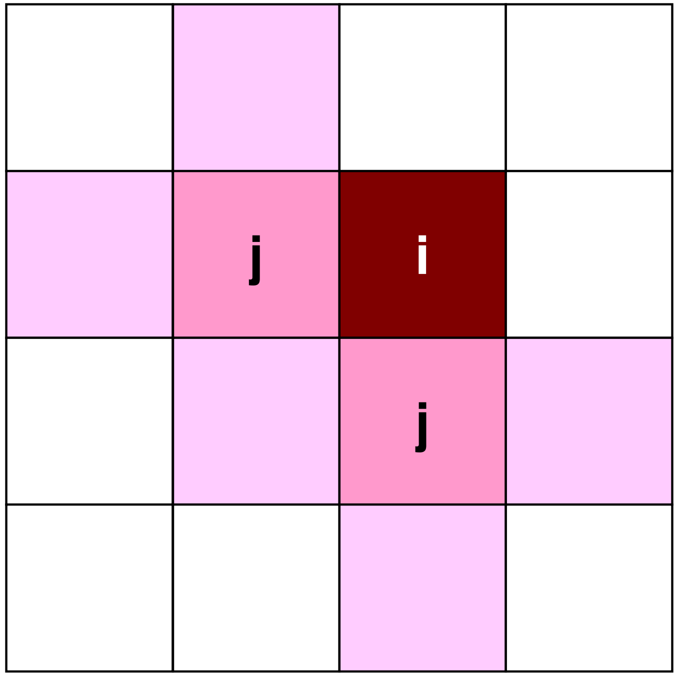
不完全修正コレスキー分解 現在のようなメッシュの場合

$$i=1,2,\cdots,n$$

$$j=1,2,\cdots,i-1$$

$$\boxed{l_{ij}=a_{ij}-\sum_{k=1}^{j-1} l_{ik}\cdot d_k\cdot l_{jk}}$$

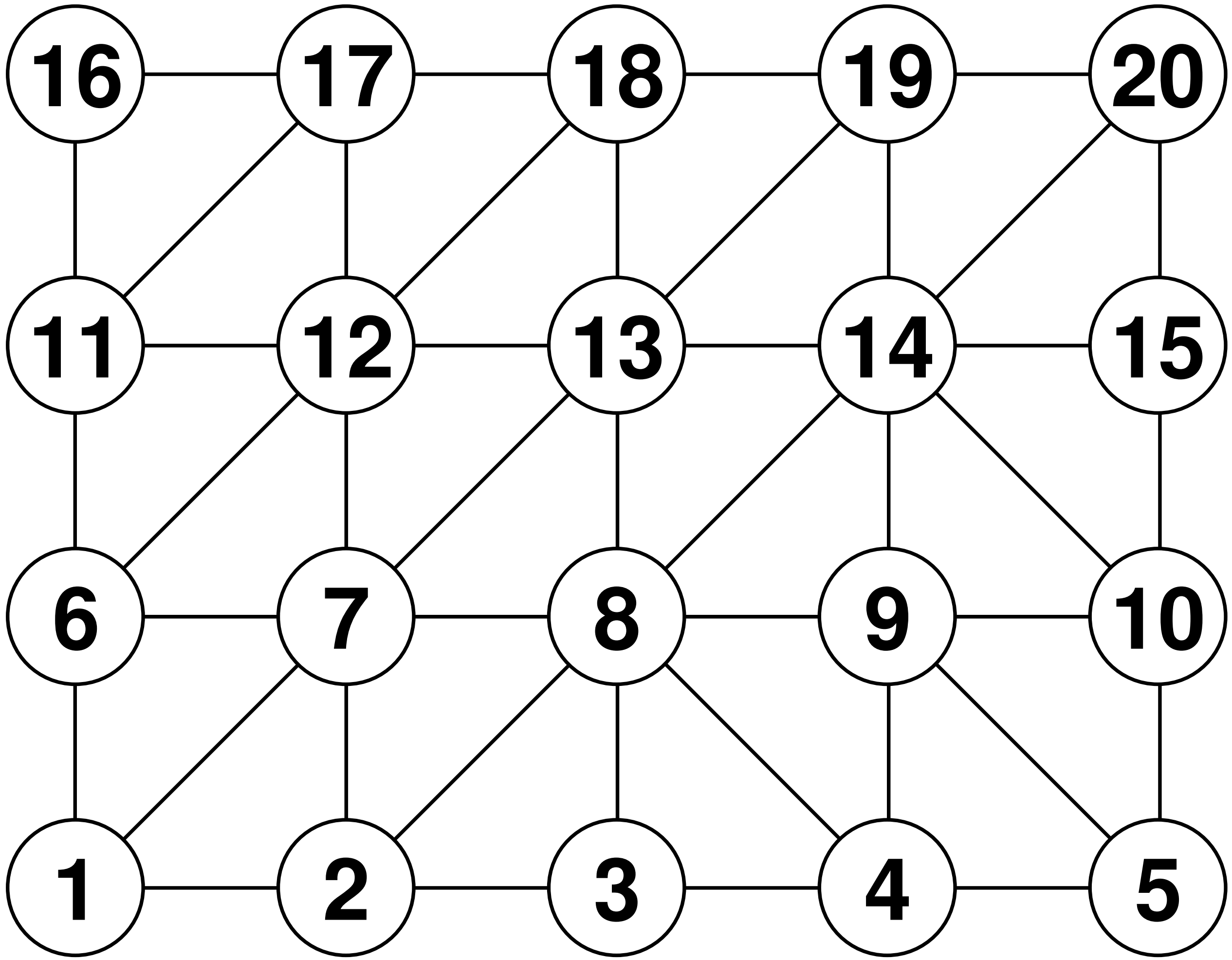
$$d_i=\left(a_{ii}-\sum_{k=1}^{i-1} l_{ik}{}^2\cdot d_k\right)^{-1}=l_{ii}{}^{-1}$$



□を満たすような(i-j-k)（i,jの両方に接続するk）が無い

従って，$l_{ij}= a_{ij}$

こういう場合はAUlu0, ALlu0が更新される可能性あり



ICCG法の並列化

- 内積：OK
- DAXPY：OK
- 行列ベクトル積：OK
- 前処理

前処理はどうするか？対角スケーリングなら簡単：でも遅い

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

```
#omp pragma parallel for private(i)
  for (i=0; i<N; i++){
    W[Z][i]= W[R][i]*W[DD][i];
  }
```

```
#omp pragma parallel for private(i,ip)
  for (ip=0; ip<PEsmpTOT; ip++){
  for (i=INDEX[ip]; i<INDEX[ip+1]; i++)
    W[Z][i]= W[R][i]*W[DD][i];
  }
  }
```

```
64*64*64
METHOD= 1
1      6.543963E+00
101    1.748392E-05
146    9.731945E-09

real    0m14.662s
```

```
METHOD= 3
1      6.299987E+00
101    1.298539E+00
201    2.725948E-02
301    3.664216E-05
401    2.146428E-08
413    9.621688E-09

real    0m19.660s
```

前処理はどうするか?

不完全修正コレスキー分解

```
for(i=0; i<N; i++) {
  VAL = D[i];
  for(j=indexL[i]; j<indexL[i+1]; j++) {
    VAL -= AL[j]*AL[j]*W[DD][itemL[j]-1];
  }
  W[DD][i] = 1.0 / VAL;
}
```

前進代入

```
for(i=0; i<N; i++) {
  WVAL = W[Z][i];
  for(j=indexL[i]; j<indexL[i+1]; j++) {
    WVAL -= AL[j] * W[Z][itemL[j]-1];
   }
   W[Z][i] = WVAL * W[DD][i];
}
```

データ依存性：メモリの読み込みと書き出しが同時に発生し，並列化困難

不完全修正コレスキー分解

```
for(i=0; i<N; i++) {
  VAL = D[i];
  for(j=indexL[i]; j<indexL[i+1]; j++) {
    VAL -= AL[j]*AL[j]*W[DD][itemL[j]-1];
  }
  W[DD][i]= 1.0 / VAL;
}
```

前進代入

```
for(i=0; i<N; i++) {
  WVAL = W[Z][i];
  for(j=indexL[i]; j<indexL[i+1]; j++) {
    WVAL -= AL[j] * W[Z][itemL[j]-1];
   }
   W[Z][i]= WVAL * W[DD][i];
}
```

Matrix-Vector Multiply: Easy to be Parallelized {q}=[A]{p}

{q}: LHS: Updated
{p}: RHS: No change

```
#pragma omp parallel for private(i,VAL,j)
 for(i=0; i<N; i++){
           VAL = D[i] * W[P][i];
           for(j=indexL[i]; j<indexL[i+1]; j++) {
               VAL += AL[j] * W[P][itemL[j]-1];
           }

           for(j=indexU[i]; j<indexU[i+1]; j++) {
               VAL += AU[j] * W[P][itemU[j]-1];
           }
           W[Q][i] = VAL;
 }
```

前進代入
4スレッドによる並列化を試みる

“icel” starts at 0

12	13	14	15
8	9	10	11
4	5	6	7
0	1	2	3

```
for(i=0; i<N; i++) {
  VAL = D[i];
  for(j=indexL[i]; j<indexL[i+1]; j++) {
    VAL -= AL[j]*AL[j]*W[DD][itemL[j]-1];
  }
  W[DD][i]= 1.0 / VAL;
}
```

“itemL” starts at 1

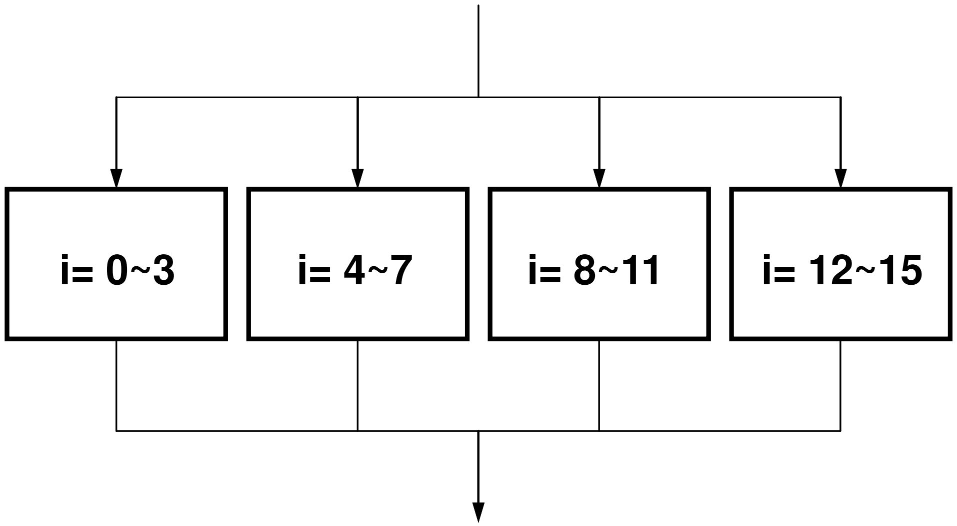
13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

前進代入
4スレッドによる並列化を試みる

12	13	14	15
8	9	10	11
4	5	6	7
0	1	2	3

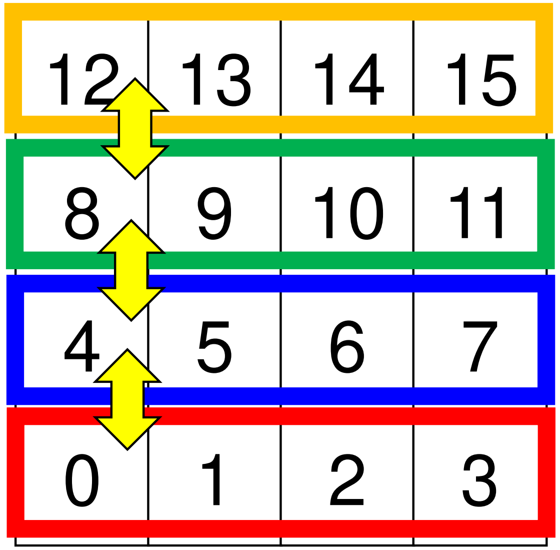
```
#pragma omp parallel private (ip,i,k,VAL)
for(ip=1; ip<4; ip++) {
for(i=INDEX[ip]; i<INDEX[ip+1];i++){
  VAL = D[i];
  for(j=indexL[i]; j<indexL[i+1]; j++) {
    VAL -= AL[j]*AL[j]*W[DD][itemL[j]-1];
  }
  W[DD][i]= 1.0 / VAL;
}
}
```

```
INDEX[0]= 0
INDEX[1]= 4
INDEX[2]= 8
INDEX[3]=12
INDEX[4]=16
```



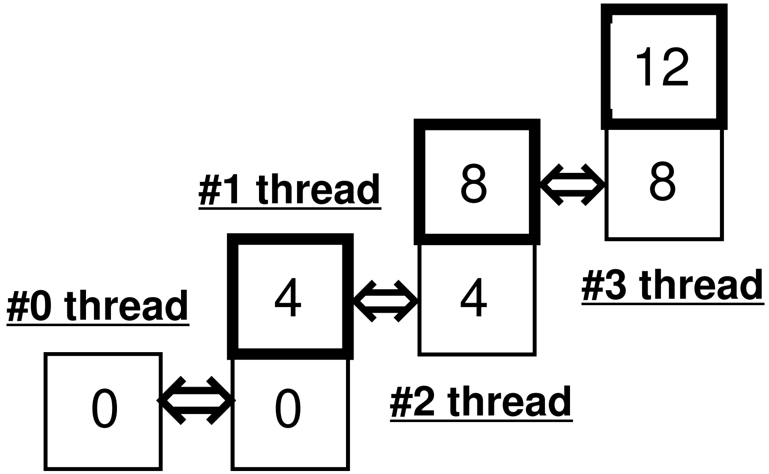
このような4スレッドが同時に実施される・・・

データ依存性：メモリへの書き出し，読み込みが同時に発生




```
#pragma omp parallel private (ip,i,k,VAL)
for(ip=1; ip<4; ip++) {
for(i=INDEX[ip]; i<INDEX[ip+1];i++){
  VAL = D[i];
  for(j=indexL[i]; j<indexL[i+1]; j++) {
    VAL -= AL[j]*AL[j]*W[DD][itemL[j]-1];
  }
  W[DD][i]= 1.0 / VAL;
}
}
```

```
INDEX[0]= 0
INDEX[1]= 4
INDEX[2]= 8
INDEX[3]=12
INDEX[4]=16
```



⇔の部分に
データ依存性発生

ICCG法の並列化

- 内積：OK
- DAXPY：OK
- 行列ベクトル積：OK
- 前処理：なんとかしなければならない
  - 単純にOpenMPなどの指示行（directive）を挿入しただけでは「並列化」できない。