

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 Fortran編

Part-A1: 概要, 対象アプリケーション

中島研吾

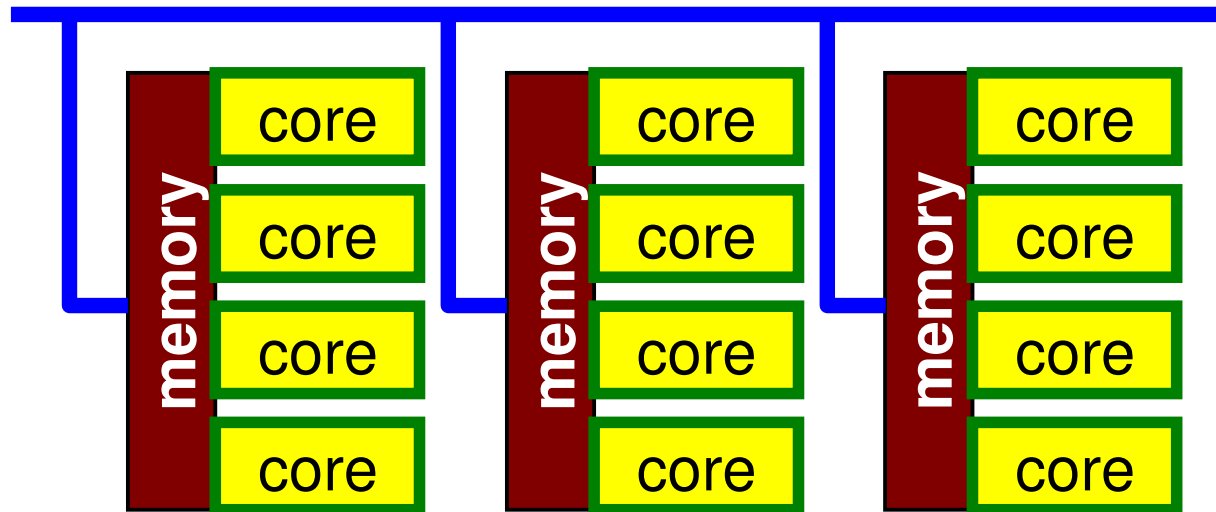
東京大学情報基盤センター

本編の背景

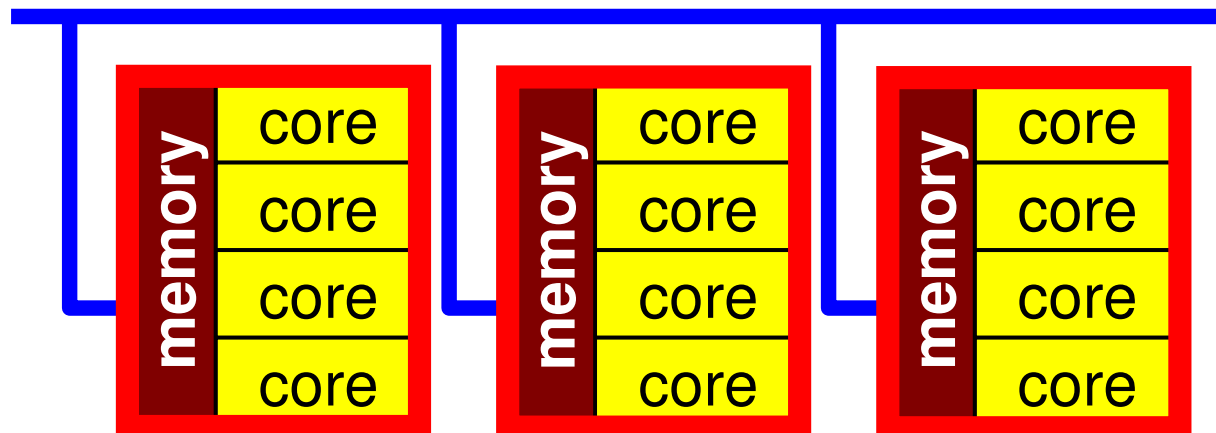
- マイクロプロセッサのマルチコア化, メニーコア化
 - 低消費電力, 様々なプログラミングモデル
- OpenMP
 - 指示行(ディレクティブ)を挿入するだけで手軽に「並列化」ができるため, 広く使用されている
 - 様々な解説書
- データ依存性 (data dependency)
 - メモリへの書き込みと参照が同時に発生
 - 並列化を実施するには, 適切なデータの並べ替えを施す必要がある
 - このような対策はOpenMP向けの解説書でも詳しく取り上げられることは余りない: ととても面倒くさい
- **Hybrid 並列プログラミングモデル**

Flat MPI vs. Hybrid

Flat-MPI: Each PE -> Independent



Hybrid: Hierarchical Structure



Key-Issues towards Appl./Algorithms on Exa-Scale Systems

Jack Dongarra (ORNL/U. Tennessee) at SIAM/PP10

- Hybrid/Heterogeneous Architecture
 - Multicore + GPU
 - Multicore + Manycore (more intelligent)
- Mixed Precision Computation
- Auto-Tuning/Self-Adapting
- Fault Tolerant
- Communication Reducing Algorithms

Hybrid並列プログラミングモデルは必須

- Message Passing
 - MPI
- Multi Threading
 - OpenMP, CUDA, OpenCL, OpenACC
 - OpenACC
 - GPUなどのアクセラレータのためのプログラミング環境として広く利用されており, OpenMPと同様に指示行挿入によってプログラミングが可能

本編の目的

- 「有限体積法から導かれる疎行列を対象としたPCG法」を題材とした，データ配置など，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
- 更に理解を深めるための，Wisteria/BDEC-01-Odyssey (A64FX) を利用した実習
- 単一のアプリケーションに特化した内容であるが，基本的な考え方は様々な分野に適用可能である
 - 実はこの方法は意外に効果的である
- 従来は2日間で(OpenMP + OpenACC)の内容であったが，オンライン化にあたって1日，OpenMPのみとした
⇒今回Wisteria/BDEC-01-Odysseyへの移行に従い，プログラムを大幅に簡略化

本編の目的(続き)

- 単一のアプリケーションに特化した内容であるが、基本的な考え方は様々な分野に適用可能である
 - 実はこの方法は意外に効果的である
- いわゆる「並列化講習会」とはだいぶ趣が異なる
- SMASH:「Science」無き科学技術計算はあり得ない！

Science

Modeling

Algorithm

Software

Hardware

スケジュール

- 09:00 - 12:00 有限体積法
- 13:00 - 14:00 Wisteria/BDEC-01ログイン
- 14:00 - 17:30 OpenMP並列化(実習含む)
- 17:30 - 18:00 質疑

ファイルの用意 on your PC

コピー, 展開

<http://nkl.cc.u-tokyo.ac.jp/files/fvm.tar>

```
>$ cd
```

```
>$ tar xvf fvm.tar
```

```
>$ cd fvm
```

以下のディレクトリが出来ていることを確認

`src-c, src-f, run`

これらを以降 `<$P-FVM>`

Your PC

Odyssey

可視化にはParaViewを使用

<http://www.paraview.org/>

フリーソフトウェア
Windows版, Mac版がある
UNIX版もあり

<http://nkl.cc.u-tokyo.ac.jp/class/HowtouseParaView.pdf>

資料はWeb上にもあります

<http://nkl.cc.u-tokyo.ac.jp/seminars/multicore2021/>

- 背景
 - 有限体積法
 - 前処理付反復法
- PCG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
 - PCG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PCG法」を題材とした，データ配置など，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
- PCG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 @ $Z=Z_{\max}$, 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 (Preconditioning) \Rightarrow PCG法

解いている問題：三次元ポアソン方程式

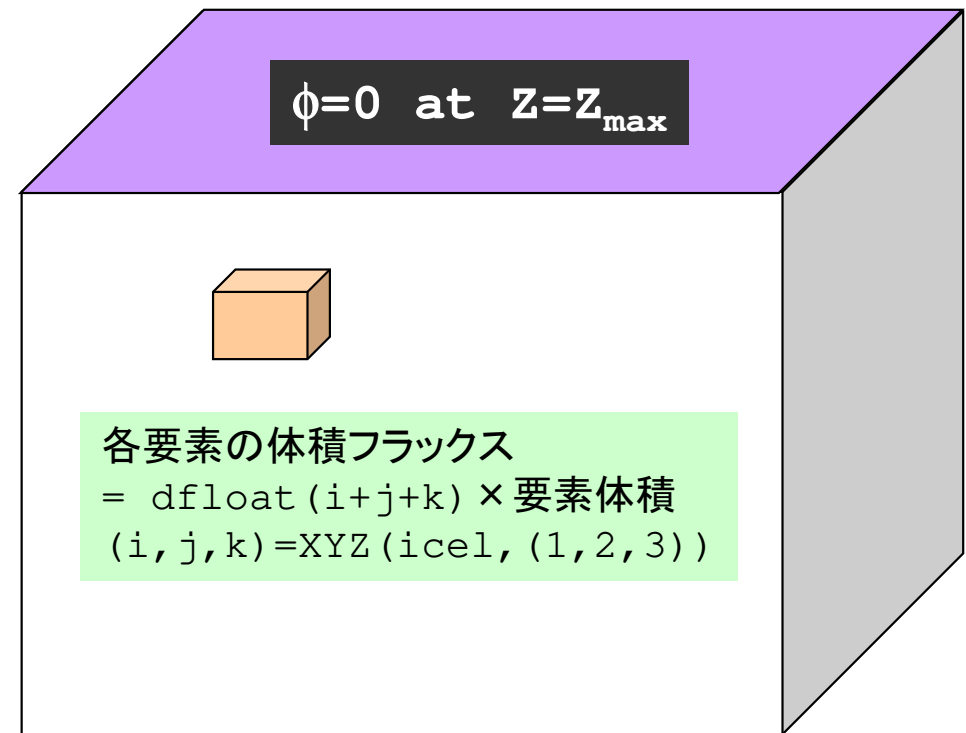
変数：要素中心で定義

ポアソン方程式

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

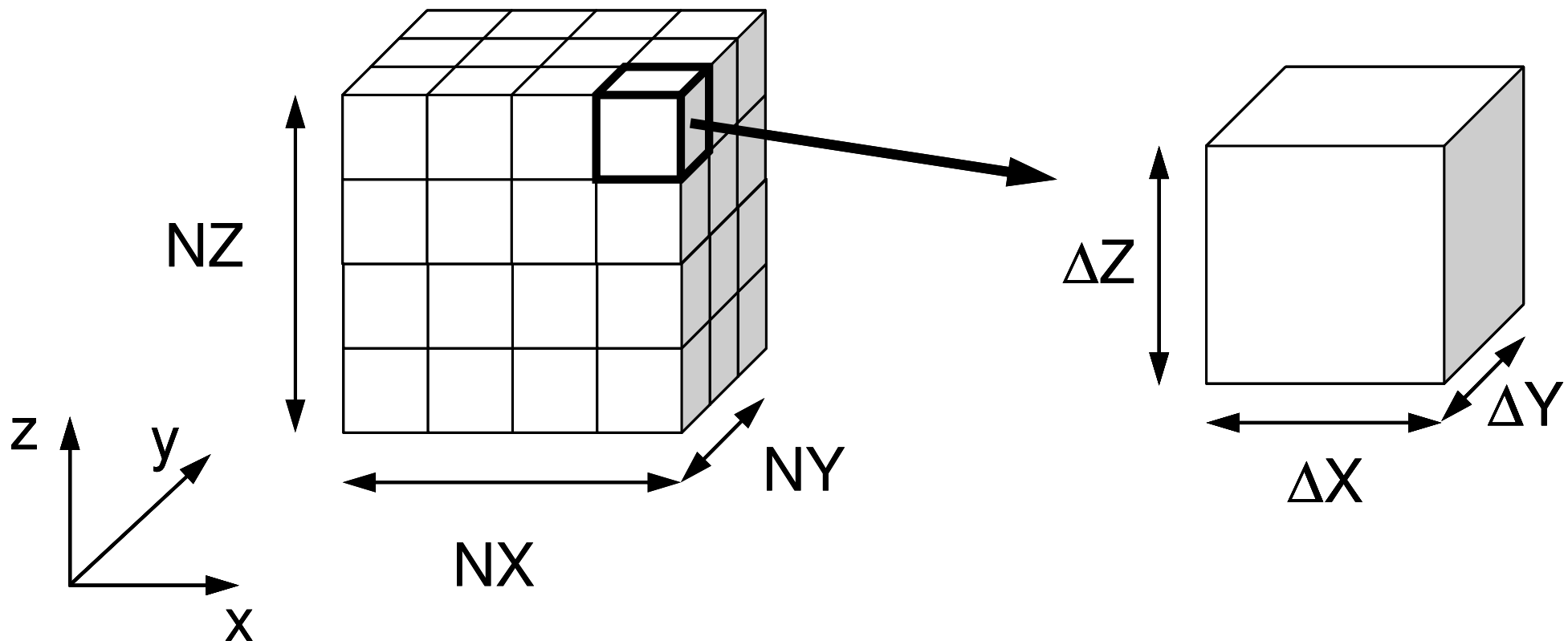
境界条件他

- 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

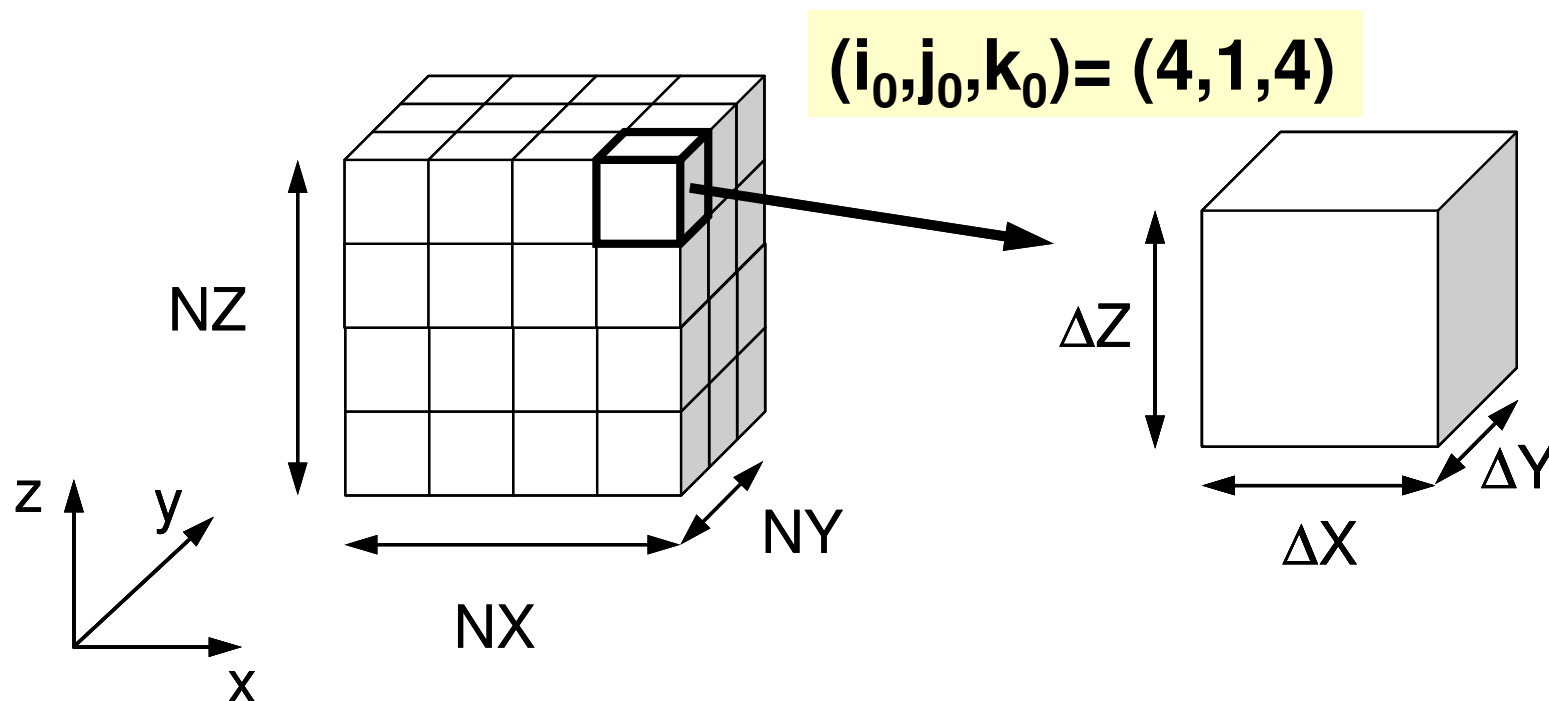
半非構造的に扱う



体積フラックスfの内容 $\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} + f = 0$

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

$i_0 = XYZ(icel, 1)$, $XYZ(icel, k)$ ($k=1,2,3$) は
 $j_0 = XYZ(icel, 2)$, X, Y, Z方向の差分格子のインデックス
 $k_0 = XYZ(icel, 3)$ 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。



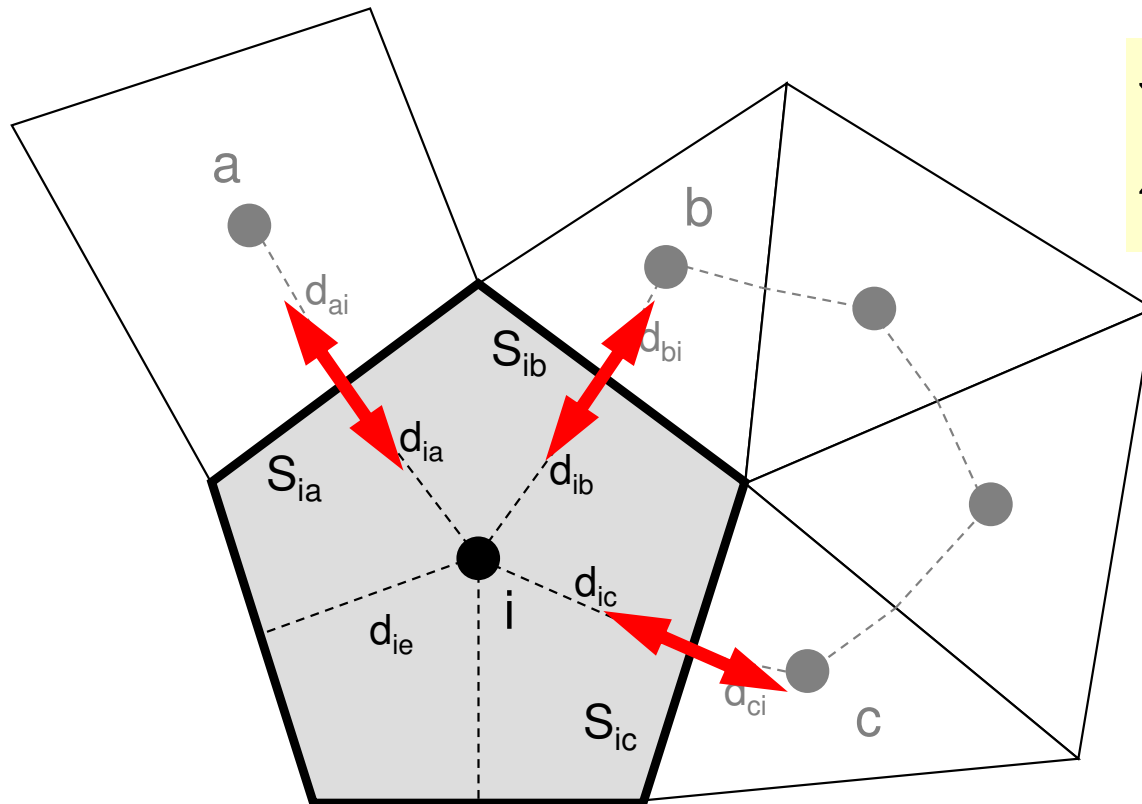
ポアソン方程式:

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

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

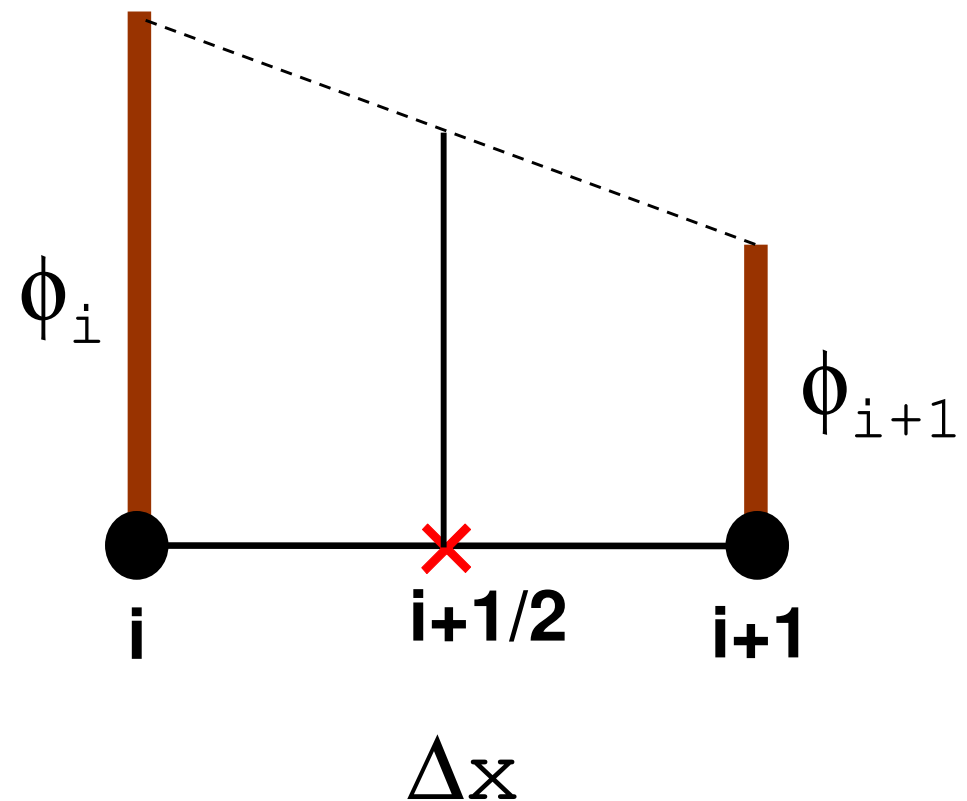
体積
フラックス

- 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

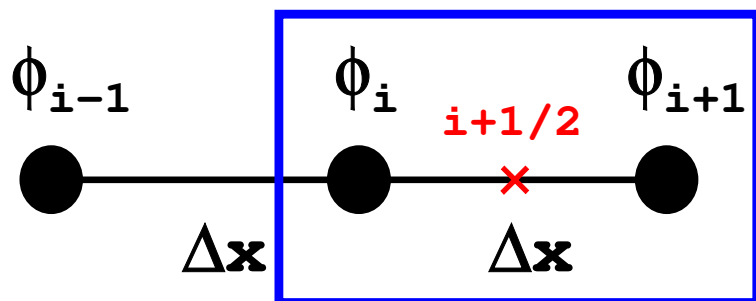
$$\left(\frac{d\phi}{dx}\right)_{i+1/2} \approx \frac{\phi_{i+1} - \phi_i}{\Delta x}$$
$$\left(\frac{d\phi}{dx}\right)_{i+1/2} = \lim_{\Delta x \rightarrow 0} \frac{\phi_{i+1} - \phi_i}{\Delta x}$$



差分法における二次微分係数・導関数

Taylor Series Expansion: Taylor展開

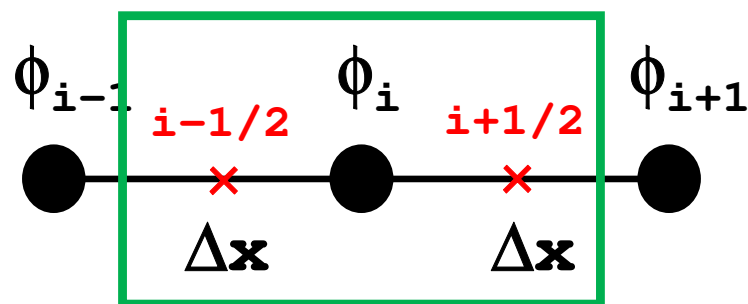
- 点 \times における「傾き」(中点: $i - i+1$)



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

$\Delta x \rightarrow 0$: 本当の微分係数

- 点 i における二次微分係数



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

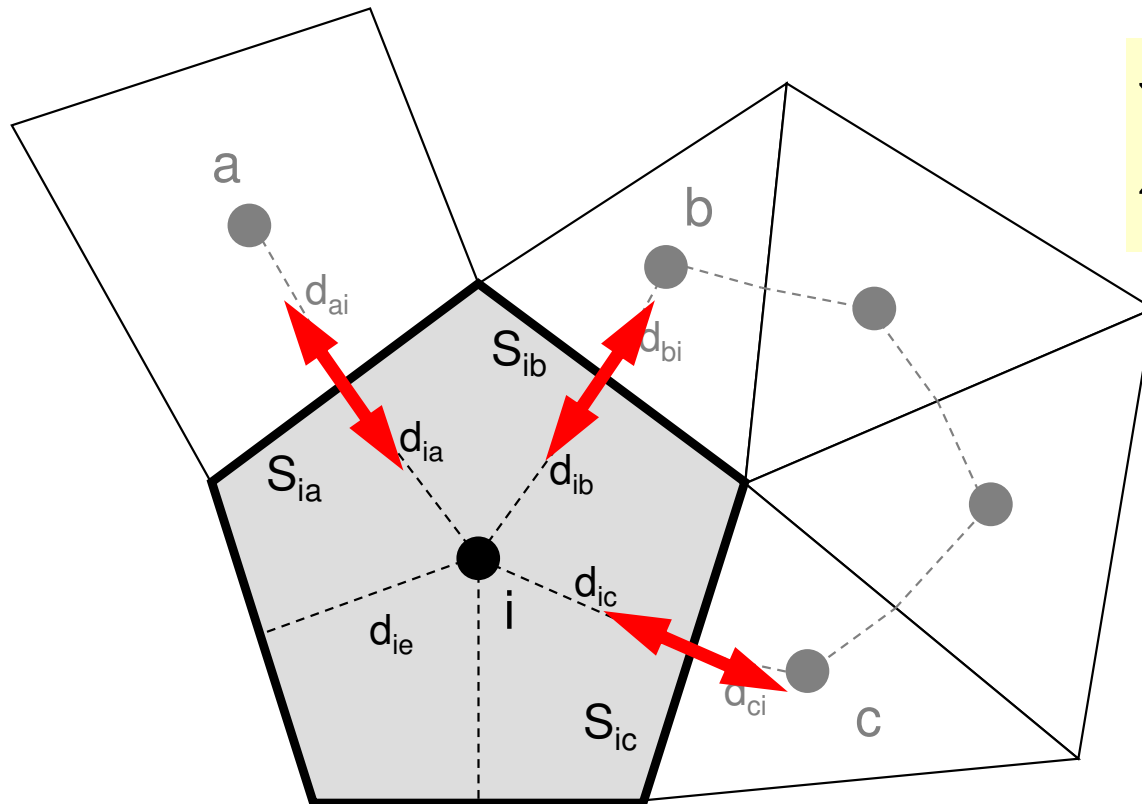
ポアソン方程式:

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



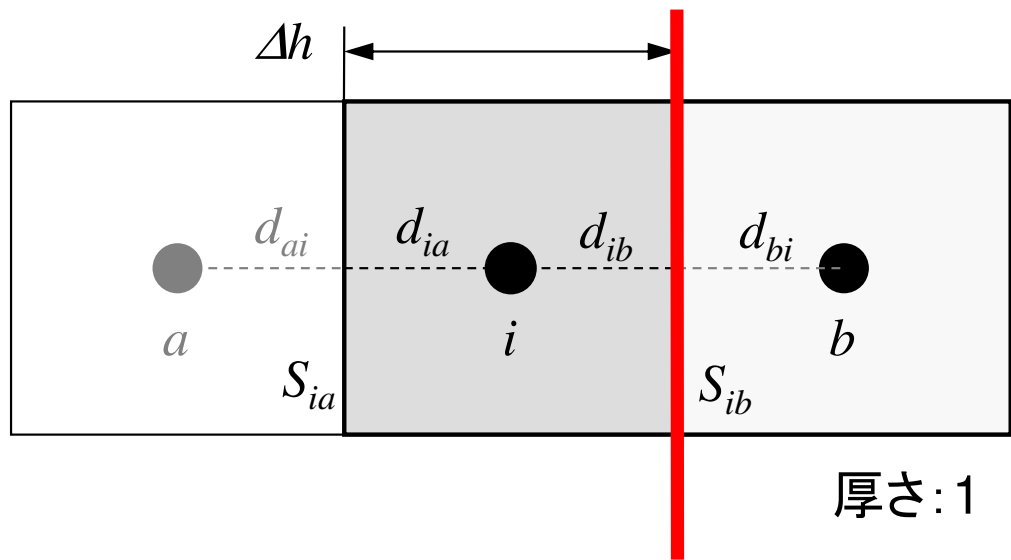
隣接要素との拡散

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

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: $d_{ij} = \Delta h/2$

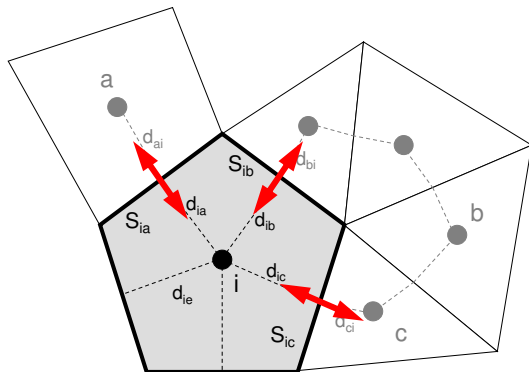
この面を通過するフラックス: $Q_{S_{ib}}$

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

フーリエ (Fourier) の法則

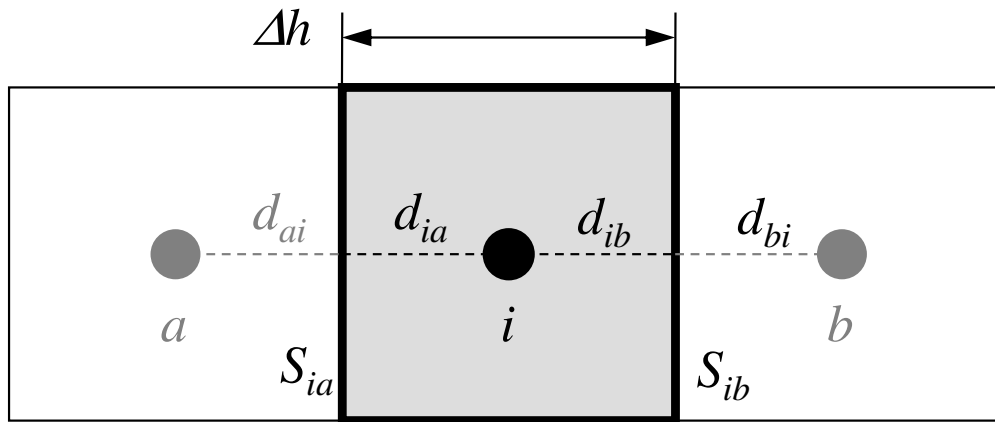
面を通過するフラックス (流束)

= - (ポテンシャル勾配)



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

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: $d_{ij} = \Delta h/2$

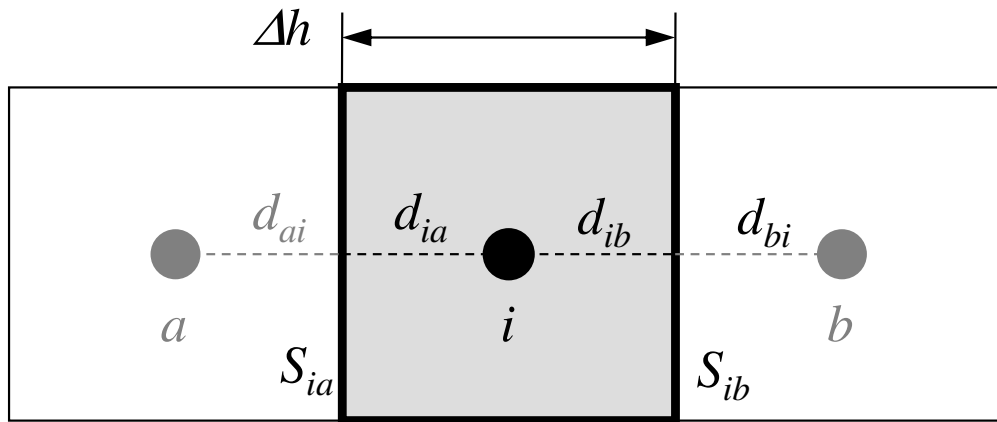
$$\sum_k \frac{S_{ik}}{d_{ik} + d_{ki}} (\phi_k - \phi_i) + V_i \dot{Q}_i = 0$$

両辺を V_i で割る:

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

この部分に注目すると

一次元差分法との比較 (3/3)



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: $d_{ij} = \Delta h/2$

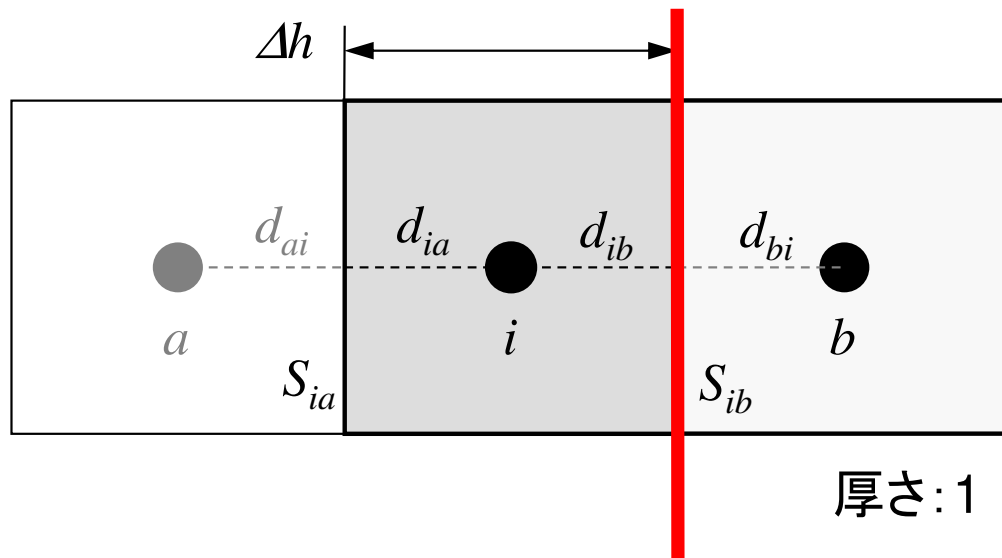
$$\begin{aligned} \frac{1}{V_i} \sum_k \frac{S_{ik}}{d_{ik} + d_{ki}} (\phi_k - \phi_i) &= \frac{1}{(\Delta h)^2} \sum_{k=a,b} \frac{\Delta h}{\frac{\Delta h}{2} + \frac{\Delta h}{2}} (\phi_k - \phi_i) \\ &= \frac{1}{(\Delta h)^2} \sum_{k=a,b} \frac{\Delta h}{\frac{\Delta h}{2} + \frac{\Delta h}{2}}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

要素*i*について成立
連立一次方程式

熱伝導の場合 (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 : 熱伝導率



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: $d_{ij} = \Delta h/2$

この面を通過する熱フラックス: $Q_{S_{ib}}$

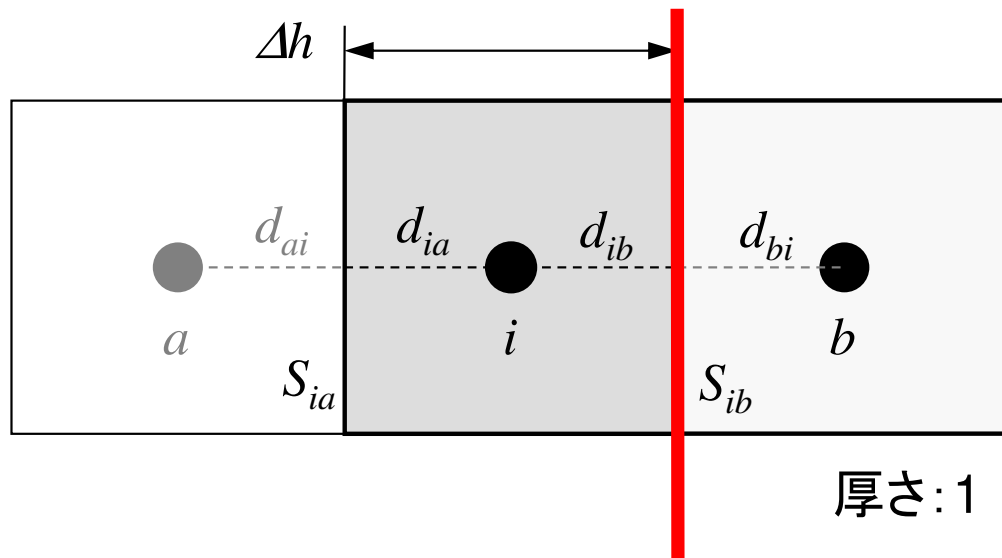
$$Q_{S_{ib}} = -\lambda \frac{T_b - T_i}{\Delta h} \cdot S_{ib}$$

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

熱伝導の場合 (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 : 熱伝導率



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: $d_{ij} = \Delta h/2$

この面を通過する熱フラックス: $Q_{S_{ib}}$

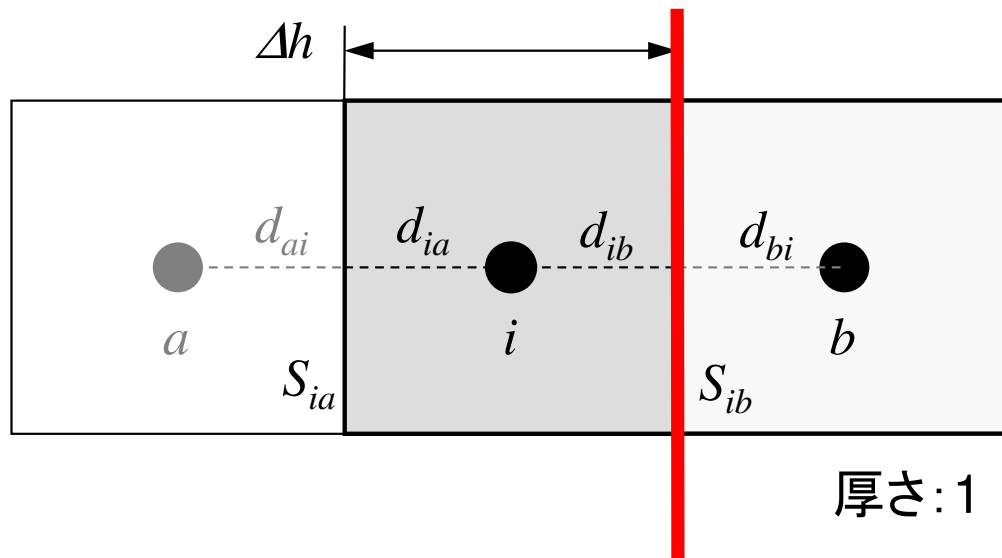
$$Q_{S_{ib}} = -\lambda \frac{T_b - T_i}{\frac{\Delta h}{2} + \frac{\Delta h}{2}} \cdot S_{ib} = -\frac{T_b - T_i}{\left[\left(\frac{\Delta h}{2} \right) / \lambda \right] + \left[\left(\frac{\Delta h}{2} \right) / \lambda \right]} \cdot S_{ib}$$

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

熱伝導の場合 (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 : 熱伝導率



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: $V_i = \Delta h^2$

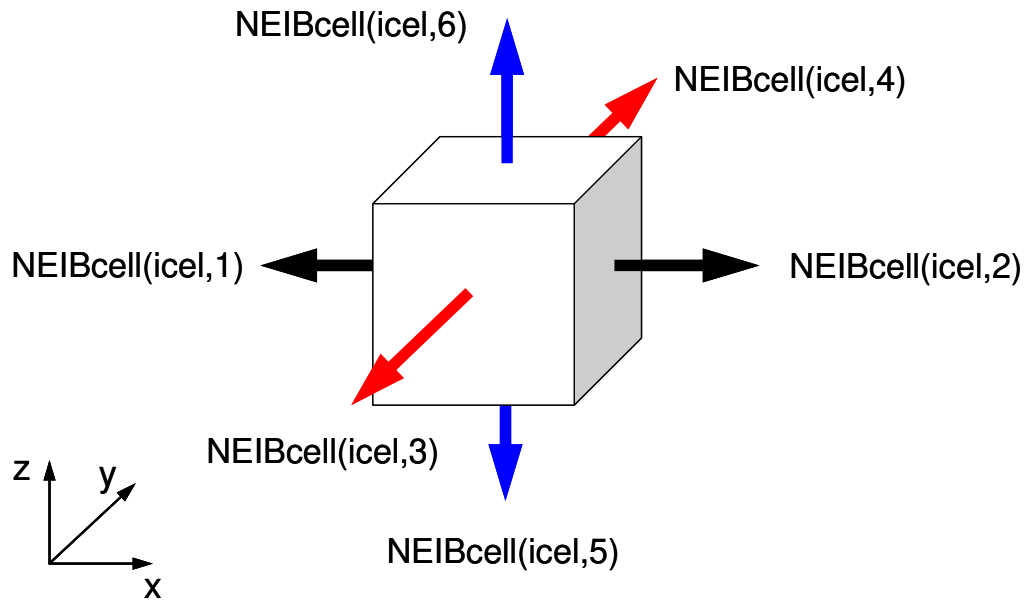
接触面までの距離: $d_{ij} = \Delta h/2$

この面を通過する熱フラックス: $Q_{S_{ib}}$

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

$$\lambda_i \neq \lambda_b$$

三次元では・・・



$$\begin{aligned}
 & \frac{\phi_{neib(icel,1)} - \phi_{icel}}{\Delta x} \Delta y \Delta z + \frac{\phi_{neib(icel,2)} - \phi_{icel}}{\Delta x} \Delta y \Delta z + \\
 & \frac{\phi_{neib(icel,3)} - \phi_{icel}}{\Delta y} \Delta z \Delta x + \frac{\phi_{neib(icel,4)} - \phi_{icel}}{\Delta y} \Delta z \Delta x + \\
 & \frac{\phi_{neib(icel,5)} - \phi_{icel}}{\Delta z} \Delta x \Delta y + \frac{\phi_{neib(icel,6)} - \phi_{icel}}{\Delta z} \Delta x \Delta y + f_{icel} \Delta x \Delta y \Delta z = 0
 \end{aligned}$$

整理すると: 連立一次方程式

$$\begin{aligned} & \frac{\phi_{neib(icel,1)} - \phi_{icel}}{\Delta x} \Delta y \Delta z + \frac{\phi_{neib(icel,2)} - \phi_{icel}}{\Delta x} \Delta y \Delta z + \\ & \frac{\phi_{neib(icel,3)} - \phi_{icel}}{\Delta y} \Delta z \Delta x + \frac{\phi_{neib(icel,4)} - \phi_{icel}}{\Delta y} \Delta z \Delta x + \\ & \frac{\phi_{neib(icel,5)} - \phi_{icel}}{\Delta z} \Delta x \Delta y + \frac{\phi_{neib(icel,6)} - \phi_{icel}}{\Delta z} \Delta x \Delta y + f_{icel} \Delta x \Delta y \Delta z = 0 \end{aligned}$$

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

$$-\left[\sum_k \frac{S_{icel-k}}{d_{icel-k}} \right] \phi_{icel} + \left[\sum_k \frac{S_{icel-k}}{d_{icel-k}} \phi_k \right] = -f_{icel} V_{icel} \quad (icel = 1, N)$$

対角項

非対角項

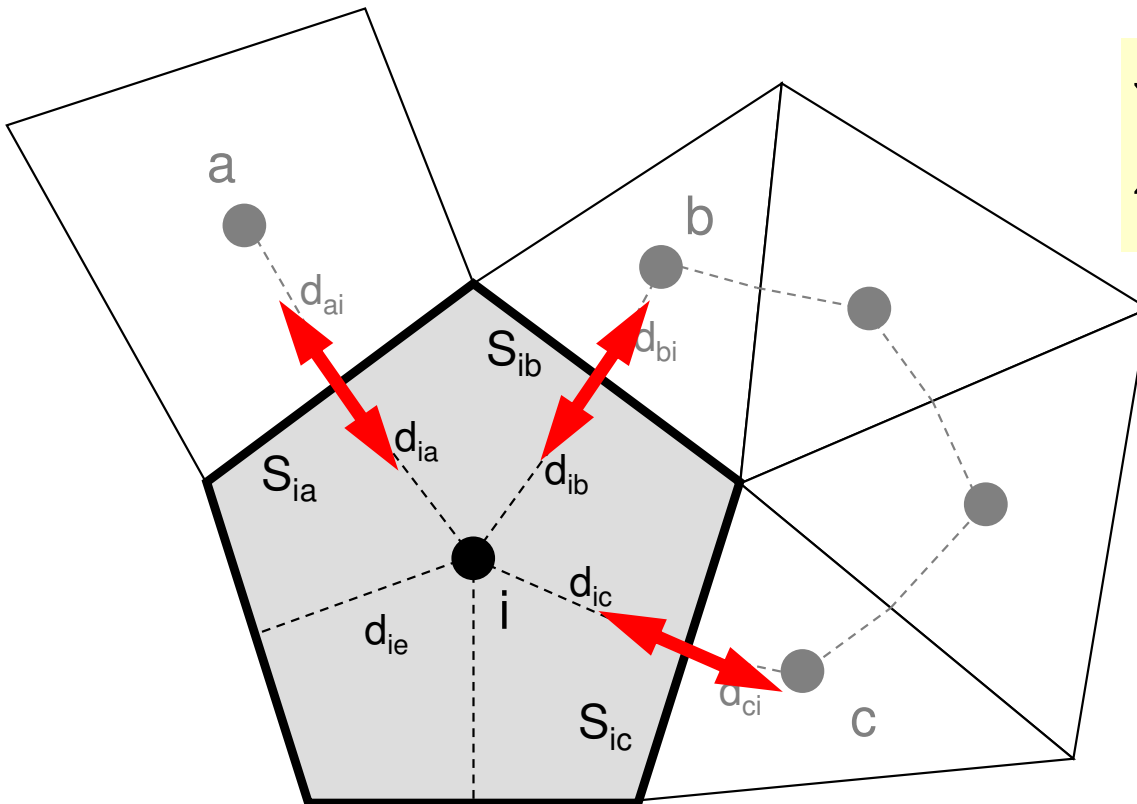


$$[A]\{\phi\} = \{f\}$$

FVMの係数行列も疎行列

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

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



隣接要素との拡散

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

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

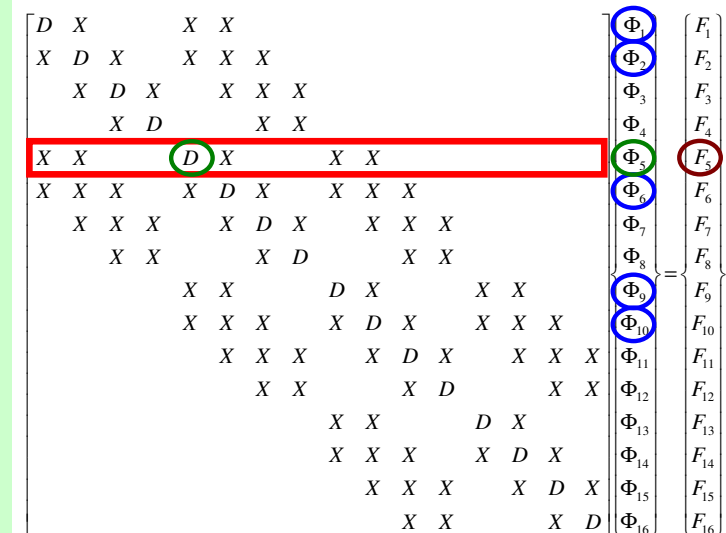
行列ベクトル積への適用

(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) 非零非対角成分に関する一次元配列(通し番号)
 (整数, $i=0, N$)
Item (k) 非零非対角成分の要素(列)番号
 (整数, $k=1, \text{index}(N)$)
AMat (k) 非零非対角成分
 (実数, $k=1, \text{index}(N)$)

$$\{Y\} = [A] \{X\}$$

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



行列ベクトル積：密行列⇒とても簡単

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

$$\{Y\} = [A] \{X\}$$

```

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

```

Compressed Row Storage (CRS)

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

Compressed Row Storage (CRS)

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

NODE_tot= 8

対角成分

$$D(1) = 1.1$$

$$D(2) = 3.6$$

$$D(3) = 5.7$$

$$D(4) = 9.8$$

$$D(5) = 11.5$$

$$D(6) = 12.4$$

$$D(7) = 23.1$$

$$D(8) = 51.3$$

Compressed Row Storage (CRS)

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

Compressed Row Storage (CRS)

	非対角成分数						
①	1.1 ①	2.4 ②	3.2 ⑤			2	$\text{index}(0) = 0$
②	3.6 ②	4.3 ①	2.5 ④	3.7 ⑥	9.1 ⑧	4	$\text{index}(1) = 2$
③	5.7 ③	1.5 ⑤	3.1 ⑦			2	$\text{index}(2) = 6$
④	9.8 ④	4.1 ②	2.5 ⑤	2.7 ⑥		3	$\text{index}(3) = 8$
⑤	11.5 ⑤	3.1 ①	9.5 ②	10.4 ③	4.3 ⑦	4	$\text{index}(4) = 11$
⑥	12.4 ⑥	6.5 ③	9.5 ⑦			2	$\text{index}(5) = 15$
⑦	23.1 ⑦	6.4 ②	2.5 ③	1.4 ⑥	13.1 ⑧	4	$\text{index}(6) = 17$
⑧	51.3 ⑧	9.5 ②	1.3 ③	9.6 ④	3.1 ⑥	4	$\text{index}(7) = 21$
							$\text{index}(8) = 25$

NPLU= 25
(=index(N))

$\text{index}(i-1)+1 \sim \text{index}(i)$ 番目が i 行目の非対角成分

Compressed Row Storage (CRS)

	非対角成分						
①	1.1 ①	2.4 ②,1	3.2 ⑤,2			2	$\text{index}(0) = 0$
②	3.6 ②	4.3 ①,3	2.5 ④,4	3.7 ⑥,5	9.1 ⑧,6	4	$\text{index}(1) = 2$
③	5.7 ③	1.5 ⑤,7	3.1 ⑦,8			2	$\text{index}(2) = 6$
④	9.8 ④	4.1 ②,9	2.5 ⑤,10	2.7 ⑥,11		3	$\text{index}(3) = 8$
⑤	11.5 ⑤	3.1 ①,12	9.5 ②,13	10.4 ③,14	4.3 ⑦,15	4	$\text{index}(4) = 11$
⑥	12.4 ⑥	6.5 ③,16	9.5 ⑦,17			2	$\text{index}(5) = 15$
⑦	23.1 ⑦	6.4 ②,18	2.5 ③,19	1.4 ⑥,20	13.1 ⑧,21	4	$\text{index}(6) = 17$
⑧	51.3 ⑧	9.5 ②,22	1.3 ③,23	9.6 ④,24	3.1 ⑥,25	4	$\text{index}(7) = 21$
							$\text{index}(8) = 25$

NPLU= 25
(=index(N))

$\text{index}(i-1)+1 \sim \text{index}(i)$ 番目が i 行目の非対角成分

Compressed Row Storage (CRS)

1	1.1 ①	2.4 ②,1	3.2 ⑤,2		
2	3.6 ②	4.3 ①,3	2.5 ④,4	3.7 ⑥,5	9.1 ⑧,6
3	5.7 ③	1.5 ⑤,7	3.1 ⑦,8		
4	9.8 ④	4.1 ②,9	2.5 ⑤,10	2.7 ⑥,11	
5	11.5 ⑤	3.1 ①,12	9.5 ②,13	10.4 ③,14	4.3 ⑦,15
6	12.4 ⑥	6.5 ③,16	9.5 ⑦,17		
7	23.1 ⑦	6.4 ②,18	2.5 ③,19	1.4 ⑥,20	13.1 ⑧,21
8	51.3 ⑧	9.5 ②,22	1.3 ③,23	9.6 ④,24	3.1 ⑥,25

例:

`item(7) = 5, AMAT(7) = 1.5`

`item(19) = 3, AMAT(19) = 2.5`

Compressed Row Storage (CRS)

1	1.1 ①	2.4 ②,1	3.2 ⑤,2		
2	3.6 ②	4.3 ①,3	2.5 ④,4	3.7 ⑥,5	9.1 ⑧,6
3	5.7 ③	1.5 ⑤,7	3.1 ⑦,8		
4	9.8 ④	4.1 ②,9	2.5 ⑤,10	2.7 ⑥,11	
5	11.5 ⑤	3.1 ①,12	9.5 ②,13	10.4 ③,14	4.3 ⑦,15
6	12.4 ⑥	6.5 ③,16	9.5 ⑦,17		
7	23.1 ⑦	6.4 ②,18	2.5 ③,19	1.4 ⑥,20	13.1 ⑧,21
8	51.3 ⑧	9.5 ②,22	1.3 ③,23	9.6 ④,24	3.1 ⑥,25

D (i) 対角成分(実数, $i=1, N$)
index(i) 非零非対角成分に関する
 一次元配列(通し番号)
 (整数, $i=0, N$)
item(k) 非零非対角成分の要素(列)番号
 (整数, $k=1, \text{index}(N)$)
AMAT(k) 非零非対角成分
 (実数, $k=1, \text{index}(N)$)

$$\{Y\} = [A] \{X\}$$

```

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

疎行列：非零成分のみ記憶
⇒メモリへの負担大
(**memory-bound**): 間接参照
(差分, **FEM**, **FVM**)

$$\{Y\} = [A] \{X\}$$

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

行列ベクトル積：密行列⇒とても簡単 メモリへの負担も小さい

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

$$\{Y\} = [A] \{X\}$$

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

- 背景
 - 有限体積法
 - 前処理付反復法
- PCG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
 - PCG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

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

A **x** **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 $\mathbf{x}^{(0)}$ 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

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

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 $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k$ を求める:

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

$$= \mathbf{r}_{k-1} + \mathbf{x}_{k-1} \quad \text{where } \mathbf{r}_k = \mathbf{b} - \mathbf{Ax}_k : \text{残差ベクトル (residual)}$$



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

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

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}_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}_k を求めるかにある:

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

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
 - $\mathbf{x}^{(i)} = \mathbf{x}^{(i-1)} + \alpha_i \mathbf{p}^{(i)}$
 - $\mathbf{x}^{(i)}$: 反復解, $\mathbf{p}^{(i)}$: 探索方向, α_i : 定数
 - 厳密解を \mathbf{y} とするとき $\{\mathbf{x}-\mathbf{y}\}^T[A]\{\mathbf{x}-\mathbf{y}\}$ 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)

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

共役勾配法 (CG法) のアルゴリズム

```

Compute  $r^{(0)} = b - [A]x^{(0)}$ 
for  $i = 1, 2, \dots$ 
   $\rho_{i-1} = r^{(i-1)} \cdot r^{(i-1)}$ 
  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
   $q^{(i)} = [A]p^{(i)}$ 
   $\alpha_i = \rho_{i-1} / p^{(i)} \cdot q^{(i)}$ 
   $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$ 
   $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$ 
  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  $i = 1, 2, \dots$ 
   $\rho_{i-1} = r^{(i-1)} \cdot r^{(i-1)}$ 
  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
   $q^{(i)} = [A]p^{(i)}$ 
   $\alpha_i = \rho_{i-1} / p^{(i)} \cdot q^{(i)}$ 
   $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$ 
   $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$ 
  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  $i = 1, 2, \dots$ 
   $\rho_{i-1} = r^{(i-1)} \cdot r^{(i-1)}$ 
  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
   $q^{(i)} = [A]p^{(i)}$ 
   $\alpha_i = \rho_{i-1} / p^{(i)} \cdot q^{(i)}$ 
   $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$ 
   $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$ 
  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  $i = 1, 2, \dots$ 
   $\rho_{i-1} = r^{(i-1)} \cdot r^{(i-1)}$ 
  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
   $q^{(i)} = [A]p^{(i)}$ 
   $\alpha_i = \rho_{i-1} / p^{(i)} \cdot q^{(i)}$ 
   $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$ 
   $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$ 
  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
 - $a\{x\} + \{y\}$

$x^{(i)}$: 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  $i = 1, 2, \dots$ 
   $\rho_{i-1} = r^{(i-1)} \cdot r^{(i-1)}$ 
  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
   $q^{(i)} = [A]p^{(i)}$ 
   $\alpha_i = \rho_{i-1} / p^{(i)} \cdot q^{(i)}$ 
   $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$ 
   $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$ 
  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:

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

$$\begin{aligned} (x-y)^T [A](x-y) &= (x, Ax) - (y, Ax) - (x, Ay) + (y, Ay) \\ &= (x, Ax) - 2(x, Ay) + (y, Ay) = (x, Ax) - 2(x, b) + \underline{(y, b)} \quad \text{定数} \end{aligned}$$

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

$$\begin{aligned} f(x+h) &= \frac{1}{2}(x+h, A(x+h)) - (x+h, b) \\ &= \frac{1}{2}(x+h, Ax) + \frac{1}{2}(x+h, Ah) - (x, b) - (h, b) \\ &= \frac{1}{2}(x, Ax) + \frac{1}{2}(h, Ax) + \frac{1}{2}(x, Ah) + \frac{1}{2}(h, Ah) - (x, b) - (h, b) \\ &= \frac{1}{2}(x, Ax) - (x, b) + (h, Ax) - (h, b) + \frac{1}{2}(h, Ah) \\ &= f(x) + (h, Ax - b) + \frac{1}{2}(h, Ah) \end{aligned}$$

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:

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

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

$r^{(k)} = b - Ax^{(k)}$ 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:

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

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

探索方向を以下の漸化式によって求める:

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

本当のところは下記のように(k+1)回目に厳密解 y が求め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:

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

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

$$\therefore \alpha_k = \frac{(p^{(k)}, r^{(k)})}{(p^{(k)}, Ap^{(k)})}$$

従って以下が成立する:

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

CG法アルゴリズムの導出(5/5)

$$\begin{aligned} (p^{(k+1)}, Ap^{(k)}) &= (r^{(k+1)} + \beta_k p^{(k)}, Ap^{(k)}) = (r^{(k+1)}, Ap^{(k)}) + \beta_k (p^{(k)}, Ap^{(k)}) = 0 \\ \Rightarrow \beta_k &= \frac{-(r^{(k+1)}, Ap^{(k)})}{(p^{(k)}, Ap^{(k)})} \quad \underline{(4)} \end{aligned}$$

$(p^{(k+1)}, Ap^{(k)}) = 0$ $p^{(k)}$ 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  $i = 1, 2, \dots$ 
  calc.  $\alpha_{i-1}$ 
   $x^{(i)} = x^{(i-1)} + \alpha_{i-1}p^{(i-1)}$ 
   $r^{(i)} = r^{(i-1)} - \alpha_{i-1}[A]p^{(i-1)}$ 

  check convergence  $|r|$ 
  (if not converged)
  calc.  $\beta_{i-1}$ 
   $p^{(i)} = r^{(i)} + \beta_{i-1}p^{(i-1)}$ 
end
  
```

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

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

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:

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

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:

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

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 \Rightarrow 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

$$(r^{(i)}, r^{(j)}) = 0 \quad (i \neq j)$$

直交性

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

共役性

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

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

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

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

Proof (2/3)

Mathematical Induction

数学的帰納法

$$\begin{aligned} (r^{(i)}, r^{(j)}) &= 0 \quad (i \neq j) \\ (p^{(i)}, Ap^{(j)}) &= 0 \quad (i \neq j) \end{aligned} \quad (*)$$

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

$$\begin{aligned} \text{if } i < k \quad (r^{(k+1)}, r^{(i)}) &= (r^{(i)}, r^{(k+1)}) \stackrel{(2)}{=} (r^{(i)}, r^{(k)} - \alpha_k Ap^{(k)}) \\ &\stackrel{(*)}{=} -\alpha_k (r^{(i)}, Ap^{(k)}) \stackrel{(3)}{=} -\alpha_k (p^{(i)} - \beta_{i-1} p^{(i-1)}, Ap^{(k)}) \\ &= -\alpha_k (p^{(i)}, Ap^{(k)}) + \alpha_k \beta_{i-1} (p^{(i-1)}, Ap^{(k)}) \stackrel{(*)}{=} 0 \end{aligned}$$

$$\begin{aligned} \text{if } i = k \quad (r^{(k+1)}, r^{(k)}) &\stackrel{(2)}{=} (r^{(k)}, r^{(k)}) - (r^{(k)}, \alpha_k Ap^{(k)}) \\ &\stackrel{(3)}{=} (r^{(k)}, r^{(k)}) - (p^{(k)} - \beta_{k-1} p^{(k-1)}, \alpha_k Ap^{(k)}) \\ &\stackrel{(*)}{=} (r^{(k)}, r^{(k)}) - \alpha_k (p^{(k)}, Ap^{(k)}) \stackrel{(1)}{=} (r^{(k)}, r^{(k)}) - (p^{(k)}, r^{(k)}) \\ &\stackrel{(3)}{=} (r^{(k)}, r^{(k)}) - (\beta_{k-1} p^{(k-1)} + r^{(k)}, r^{(k)}) \\ &= -\beta_{k-1} (p^{(k-1)}, r^{(k)}) \stackrel{(2)}{=} -\beta_{k-1} (p^{(k-1)}, r^{(k-1)} - \alpha_{k-1} Ap^{(k-1)}) \\ &= -\beta_{k-1} \left\{ (p^{(k-1)}, r^{(k-1)}) - \alpha_{k-1} (p^{(k-1)}, Ap^{(k-1)}) \right\} \stackrel{(1)}{=} 0 \end{aligned}$$

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

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

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

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

Proof (3/3)

Mathematical Induction

数学的帰納法

$$\begin{aligned} (r^{(i)}, r^{(j)}) &= 0 \quad (i \neq j) \\ (p^{(i)}, Ap^{(j)}) &= 0 \quad (i \neq j) \end{aligned} \quad (*)$$

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

$$\begin{aligned} \text{if } i < k \quad (p^{(k+1)}, Ap^{(i)}) &\stackrel{(3)}{=} (r^{(k+1)} + \beta_k p^{(k)}, Ap^{(i)}) \\ &\stackrel{(*)}{=} (r^{(k+1)}, Ap^{(i)}) \\ &\stackrel{(2)}{=} \frac{1}{\alpha_i} (r^{(k+1)}, r^{(i)} - r^{(i+1)}) = 0 \end{aligned}$$

$$\begin{aligned} \text{if } i = k \quad (p^{(k+1)}, Ap^{(k)}) &\stackrel{(3)}{=} (r^{(k+1)}, Ap^{(k)}) + \beta_k (p^{(k)}, Ap^{(k)}) \\ &\stackrel{(4)}{=} 0 \end{aligned}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

α_k, β_k

実際は α_k, β_k はもうちょっと簡単な形に変形できる:

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

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

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

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

共役勾配法 (CG法) のアルゴリズム

```

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

```

$x^{(i)}$: Vector

α_i : Scalar

$$\beta_{i-1} = \frac{\left(r^{(i-1)}, r^{(i-1)} \right)}{\left(r^{(i-2)}, r^{(i-2)} \right)} \quad \left(= \rho_{i-1} \right)$$

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

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: PCG

Preconditioned Conjugate Gradient Method (PCG)

```

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

```

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

$$[A']x' = b'$$

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

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

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

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

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

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

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

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

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

CG法では通常, $[M_2] = [M_1]^T$ である(例: 不完全コレスキー分解)
 従って $[M_1]$ と $[M_2]$ を以下のように定義する:

$$[M_1] = [X]^T, [M_2] = [X], [M] = [M_1][M_2]$$

$$[A']x' = b'$$

$$[A'] = [M_1]^{-1}[A][M_2]^{-1} = [[X]^T]^{-1}[A][X]^{-1} = [X]^{-T}[A][X]^{-1}$$

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

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

$$\begin{aligned}
\beta'_{i-1} &= \frac{\left(r^{(i-1)}, r^{(i-1)} \right)}{\left(r^{(i-2)}, r^{(i-2)} \right)} = \frac{\left([\mathbf{X}]^{-T} r^{(i-1)}, [\mathbf{X}]^{-T} r^{(i-1)} \right)}{\left([\mathbf{X}]^{-T} r^{(i-2)}, [\mathbf{X}]^{-T} r^{(i-2)} \right)} \\
&= \frac{\left(\left([\mathbf{X}]^{-T} r^{(i-1)} \right)^T, [\mathbf{X}]^{-T} r^{(i-1)} \right)}{\left(\left([\mathbf{X}]^{-T} r^{(i-2)} \right)^T, [\mathbf{X}]^{-T} r^{(i-2)} \right)} = \frac{\left(\left(r^{(i-1)} \right)^T [\mathbf{X}]^{-1}, [\mathbf{X}^T]^{-1} r^{(i-1)} \right)}{\left(\left(r^{(i-2)} \right)^T [\mathbf{X}]^{-1}, [\mathbf{X}^T]^{-1} r^{(i-2)} \right)} \\
&= \frac{\left(r^{(i-1)}, \left[[\mathbf{X}^T] [\mathbf{X}] \right]^{-1} r^{(i-1)} \right)}{\left(r^{(i-2)}, \left[[\mathbf{X}^T] [\mathbf{X}] \right]^{-1} r^{(i-2)} \right)} = \frac{\left(r^{(i-1)}, [\mathbf{M}]^{-1} r^{(i-1)} \right)}{\left(r^{(i-2)}, [\mathbf{M}]^{-1} r^{(i-2)} \right)} = \frac{\left(r^{(i-1)}, \mathcal{Z}^{(i-1)} \right)}{\left(r^{(i-2)}, \mathcal{Z}^{(i-2)} \right)}
\end{aligned}$$

前処理付共役勾配法: PCG

Preconditioned Conjugate Gradient Method (PCG)

```

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

```

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

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

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)

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。

- 背景
 - 有限体積法
 - 前処理付反復法
- **PCG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
 - PCG法

対象とするアプリケーションの概要

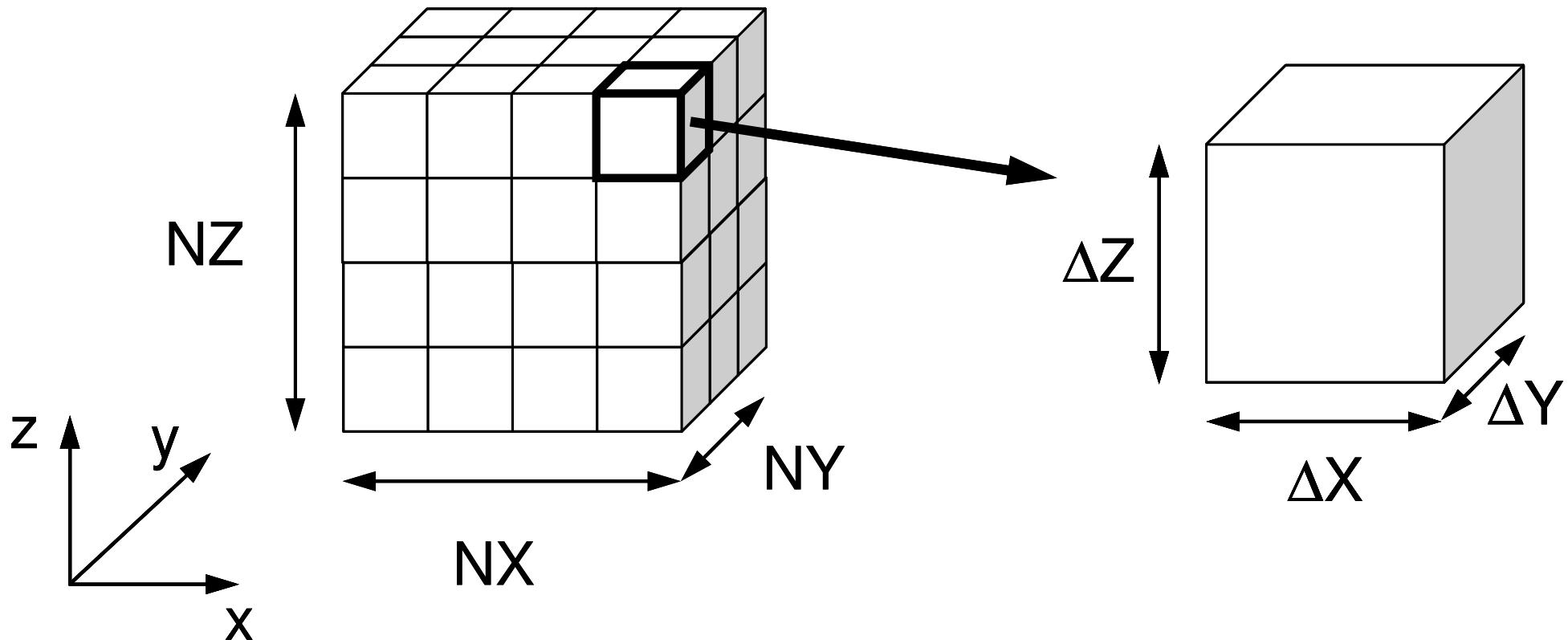
- 支配方程式：三次元ポアソン方程式

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

- 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 @ $Z=Z_{\max}$, 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

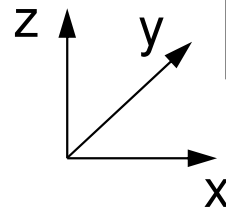
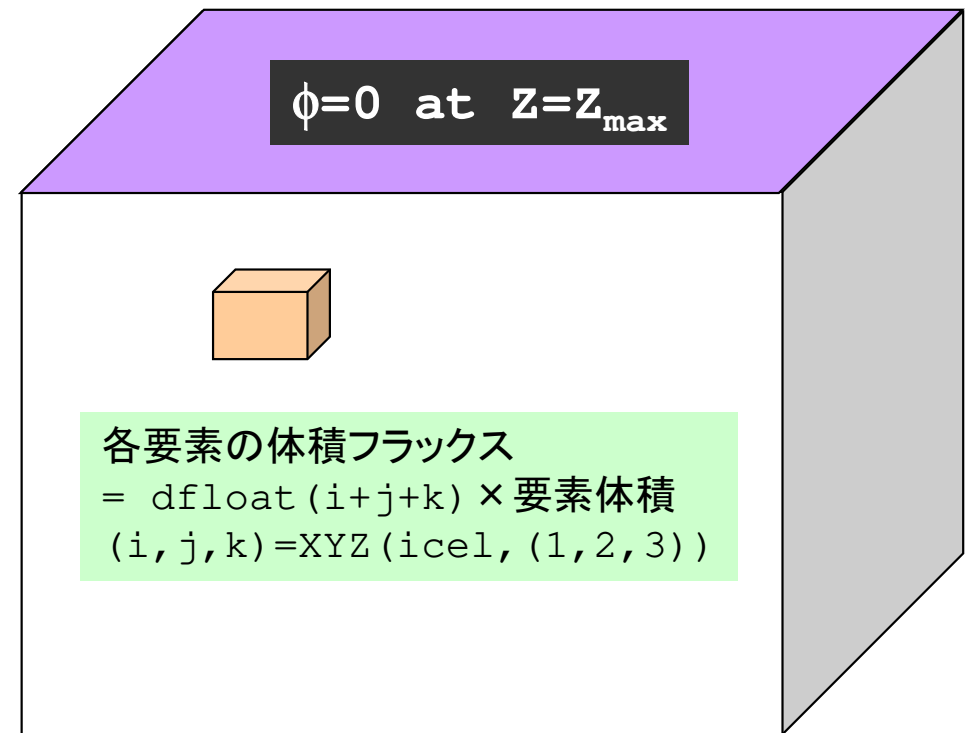
変数：要素中心で定義

ポアソン方程式

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

境界条件他

- 各要素で体積フラックス
- $Z=Z_{\max}$ 面で $\phi=0$



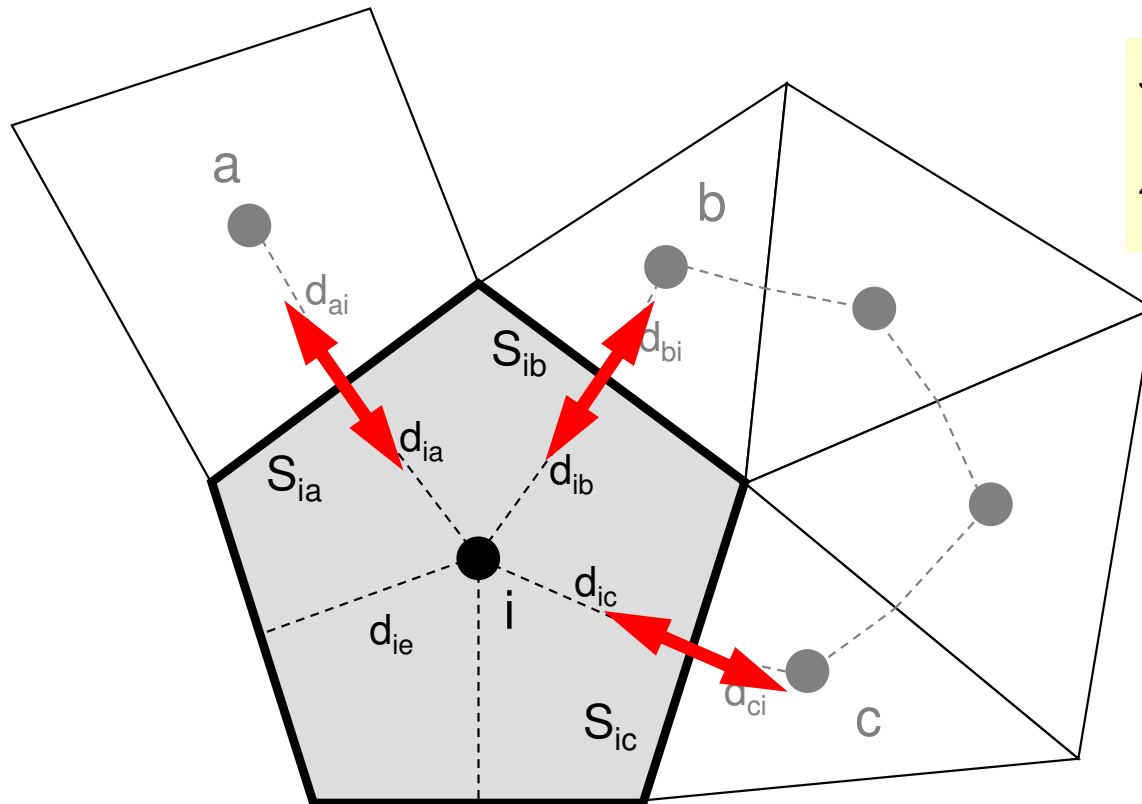
ポアソン方程式:

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

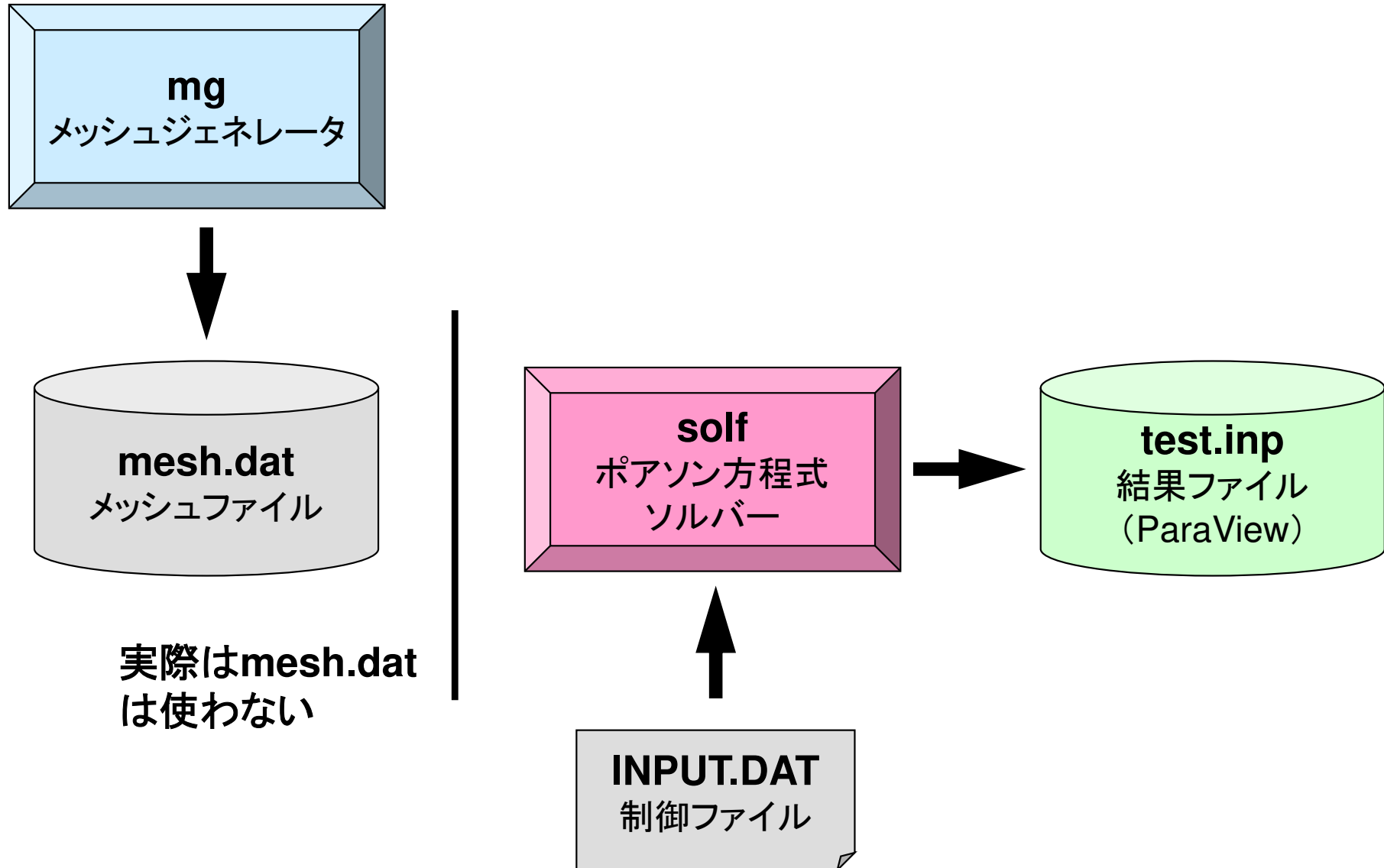
$$\sum_k \frac{S_{ik}}{d_{ik} + d_{ki}} (\phi_k - \phi_i) + V_i \dot{Q}_i = 0$$

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-FVM>/run
```

```
$> gfortran -O mg.f -o mg
```

```
$> ls mg
```

```
mg
```

メッシュジェネレータ: mg

```
$> cd ../src-f
```

```
$> make
```

```
$> ls ../run/solf
```

```
solf
```

ポアソン方程式ソルバー(FVM):
solf

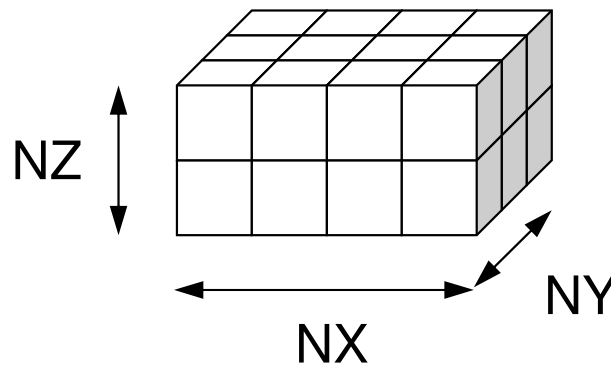
gfortranが無い場合はg95などをご利用ください
Makefileも書き換える

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

```

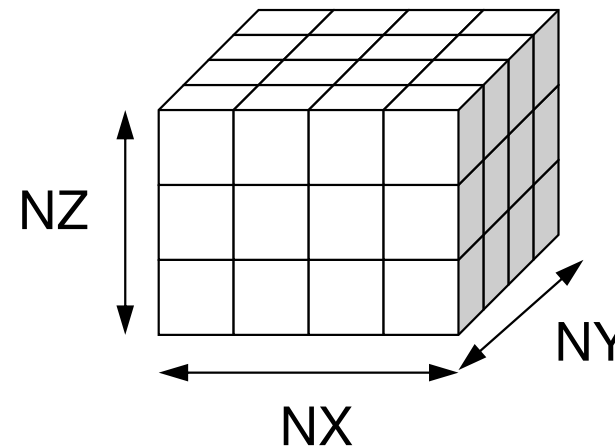
```

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: $NX \times NY \times NZ$

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
```

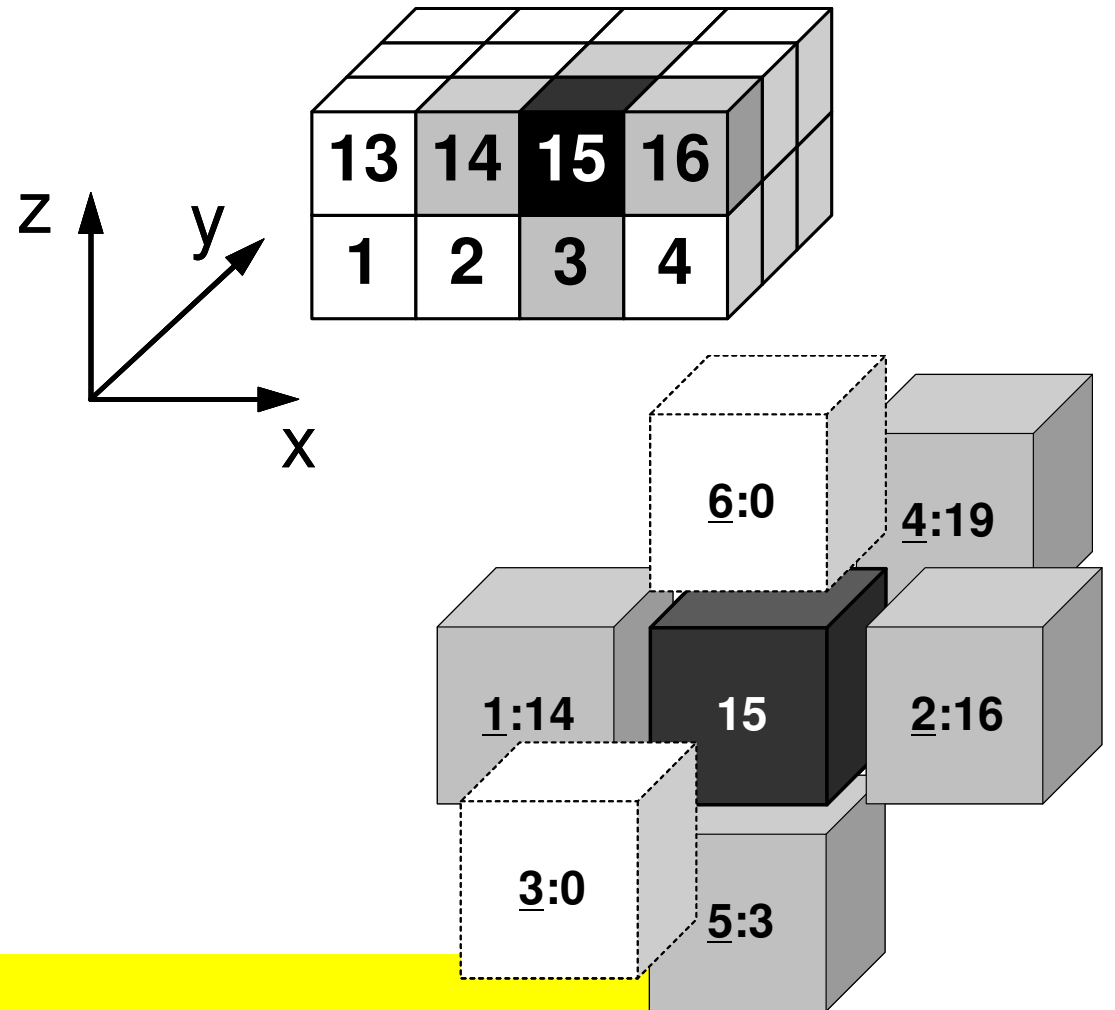
```
  read (21, '(10i10)') ii, (NEIBcell(i,k), k= 1, 6), (XYZ(i,j), j= 1, 3)
```

```
enddo
```


mesh.dat (4/5)

隣接要素: NEIBcell(i,k)

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

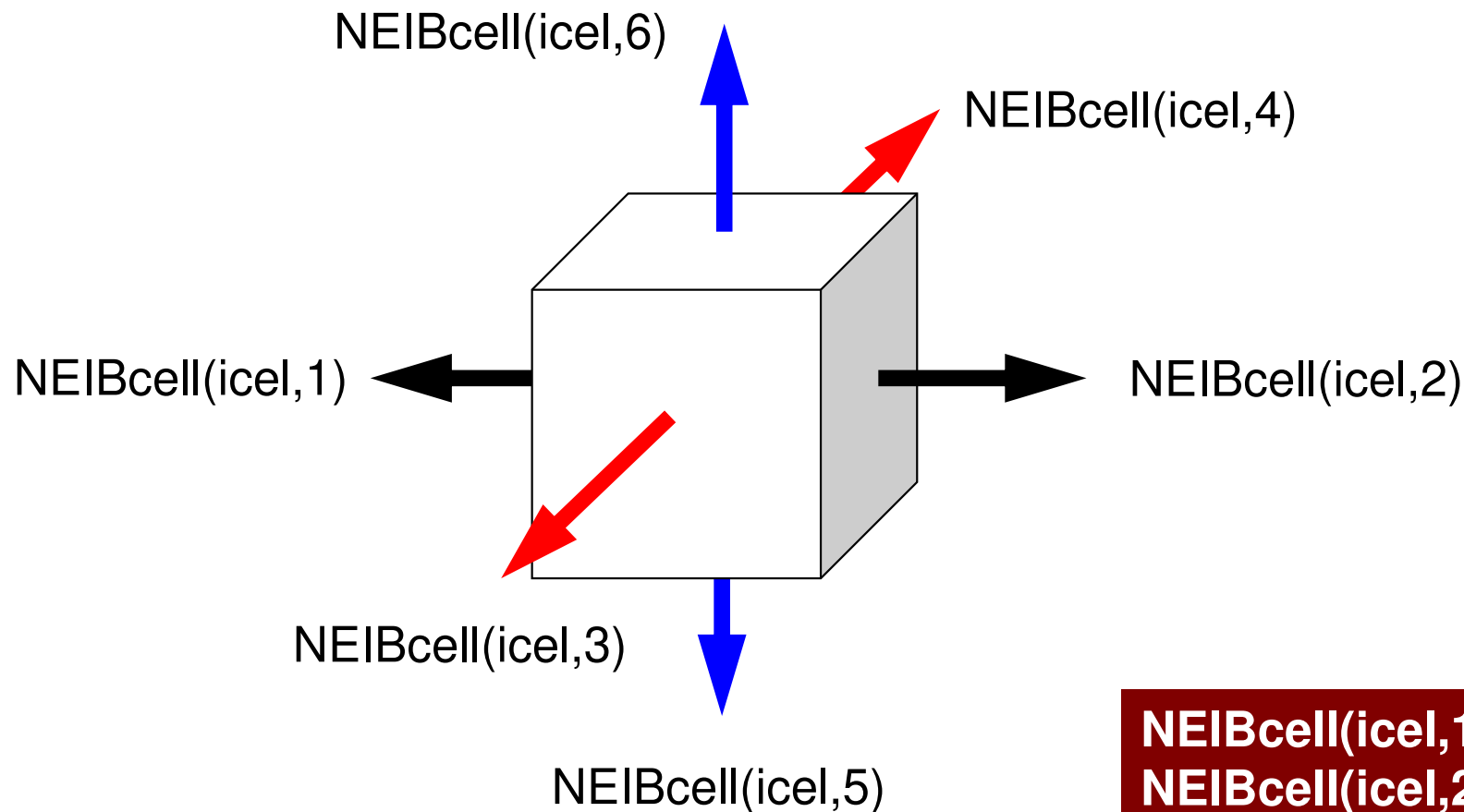


```
read (21, '(10i10)') NX, NY, NZ
read (21, '(10i10)') ICELTOT
```

1項目目は通し番号です(読み飛ばし)

```
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



NEIBcell(icel,1) = icel - 1
NEIBcell(icel,2) = icel + 1
NEIBcell(icel,3) = icel - NX
NEIBcell(icel,4) = icel + NX
NEIBcell(icel,5) = icel - NX*NY
NEIBcell(icel,6) = icel + NX*NY

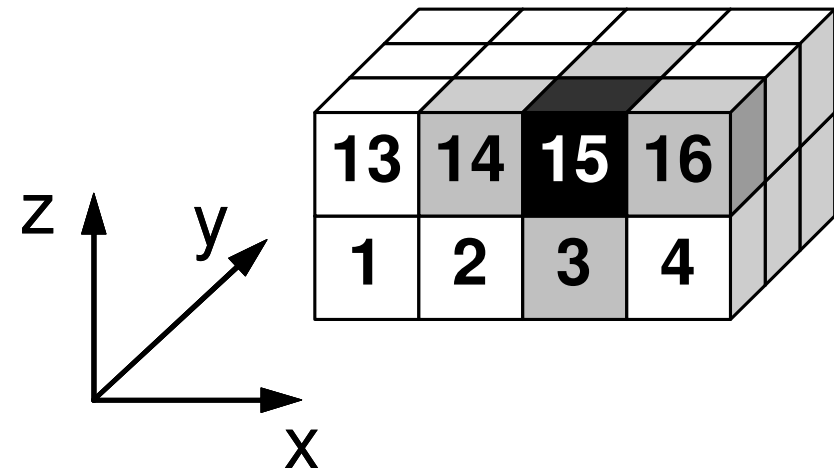
mesh.dat (5/5)

X,Y,Z方向の位置:XYZ(i,j)

```

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

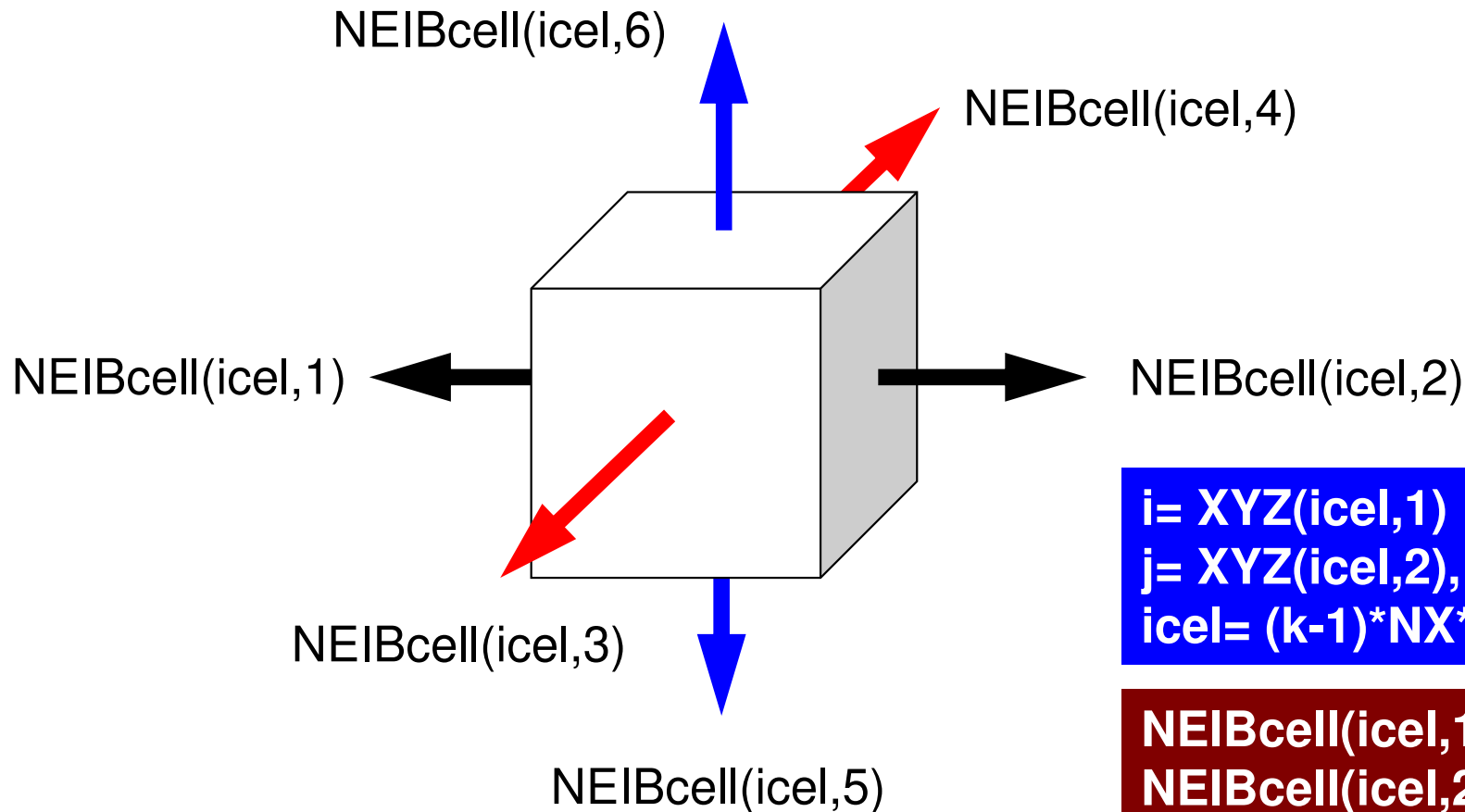
```

```

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

```

NEIBcell: 隣接している要素番号 境界面の場合は0



$$i = XYZ(icel,1)$$

$$j = XYZ(icel,2), k = XYZ(icel,3)$$

$$icel = (k-1)*NX*NY + (j-1)*NX + i$$

$$NEIBcell(icel,1) = icel - 1$$

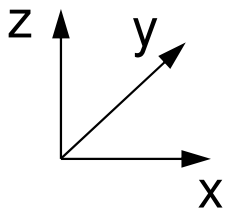
$$NEIBcell(icel,2) = icel + 1$$

$$NEIBcell(icel,3) = icel - NX$$

$$NEIBcell(icel,4) = icel + NX$$

$$NEIBcell(icel,5) = icel - NX*NY$$

$$NEIBcell(icel,6) = icel + NX*NY$$

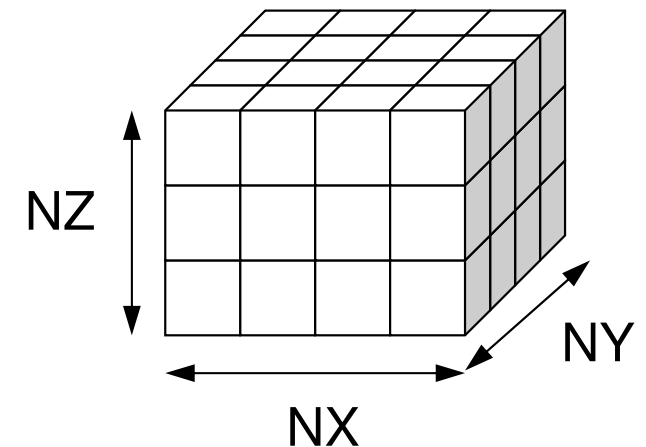


プログラムの実行

制御データ「<\$P-FVM>/run/INPUT.DAT」の作成

32 32 32	NX/NY/NZ
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
- DX, DY, DZ
 - 各要素のX,Y,Z方向辺長さ
- EPSICCG
 - ICCG法の収束判定値



プログラムの実行

```
$> cd <$P-FVM>/run  
$> ./solf
```

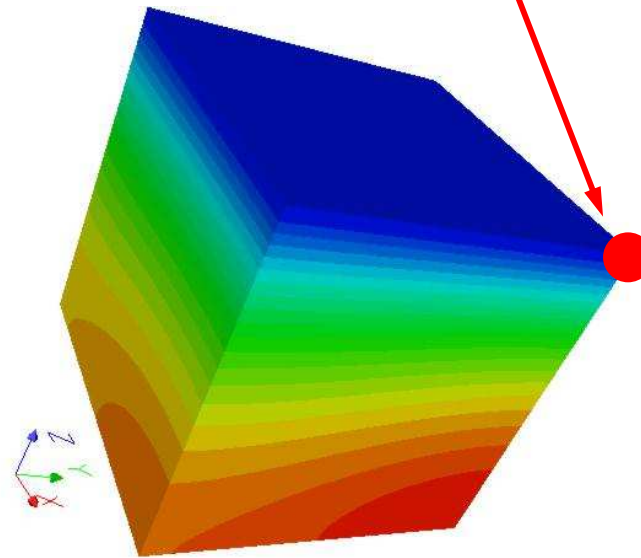
```
32  32  32  
  1    4.409359E+00  
101    1.807571E-02  
201    2.194680E-08  
208    9.354536E-09
```

NX, NY, NZ
1反復目の残差

収束時の反復回数残差 ($<10^{-8}$)

```
##ANSWER      32768      9.297409E+02  下図●点の答え
```

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



ParaView

- ファイルを開く
- 図の表示
- イメージファイルの保存

UCD Format (1/3)

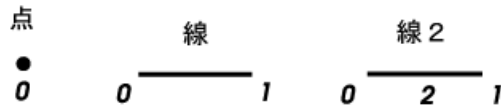
Unstructured Cell Data

要素の種類

キーワード

点

pt

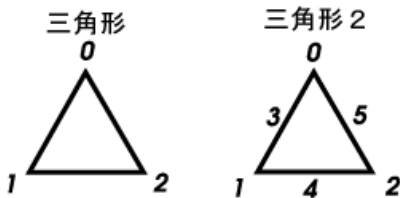


線

line

三角形

tri



四角形

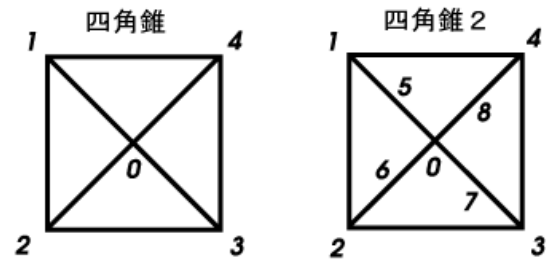
quad

四面体

tet

角錐

pyr

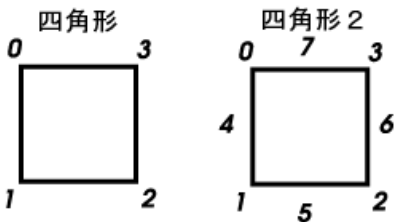


三角柱

prism

六面体

hex



二次要素

line2

三角形2

tri2

四角形2

quad2

四面体2

tet2

角錐2

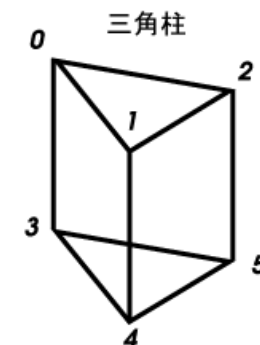
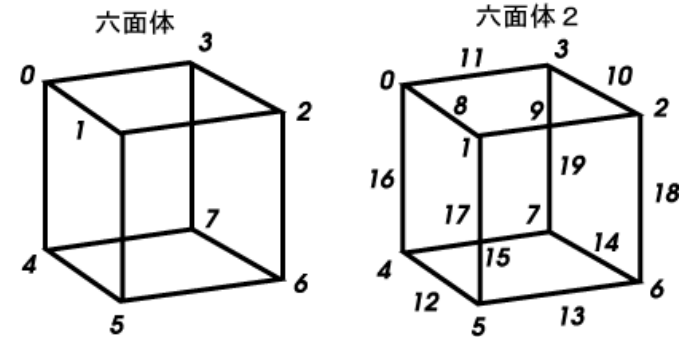
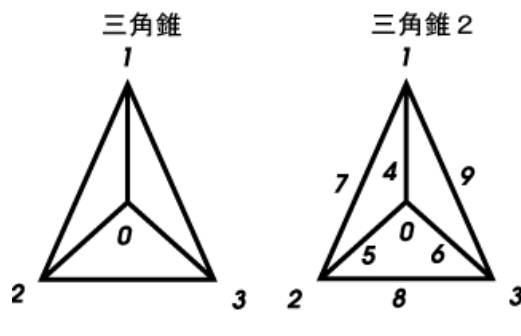
pyr2

三角柱2

prism2

六面体2

hex2



UCD Format (2/3)

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

UCD Format (3/3): Old Format

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

(節点番号1) (X座標) (Y座標) (Z座標)
(節点番号2) (X座標) (Y座標) (Z座標)

⋮

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

⋮

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

⋮

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

⋮

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

⋮

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

⋮

- 背景
 - 有限体積法
 - 前処理付反復法
- **PCG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
 - PCG法

プログラムの構成

```
program MAIN

use STRUCT
use PCG
use solver_PCG

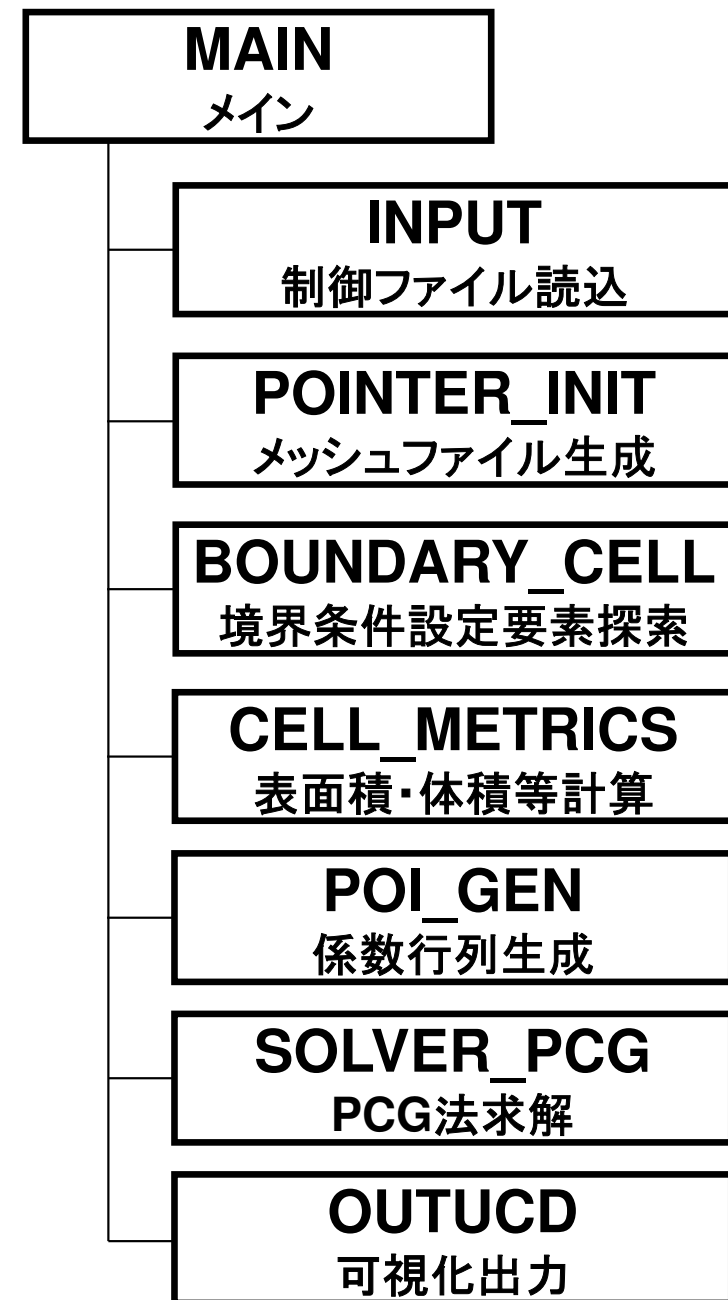
implicit REAL*8 (A-H, O-Z)

call INPUT
call POINTER_INIT
call BOUNDARY_CELL
call CELL_METRICS
call POI_GEN
PHI= 0. d0

call solve_PCG (...)

call OUTUCD

stop
end
```



変数群 (FVM関連) : struct.f

```

module STRUCT

    include 'precision.inc'

!C
!C-- METRICs & FLUX
    integer (kind=kint) :: ICELTOT, ICELTOTp, N
    integer (kind=kint) :: NX, NY, NZ, NXP1, NYP1, NZP1, IBNODTOT
    integer (kind=kint) :: NXc, NYc, NZc

    real (kind=kreal) ::
&    DX, DY, DZ, XAREA, YAREA, ZAREA, RDX, RDY, RDZ,
&    RDX2, RDY2, RDZ2, R2DX, R2DY, R2DZ
&
    real (kind=kreal), dimension(:), allocatable ::
&    VOLCEL, VOLNOD, RVC, RVN
&
    integer (kind=kint), dimension(:, :), allocatable ::
&    XYZ, NEIBcell
&
!C
!C-- BOUNDARYs
    integer (kind=kint) :: ZmaxCELtot
    integer (kind=kint), dimension(:), allocatable :: BC_INDEX, BC_NOD
    integer (kind=kint), dimension(:), allocatable :: ZmaxCEL

!C
!C-- WORK
    integer (kind=kint), dimension(:, :), allocatable :: IWKX
    real (kind=kreal), dimension(:, :), allocatable :: FCV

end module STRUCT

```

変数群 (FVM関連) : struct.f

変数名	種別	サイズ	内 容
NX, NY, NZ	I		x, y, z方向要素数
ICELTOT	I		全要素数 (=NX × NY × NZ)
N	I		節点数 (可視化用)
NXP1, NYP1, NZP1	I		x, y, z方向節点数 (可視化用)
IBNODTOT	I		NXP1 × NYP1
XYZ	I	(ICELTOT, 3)	要素座標
NEIBcell	I	(ICELTOT, 6)	隣接要素

変数群 (疎行列関連) : pcg.f

```
module PCG

integer :: N2
integer :: NLUmax, NLU, METHOD
integer :: NPLU

real(kind=8) :: EPSICCG

real(kind=8), dimension(:), allocatable :: D, PHI, BFORCE
real(kind=8), dimension(:), allocatable :: AMAT

integer, dimension(:), allocatable :: INLU
integer, dimension(:), allocatable :: indexLU, itemLU

end module PCG
```

変数群 (疎行列関連) : pcg.f

変数名	種別	サイズ	内 容
NLU	I		各要素隣接要素数最大値 (=6)
EPSICCG	R		PCG法収束判定基準
NPLU	I		非ゼロ非対角成分総数
indexLU	I	(0:ICELTOT)	疎行列 : 非零非対角成分数 (CRS)
itemLU	I	(NPLU)	疎行列 : 非零非対角成分列番号 (CRS)
PHI	R	(ICELTOT)	従属変数ベクトル
BFORCE	R	(ICELTOT)	右辺ベクトル
D	R	(ICELTOT)	疎行列 : 対角成分 (CRS)
AMAT	R	(NPLU)	疎行列 : 非零非対角成分 (CRS)
INLU	I	(ICELTOT)	各行の非ゼロ非対角成分数, NPLU算出に使用
N2, NLUmax, METHOD, ORDER_METHOD	I		(未使用)

行列ベクトル積： $\{q\}=[A]\{p\}$

```
do i= 1, N
```

```
    VAL= D(i)*p(i)
```

```
    do k= indexLU(i-1)+1, indexLU(i)
```

```
        VAL= VAL + AMAT(k)*p(itemLU(k))
```

```
    enddo
```

```
    q(i)= VAL
```

```
enddo
```

プログラムの構成

```
program MAIN

use STRUCT
use PCG
use solver_PCG

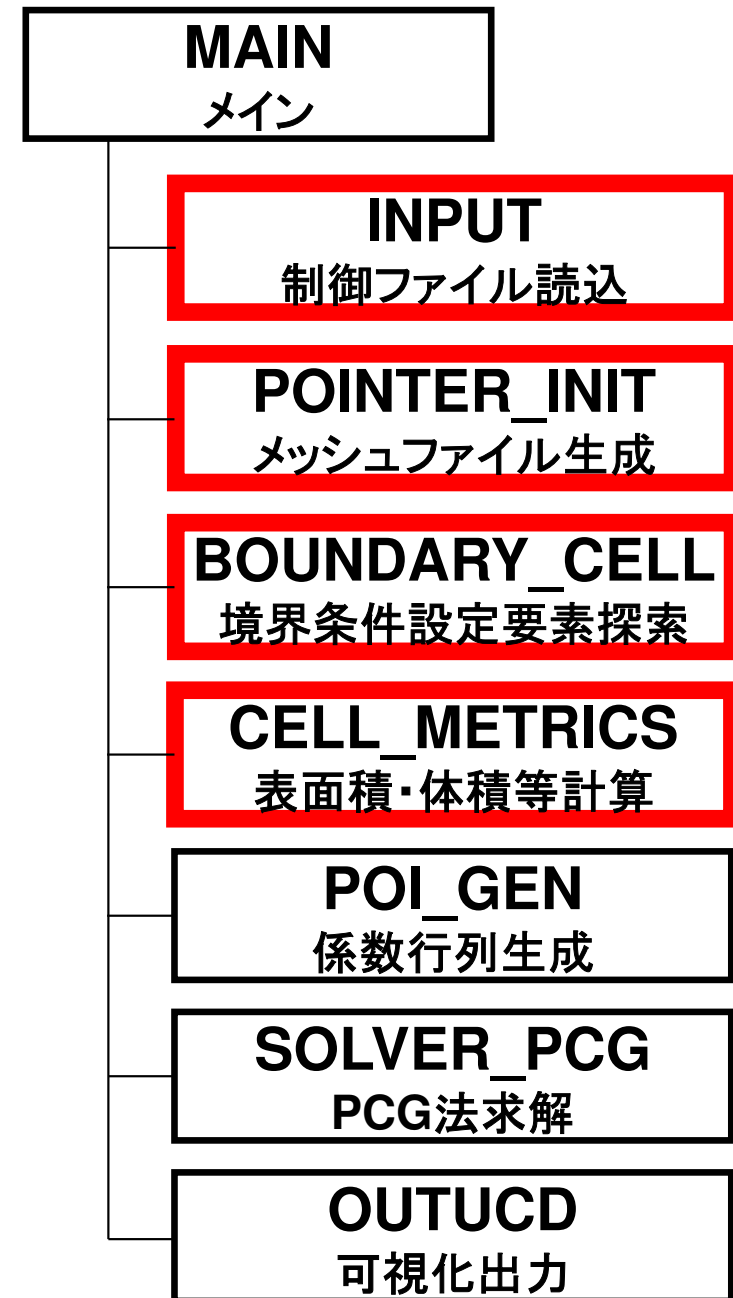
implicit REAL*8 (A-H, O-Z)

call INPUT
call POINTER_INIT
call BOUNDARY_CELL
call CELL_METRICS
call POI_GEN
PHI= 0. d0

call solve_PCG (...)

call OUTUCD

stop
end
```



input: 「INPUT.DAT」の読み込み

```
subroutine INPUT
use STRUCT
use PCG

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

character*80 CNTFIL

open  (11, file='INPUT.DAT', status='unknown')
  read (11,*) NX, NY, NZ
  read (11,*) DX, DY, DZ
  read (11,*) EPSICCG
close (11)

return
end
```

```
32 32 32          NX/NY/NZ
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

```
!C
!C***
!C*** POINTER_INIT
!C***
!C
      subroutine POINTER_INIT

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

!C
!C +-----+
!C | Generating MESH info. |
!C +-----+
!C===
      ICELTOT= NX  * NY  * NZ

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

      allocate (NEIBcell(ICELTOT,6), XYZ(ICELTOT,3))
      NEIBcell= 0
```

ICELTOT

要素数 ($NX \times NY \times NZ$)

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

$NXP1 \times NYP1$

XYZ (ICELTOT, 3)

要素座標

NEIBcell (ICELTOT, 6)

隣接要素

**allocate/deallocate
for C**

allocate, deallocate (1/2)

Same interface with FORTRAN

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

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

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

```
VOLCEL = (double *)allocate_vector(sizeof(double), ICELTOT);
indexLU= (int *)allocate_vector(sizeof(int), ICELTOT+1);
```

allocate, deallocate (2/2)

Same interface with FORTRAN

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

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

NEIBcell = (int **)allocate_matrix(sizeof(int), ICELTOT, 6);
XYZ      = (int **)allocate_matrix(sizeof(int), ICELTOT, 3);
```

pointer_init(2/3) : 「mesh.dat」の作成

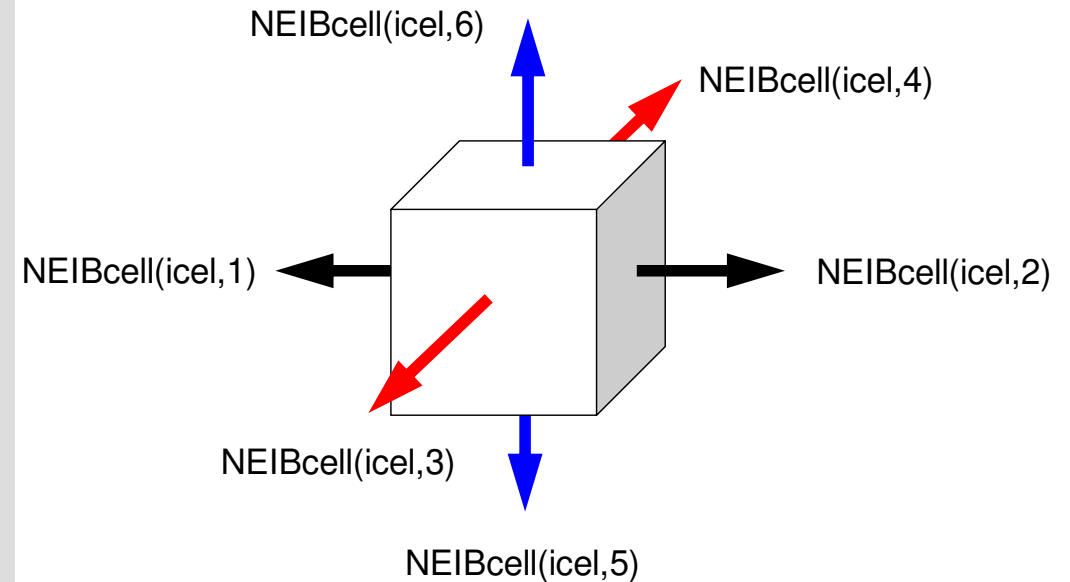
```

do k= 1, NZ
  do j= 1, NY
    do i= 1, NX
      icel= (k-1)*NX*NY + (j-1)*NX + i
      NEIBcell(icel, 1)= icel - 1
      NEIBcell(icel, 2)= icel + 1
      NEIBcell(icel, 3)= icel - NX
      NEIBcell(icel, 4)= icel + NX
      NEIBcell(icel, 5)= icel - NX*NY
      NEIBcell(icel, 6)= icel + NX*NY
      if (i. eq. 1) NEIBcell(icel, 1)= 0
      if (i. eq. NX) NEIBcell(icel, 2)= 0
      if (j. eq. 1) NEIBcell(icel, 3)= 0
      if (j. eq. NY) NEIBcell(icel, 4)= 0
      if (k. eq. 1) NEIBcell(icel, 5)= 0
      if (k. eq. NZ) NEIBcell(icel, 6)= 0

      XYZ(icel, 1)= i
      XYZ(icel, 2)= j
      XYZ(icel, 3)= k

    enddo
  enddo
enddo
!C===

```



```

i= XYZ(icel,1)
j= XYZ(icel,2), k= XYZ(icel,3)
icel= (k-1)*NX*NY + (j-1)*NX + i

```

```

NEIBcell(icel,1)= icel - 1
NEIBcell(icel,2)= icel + 1
NEIBcell(icel,3)= icel - NX
NEIBcell(icel,4)= icel + NX
NEIBcell(icel,5)= icel - NX*NY
NEIBcell(icel,6)= icel + NX*NY

```

pointer_init(3/3)

```
!C
!C +-----+
!C | Parameters |
!C +-----+
!C===
    if (DX.le.0.0e0) then
        DX= 1.d0 / dfloat(NX)
        DY= 1.d0 / dfloat(NY)
        DZ= 1.d0 / dfloat(NZ)
    endif

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

    IBNODTOT= NXP1 * NYP1
    N          = NXP1 * NYP1 * NZP1
!C===
return
end
```

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) : 節点 ⇒ 可視化用

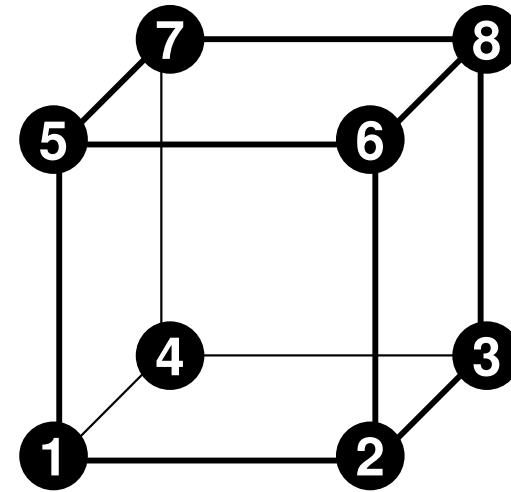
```

!C
!C +-----+
!C | Parameters |
!C +-----+
!C===
    if (DX.le.0.0e0) then
        DX= 1.d0 / dfloat(NX)
        DY= 1.d0 / dfloat(NY)
        DZ= 1.d0 / dfloat(NZ)
    endif

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

    IBNODTOT= NXP1 * NYP1
    N        = NXP1 * NYP1 * NZP1
!C===
    return
    end

```



NXP1, NYP1, NZP1 :
x, y, z方向節点数

IBNODTOT :
 $NXP1 \times NYP1$

N :
節点数 (可視化に使用)

boundary_cell

```

!C
!C***
!C*** BOUNDARY_CELL
!C***
!C
subroutine BOUNDARY_CELL
use STRUCT

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

!C
!C +-----+
!C | Zmax |
!C +-----+
!C===
  IFACTOT= NX * NY
  ZmaxCELtot= IFACTOT

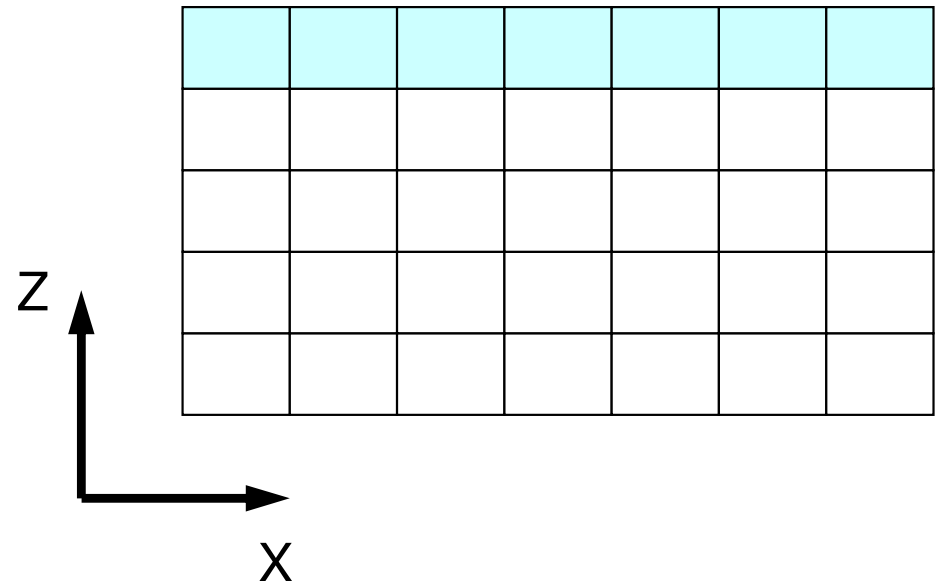
  allocate (ZmaxCEL(ZmaxCELtot))

  icou= 0
  k = NZ
  do j= 1, NY
  do i= 1, NX
    icel= (k-1)*IFACTOT + (j-1)*NX + i
    icou= icou + 1
    ZmaxCEL(icou)= icel
  enddo
  enddo
!C===
return
end

```

$Z=Z_{\max}$ の要素の定義

総数: ZmaxCELtot
 要素番号: ZmaxCEL(:)



cell_metrics

```

!C
!C***
!C*** CELL_METRICS
!C***
!C
      subroutine CELL_METRICS
      use STRUCT
      use PCG
      implicit REAL*8 (A-H, O-Z)
!C
!C-- ALLOCATE
      allocate (VOLCEL(ICELTOT))
      allocate (   RVC(ICELTOT))
!C
!C-- VOLUME, AREA, PROJECTION etc.
      XAREA= DY * DZ
      YAREA= DX * DZ
      ZAREA= DX * DY

      RDX= 1. d0 / DX
      RDY= 1. d0 / DY
      RDZ= 1. d0 / DZ

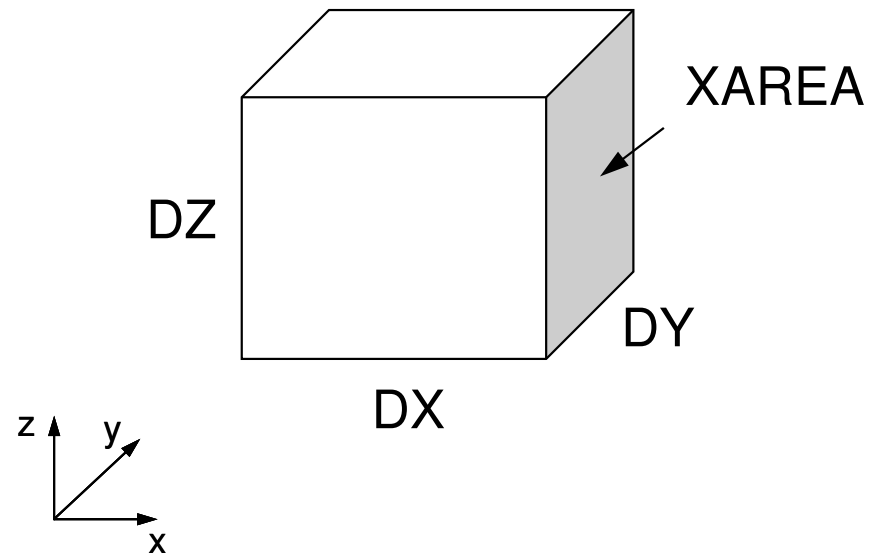
      RDX2= 1. d0 / (DX**2)
      RDY2= 1. d0 / (DY**2)
      RDZ2= 1. d0 / (DZ**2)
      R2DX= 1. d0 / (0. 50d0*DX)
      R2DY= 1. d0 / (0. 50d0*DY)
      R2DZ= 1. d0 / (0. 50d0*DZ)

      V0= DX * DY * DZ
      RVO= 1. d0/V0
      VOLCEL= V0
      RVC   = RVO

      return
      end

```

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

```

!C
!C***
!C*** CELL_METRICS
!C***
!C
      subroutine CELL_METRICS
      use STRUCT
      use PCG
      implicit REAL*8 (A-H, O-Z)

!C
!C-- ALLOCATE
      allocate (VOLGEL(ICELTOT))
      allocate (   RVC(ICELTOT))

!C
!C-- VOLUME, AREA, PROJECTION etc.
      XAREA= DY * DZ
      YAREA= DX * DZ
      ZAREA= DX * DY

      RDX= 1. d0 / DX
      RDY= 1. d0 / DY
      RDZ= 1. d0 / DZ

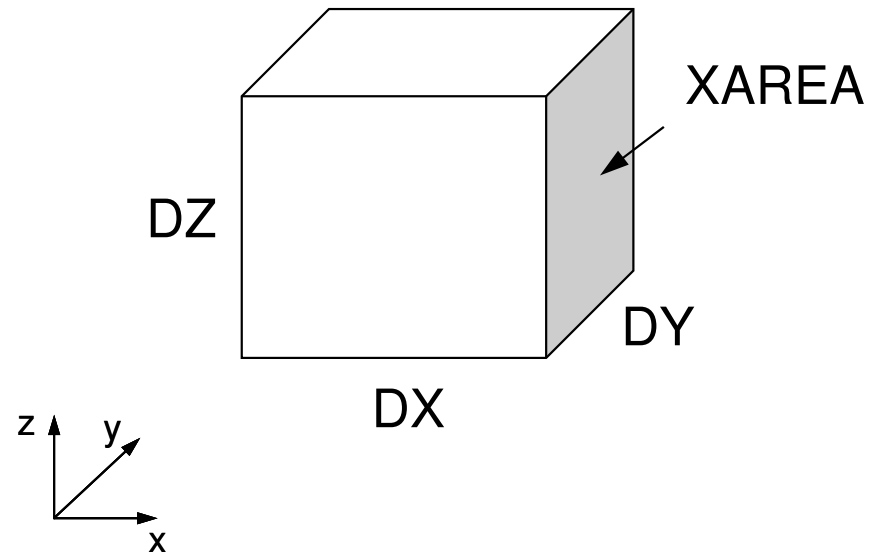
      RDX2= 1. d0 / (DX**2)
      RDY2= 1. d0 / (DY**2)
      RDZ2= 1. d0 / (DZ**2)
      R2DX= 1. d0 / (0. 50d0*DX)
      R2DY= 1. d0 / (0. 50d0*DY)
      R2DZ= 1. d0 / (0. 50d0*DZ)

      V0= DX * DY * DZ
      RVO= 1. d0/V0
      VOLGEL= V0
      RVC   = RVO

      return
      end

```

計算に必要な諸パラメータ



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

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

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

cell_metrics

```

!C
!C***
!C*** CELL_METRICS
!C***
!C
      subroutine CELL_METRICS
      use STRUCT
      use PCG
      implicit REAL*8 (A-H, O-Z)
!C
!C-- ALLOCATE
      allocate (VOLCEL(ICELTOT))
      allocate (   RVC(ICELTOT))
!C
!C-- VOLUME, AREA, PROJECTION etc.
      XAREA= DY * DZ
      YAREA= DX * DZ
      ZAREA= DX * DY

      RDX= 1. d0 / DX
      RDY= 1. d0 / DY
      RDZ= 1. d0 / DZ

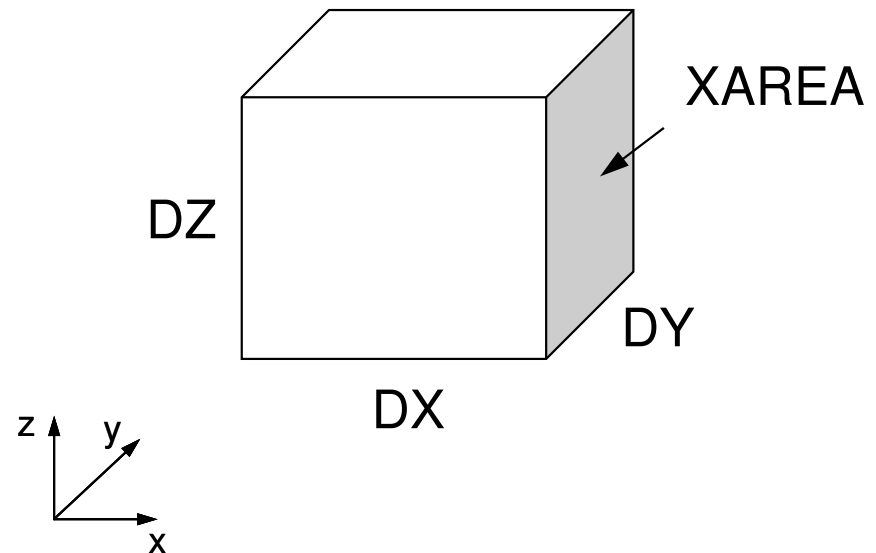
      RDX2= 1. d0 / (DX**2)
      RDY2= 1. d0 / (DY**2)
      RDZ2= 1. d0 / (DZ**2)
      R2DX= 1. d0 / (0. 50d0*DX)
      R2DY= 1. d0 / (0. 50d0*DY)
      R2DZ= 1. d0 / (0. 50d0*DZ)

      V0= DX * DY * DZ
      RVO= 1. d0/V0
      VOLCEL= V0
      RVC   = RVO

      return
      end

```

計算に必要な諸パラメータ



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

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

プログラムの構成

```
program MAIN

use STRUCT
use PCG
use solver_PCG

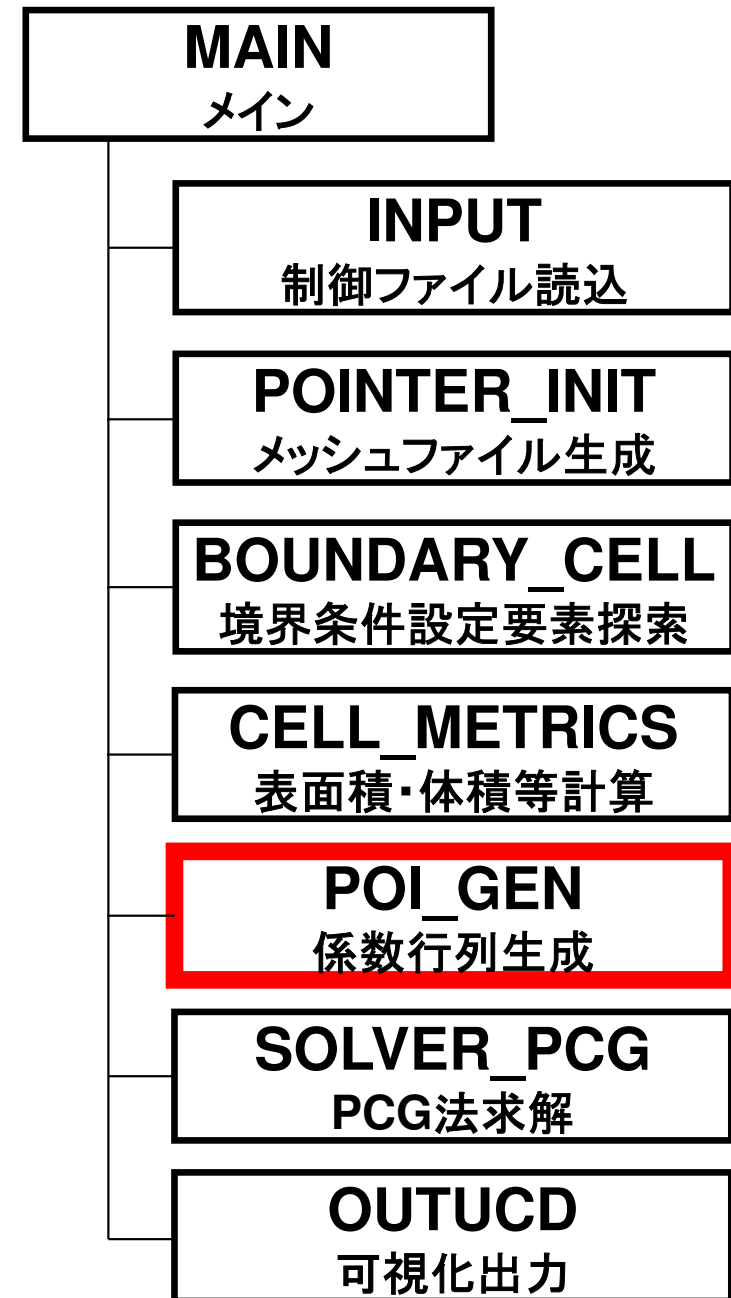
implicit REAL*8 (A-H, O-Z)

call INPUT
call POINTER_INIT
call BOUNDARY_CELL
call CELL_METRICS
call POI_GEN
PHI= 0. d0

call solve_PCG (...)

call OUTUCD

stop
end
```



poi_gen(1/6)

```
use STRUCT
use PCG

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

!C
!C-- INIT.
nn = ICELTOT

NLU= 6

allocate (BFORCE(nn), D(nn), PHI(nn))
allocate (INLU(nn))

PHI = 0. d0
BFORCE= 0. d0
D = 0. d0

INLU= 0
```

変数群 (疎行列関連) : pcg.f

変数名	種別	サイズ	内 容
NLU	I		各要素隣接要素数最大値 (=6)
EPSICCG	R		PCG法収束判定基準
NPLU	I		非ゼロ非対角成分総数
indexLU	I	(0:ICELTOT)	疎行列 : 非零非対角成分数 (CRS)
itemLU	I	(NPLU)	疎行列 : 非零非対角成分列番号 (CRS)
PHI	R	(ICELTOT)	従属変数ベクトル
BFORCE	R	(ICELTOT)	右辺ベクトル
D	R	(ICELTOT)	疎行列 : 対角成分 (CRS)
AMAT	R	(NPLU)	疎行列 : 非零非対角成分 (CRS)
INLU	I	(ICELTOT)	各行の非ゼロ非対角成分数, NPLU算出に使用


```

do icel= 1, ICELTOT
  icN1= NEIBcell (icel, 1)
  icN2= NEIBcell (icel, 2)
  icN3= NEIBcell (icel, 3)
  icN4= NEIBcell (icel, 4)
  icN5= NEIBcell (icel, 5)
  icN6= NEIBcell (icel, 6)

  if (icN5.ne.0) then
    INLU(icel)= INLU(icel) + 1
  endif

  if (icN3.ne.0) then
    INLU(icel)= INLU(icel) + 1
  endif

  if (icN1.ne.0) then
    INLU(icel)= INLU(icel) + 1
  endif

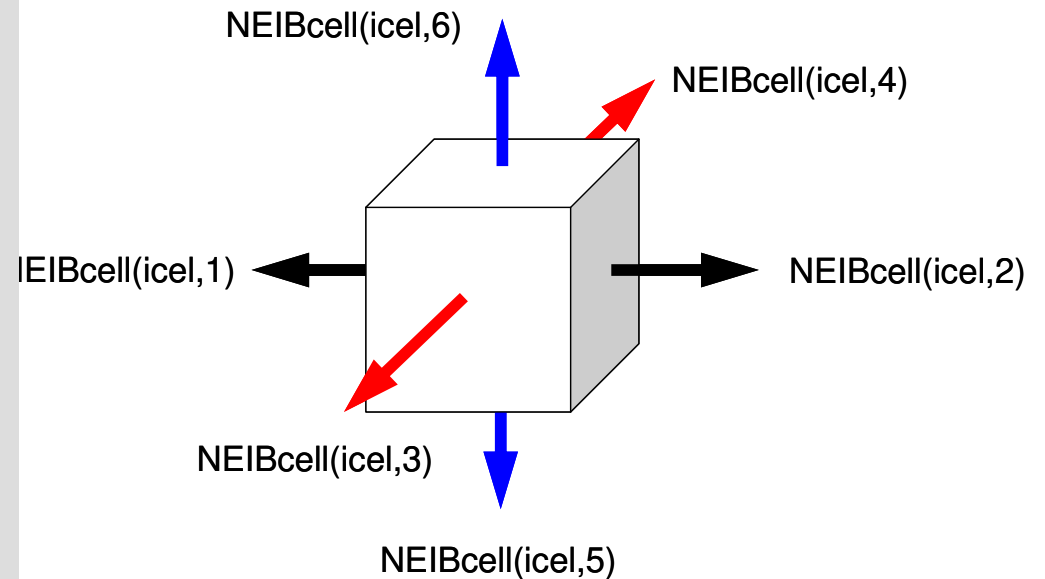
  if (icN2.ne.0) then
    INLU(icel)= INLU(icel) + 1
  endif

  if (icN4.ne.0) then
    INLU(icel)= INLU(icel) + 1
  endif

  if (icN6.ne.0) then
    INLU(icel)= INLU(icel) + 1
  endif

```

poi_gen (2/6)



下三角成分

$$\text{NEIBcell}(\text{icel},5) = \text{icel} - \text{NX} * \text{NY}$$

$$\text{NEIBcell}(\text{icel},3) = \text{icel} - \text{NX}$$

$$\text{NEIBcell}(\text{icel},1) = \text{icel} - 1$$

上三角成分

$$\text{NEIBcell}(\text{icel},2) = \text{icel} + 1$$

$$\text{NEIBcell}(\text{icel},4) = \text{icel} + \text{NX}$$

$$\text{NEIBcell}(\text{icel},6) = \text{icel} + \text{NX} * \text{NY}$$

poi_gen (3/6)

```

!C
!C-- 1D array
      allocate (indexLU(0:nn))
      indexLU= 0

      do icel= 1, ICELTOT
        indexLU(icel)= INLU(icel)
      enddo

      do icel= 1, ICELTOT
        indexLU(icel)= indexLU(icel) + indexLU(icel-1)
      enddo

      NPLU= indexLU(ICELTOT)

      allocate (itemLU(NPLU), AMAT(NPLU))
      itemLU= 0
      AMAT = 0.d0

```

```

do i= 1, N
  VAL= D(i)*p(i)
  do k= indexLU(i-1)+1, indexLU(i)
    VAL= VAL + AMAT(k)*p(itemLU(k))
  enddo
  q(i)= VAL
enddo

```

変数名	種別	サイズ	内 容
NLU	I		各要素隣接要素数最大値 (=6)
EPSICCG	R		PCG法収束判定基準
NPLU	I		非ゼロ非対角成分総数
indexLU	I	(0:ICELTOT)	疎行列：非零非対角成分数 (CRS)
itemLU	I	(NPLU)	疎行列：非零非対角成分列番号 (CRS)
PHI	R	(ICELTOT)	従属変数ベクトル
BFORCE	R	(ICELTOT)	右辺ベクトル
D	R	(ICELTOT)	疎行列：対角成分 (CRS)
AMAT	R	(NPLU)	疎行列：非零非対角成分 (CRS)
INLU	I	(ICELTOT)	各行の非ゼロ非対角成分数, NPLU算出に使用

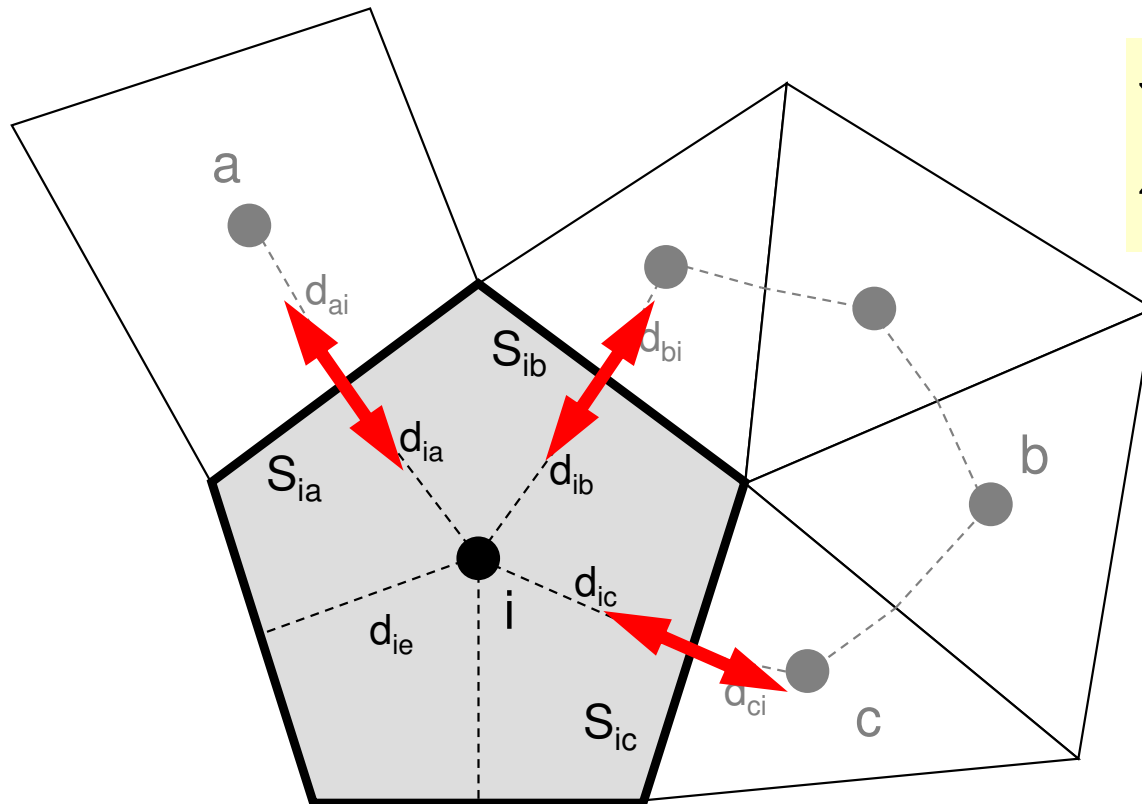
ポアソン方程式:

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

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

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

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

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

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

D (対角成分)

AMAT
(非対角成分)

BFORCE
(右辺)

```

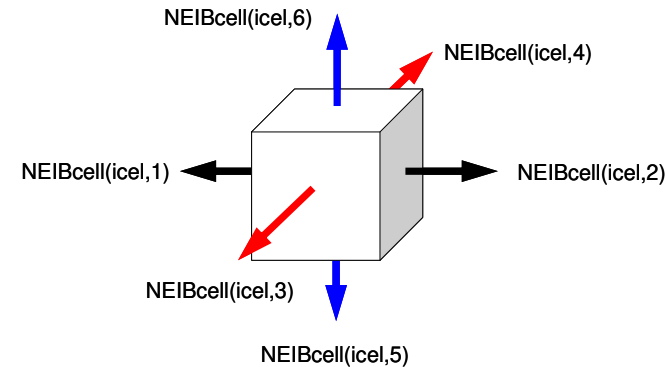
do icel= 1, ICELTOT
  icN1= NEIBcell (icel, 1)
  icN2= NEIBcell (icel, 2)
  icN3= NEIBcell (icel, 3)
  icN4= NEIBcell (icel, 4)
  icN5= NEIBcell (icel, 5)
  icN6= NEIBcell (icel, 6)
  VOL0= VOLCEL (icel)
  icou= 0
  if (icN5.ne.0) then
    coef =RDZ * ZAREA
    D(icel)= D(icel) - coef
    icou= icou + 1
    k= icou + indexLU(icel-1)
    itemLU(k)= icN5
    AMAT(k)= coef
  endif

  if (icN3.ne.0) then
    coef =RDY * YAREA
    D(icel)= D(icel) - coef
    icou= icou + 1
    k= icou + indexLU(icel-1)
    itemLU(k)= icN3
    AMAT(k)= coef
  endif

  if (icN1.ne.0) then
    coef =RDX * xAREA
    D(icel)= D(icel) - coef
    icou= icou + 1
    k= icou + indexLU(icel-1)
    itemLU(k)= icN1
    AMAT(k)= coef
  endif
endif

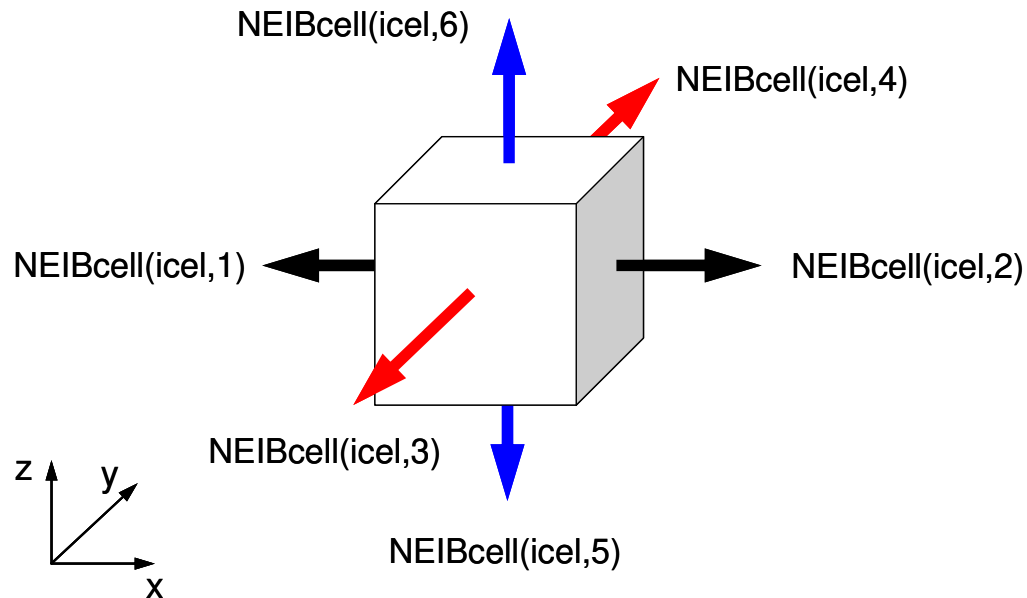
```

poi_gen (4/6)



$$\begin{aligned}
 & \frac{\phi_{neib(icel,1)} - \phi_{icel}}{\Delta x} \Delta y \Delta z + \frac{\phi_{neib(icel,2)} - \phi_{icel}}{\Delta x} \Delta y \Delta z + \\
 & \frac{\phi_{neib(icel,3)} - \phi_{icel}}{\Delta y} \Delta z \Delta x + \frac{\phi_{neib(icel,4)} - \phi_{icel}}{\Delta y} \Delta z \Delta x + \\
 & \boxed{\frac{\phi_{neib(icel,5)} - \phi_{icel}}{\Delta z} \Delta x \Delta y} + \frac{\phi_{neib(icel,6)} - \phi_{icel}}{\Delta z} \Delta x \Delta y + f_{icel} \Delta x \Delta y \Delta z = 0
 \end{aligned}$$

係数の計算



```

if (icN5.ne.0) then
  coef = RDZ * ZAREA
  D(icel) = D(icel) - coef
  icou = icou + 1
  k = icou + indexLU(icel-1)

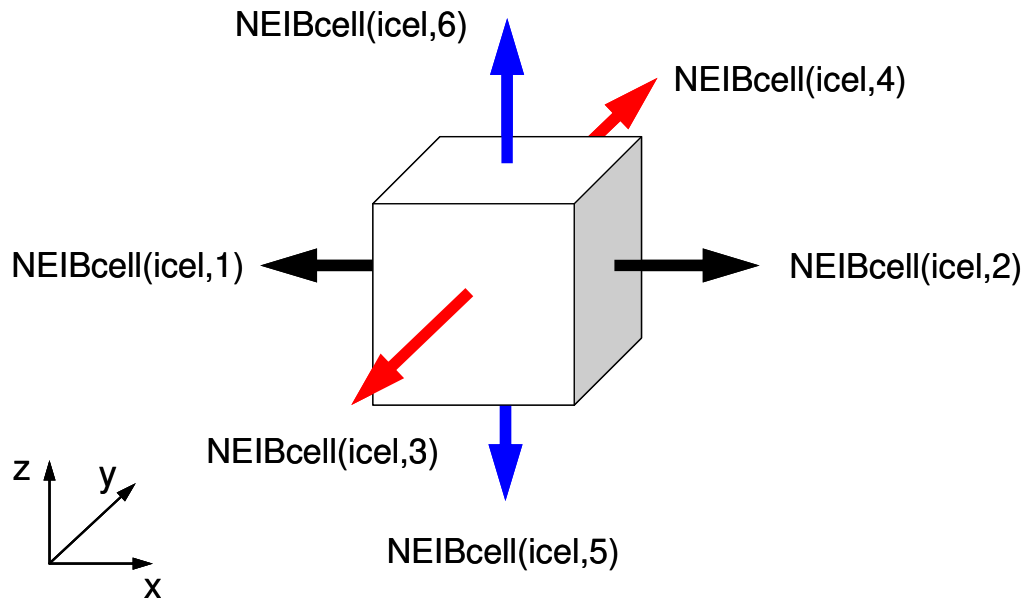
  itemLU(k) = icN5
  AMAT(k) = coef
endif
  
```

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

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

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

係数の計算



```

if (icN5.ne.0) then
  coef = RDZ * ZAREA
  D(icel) = D(icel) - coef
  icou = icou + 1
  k = icou + indexLU(icel-1)

  itemLU(k) = icN5
  AMAT(k) = coef
endif

```

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

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

ZAREA

RDZ

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

poi_gen(5/6)

```
do icel= 1, ICELTOT
  (...)
  if (icN2.ne.0) then
    coef =RDX * xAREA
    D(icel)= D(icel) - coef
    icou= icou + 1
    k= icou + indexLU(icel-1)
    itemLU(k)= icN2
    AMAT(k)= coef
  endif
  if (icN4.ne.0) then
    coef =RDY * YAREA
    D(icel)= D(icel) - coef
    icou= icou + 1
    k= icou + indexLU(icel-1)
    itemLU(k)= icN4
    AMAT(k)= coef
  endif
  if (icN6.ne.0) then
    coef =RDZ * ZAREA
    D(icel)= D(icel) - coef
    icou= icou + 1
    k= icou + indexLU(icel-1)
    itemLU(k)= icN6
    AMAT(k)= coef
  endif

  ii= XYZ(icel, 1)
  jj= XYZ(icel, 2)
  kk= XYZ(icel, 3)

  BFORCE(icel)= -dfloat(ii+jj+kk)*VOL0

enddo
```


poi_gen(5/6)

体積発熱

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

$$i_0 = XYZ(icel, 1),$$

$$j_0 = XYZ(icel, 2),$$

$$k_0 = XYZ(icel, 3)$$

$XYZ(icel, k)$ ($k=1,2,3$) 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。

```

do icel= 1, ICELTOT
  (...)
  if (icN2.ne.0) then
    coef =RDX * xAREA
    D(icel)= D(icel) - coef
    icou= icou + 1
    k= icou + indexLU(icel-1)
    itemLU(k)= icN2
    AMAT(k)= coef
  endif
  if (icN4.ne.0) then
    coef =RDY * YAREA
    D(icel)= D(icel) - coef
    icou= icou + 1
    k= icou + indexLU(icel-1)
    itemLU(k)= icN4
    AMAT(k)= coef
  endif
  if (icN6.ne.0) then
    coef =RDZ * ZAREA
    D(icel)= D(icel) - coef
    icou= icou + 1
    k= icou + indexLU(icel-1)
    itemLU(k)= icN6
    AMAT(k)= coef
  endif

  ii= XYZ(icel, 1)
  jj= XYZ(icel, 2)
  kk= XYZ(icel, 3)

  BFORCE(icel)= -dfloat(ii+jj+kk)*VOL0

enddo

```

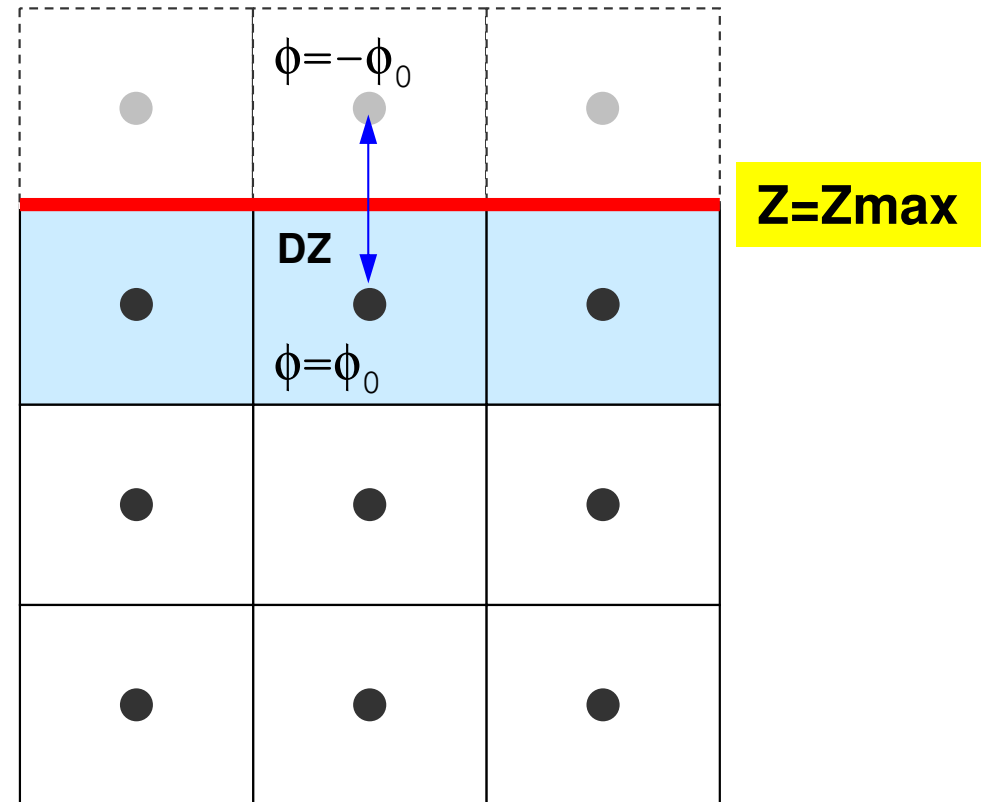
poi_gen(7/7)

係数の計算(境界面)

```

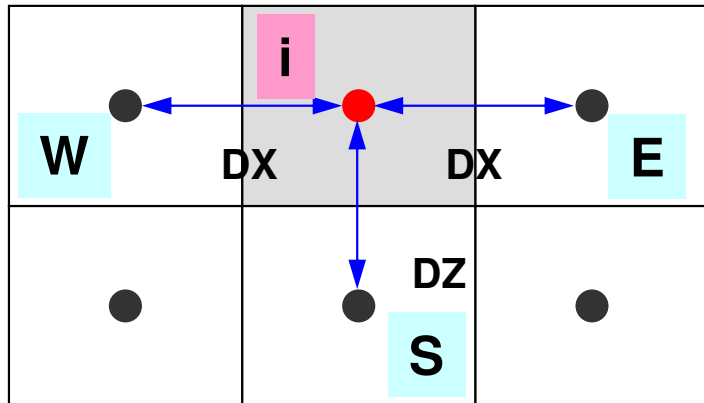
!C
!C +-----+
!C | DIRICHLET BOUNDARY CELLS |
!C +-----+
!C TOP SURFACE
!C===
do ib= 1, ZmaxCELTot
  icel= ZmaxCEL(ib)
  coef= 2. d0 * RDZ * ZAREA
  D(icel)= D(icel) - coef
enddo
!C===
return
end

```



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 ($\phi=0@Z=Z_{\max}$)



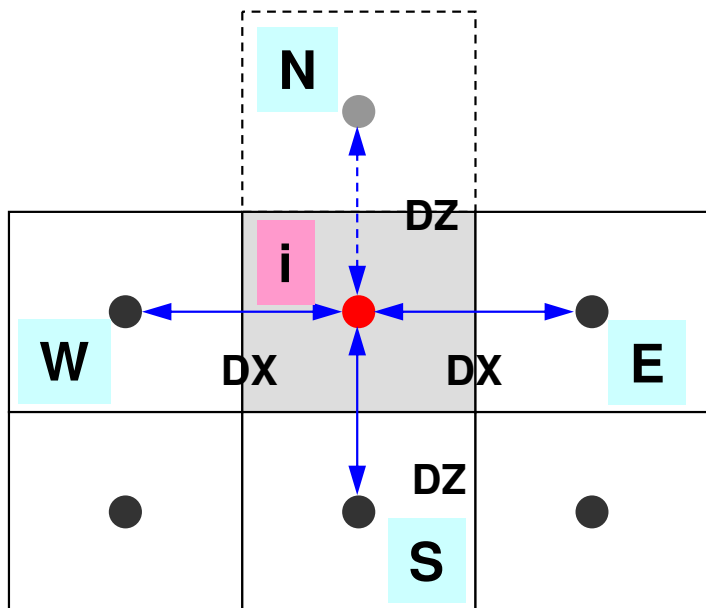
$$-\left[\sum_k \frac{S_{ik}}{d_{ik} + d_{ki}} \right] \phi_i + \left[\sum_k \frac{S_{ik}}{d_{ik} + d_{ki}} \phi_k \right] = -V_i \dot{Q}_i$$

D(対角成分)

AMAT
(非対角成分)

BFORCE
(右辺)

ディリクレ条件 ($\phi=0@Z=Z_{\max}$)



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

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

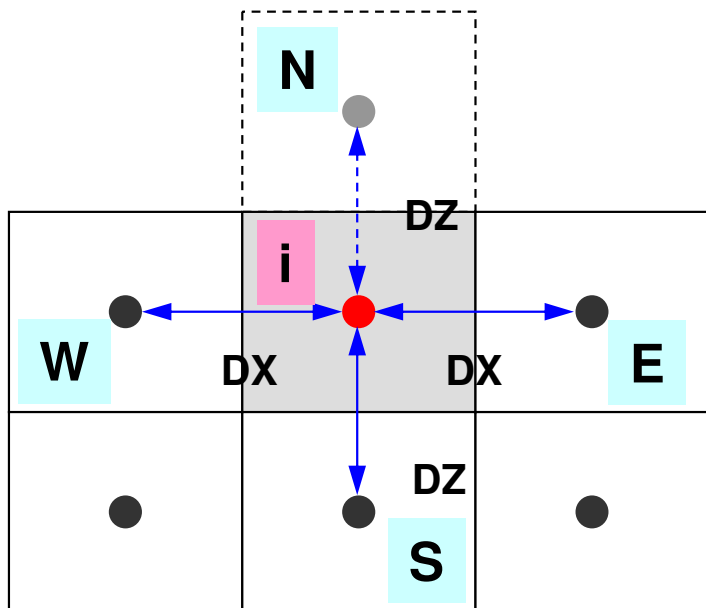
D (対角成分)

AMAT
(非対角成分)

BFORCE
(右辺)

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

ディリクレ条件 ($\phi=0@Z=Z_{\max}$)



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

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

D (対角成分)

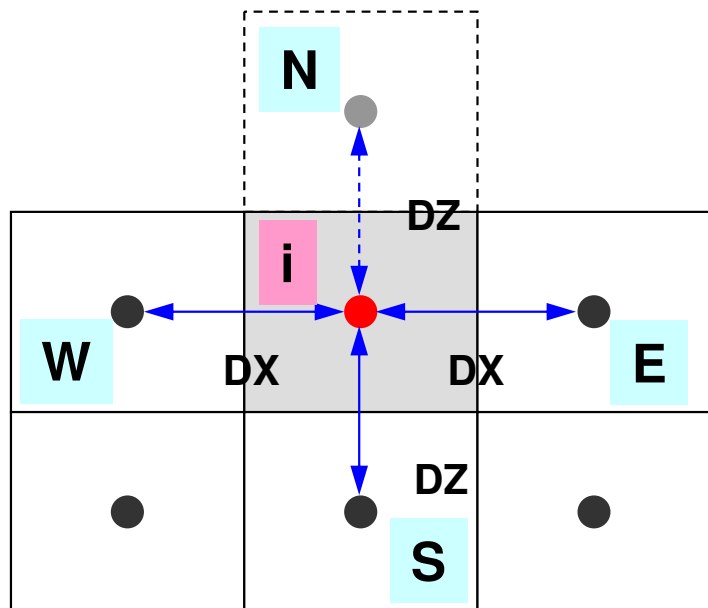
AMAT
(非対角成分)

BFORCE
(右辺)

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

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

ディリクレ条件 ($\phi=0@Z=Z_{\max}$)



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

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

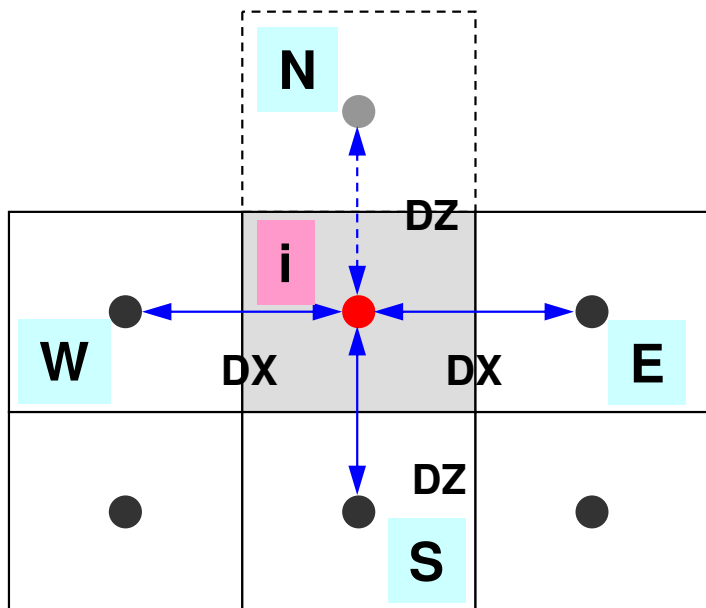
D (対角成分)

AMAT
(非対角成分)

BFORCE
(右辺)

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

ディリクレ条件 ($\phi=0@Z=Z_{\max}$)



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

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

D (対角成分)

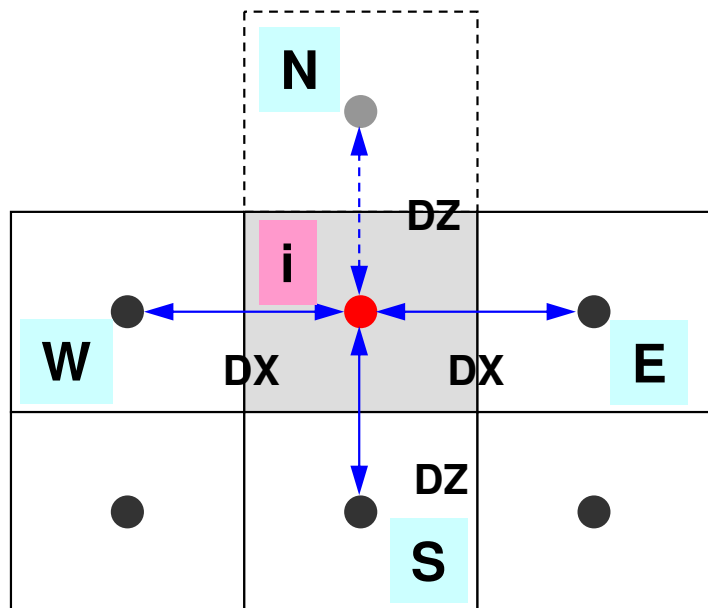
AMAT
(非対角成分)

BFORCE
(右辺)

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

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

ディリクレ条件 ($\phi=0@Z=Z_{\max}$)



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

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

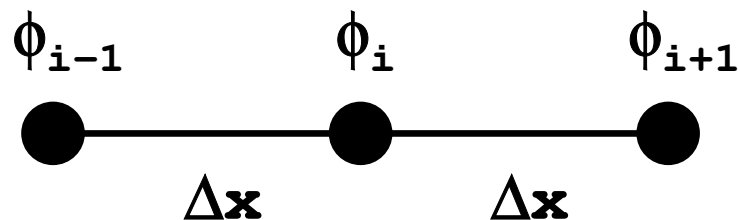
D (対角成分)

AMAT
(非対角成分)

BFORCE
(右辺)

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

```
do ib= 1, ZmaxGELtot
  icel= ZmaxCEL(ib)
  coef= 2. d0 * RDZ * ZAREA
  D(icel)= D(icel) - coef
enddo
```

Taylor展開

Taylor Series Expansion

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

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

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

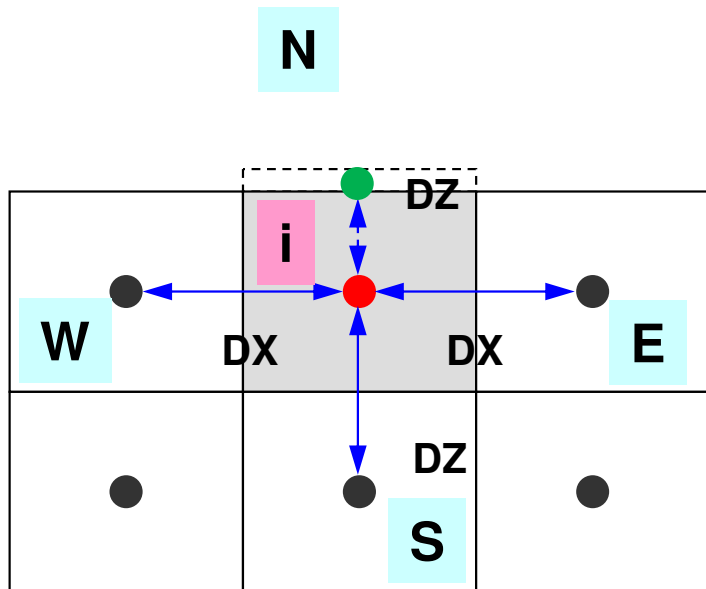
$$\frac{\phi_{i-1} - 2\phi_i + \phi_{i+1}}{(\Delta x)^2} = \left(\frac{\partial^2 \phi}{\partial x^2} \right)_i + \frac{(\Delta x)^2}{12} \left(\frac{\partial^4 \phi}{\partial x^4} \right)_i \dots$$

打切誤差: 2次オーダー
2次精度

Δx が均等でなければ, 1次
 またはより低次の精度

ディリクレ条件 ($\phi=0@Z=Z_{\max}$)

「N」が薄い場合 ($=\varepsilon$), 1次精度もしくはそれ以下



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

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

D (対角成分)

AMAT
(非対角成分)

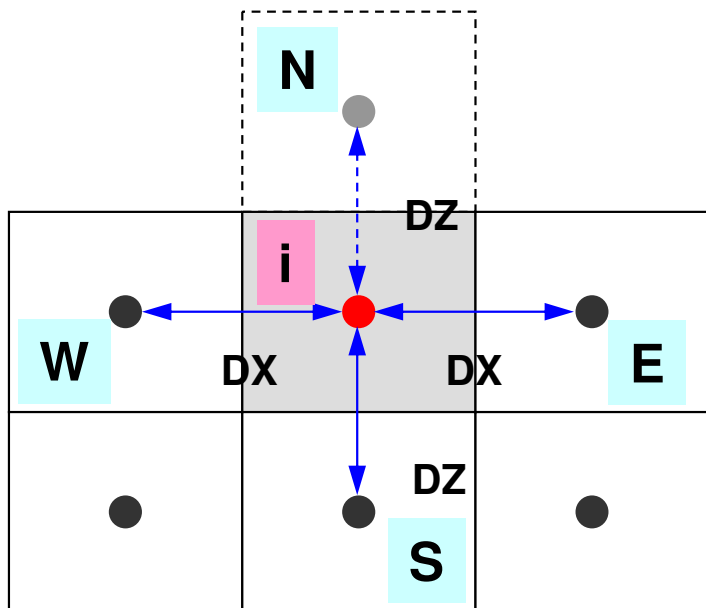
BFORCE
(右辺)

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

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

ディリクレ条件 ($\phi=0@Z=Z_{\max}$)

鏡像適用の場合: 「N」が薄い場合 ($=\varepsilon$) と同じ



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

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

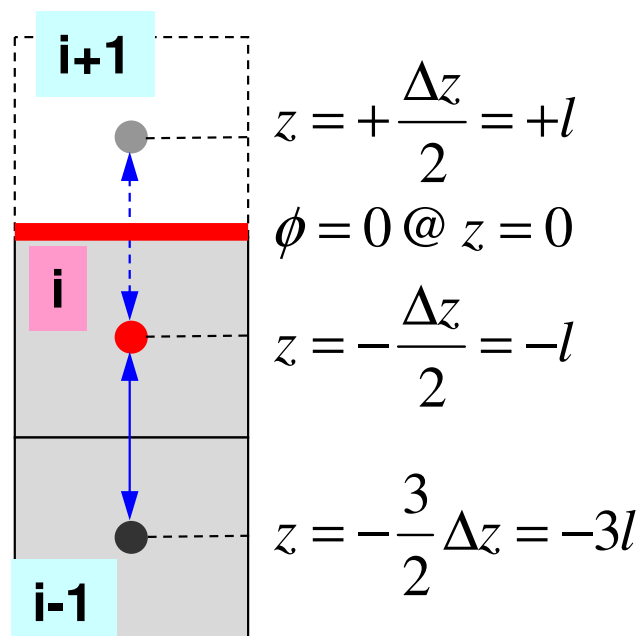
D (対角成分)

AMAT
(非対角成分)

BFORCE
(右辺)

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

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



二次精度近似 1次元の場合

2次元・3次元では
より複雑

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

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

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

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

プログラムの構成

```
program MAIN

use STRUCT
use PCG
use solver_PCG

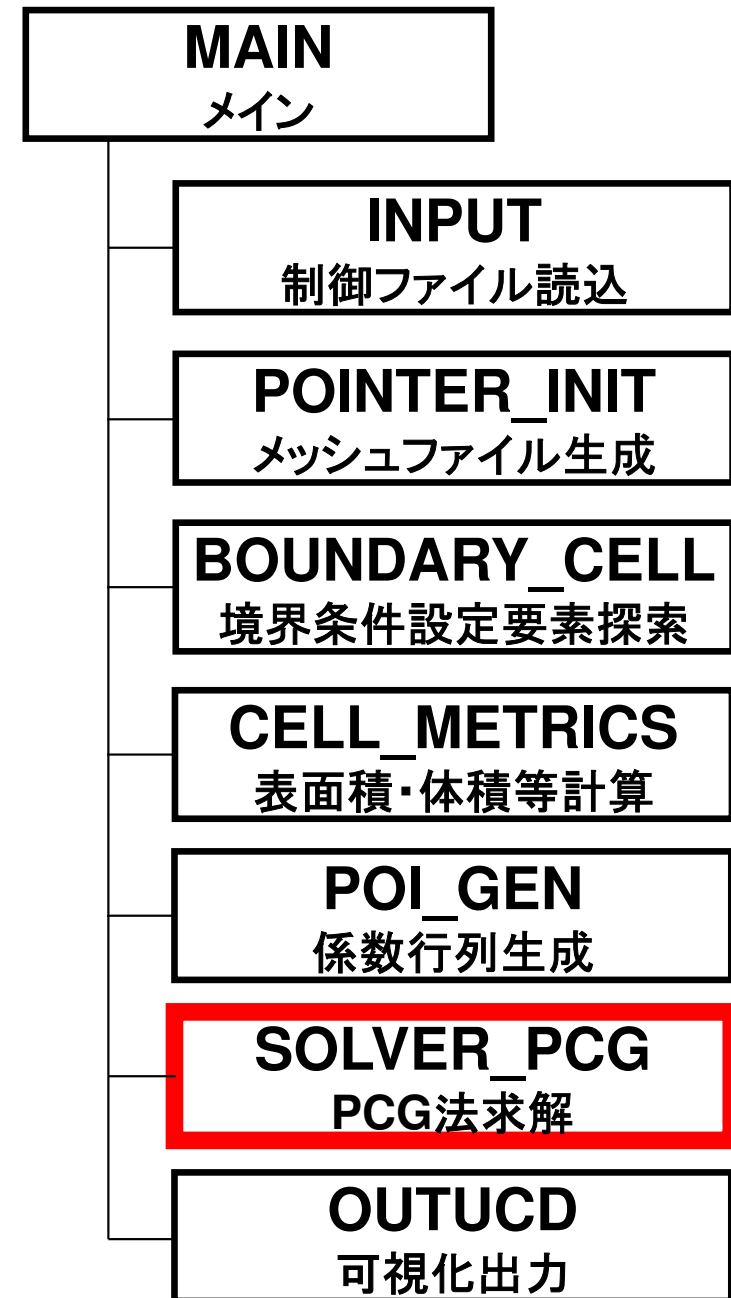
implicit REAL*8 (A-H, O-Z)

call INPUT
call POINTER_INIT
call BOUNDARY_CELL
call CELL_METRICS
call POI_GEN
PHI= 0. d0

call solve_PCG (...)

call OUTUCD

stop
end
```



- 背景
 - 有限体積法
 - 前処理付反復法
- **PCG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
 - **PCG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 (Preconditioning)
 - 点ヤコビ, 対角スケーリング
- PCG法

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

```
module solver_PCG
contains
```

```
subroutine solve_PCG                                &
&          ( N, NPLU, indexLU, itemLU, D, B, X,     &
&          AMAT, EPS, ITR, IER)
```

```
implicit REAL*8 (A-H, 0-Z)
```

```
real(kind=8), dimension(N)   :: D
real(kind=8), dimension(N)   :: B
real(kind=8), dimension(N)   :: X
real(kind=8), dimension(NPLU) :: AMAT
```

```
integer, dimension(0:N) :: indexLU
integer, dimension(NPLU) :: itemLU
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
```

solve_PCG

(1/6)

変数名:

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

$$W(i, 1) = W(i, R) \Rightarrow \{r\}$$

$$W(i, 2) = W(i, Z) \Rightarrow \{z\}$$

$$W(i, 2) = W(i, Q) \Rightarrow \{q\}$$

$$W(i, 3) = W(i, P) \Rightarrow \{p\}$$

$$W(i, 4) = W(i, DD) \Rightarrow \{1/d\}$$

solve_PCG (2/6)

```
!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

    do i= 1, N
        W(i, DD) = 1. d0/D(i)
    enddo
!C====
```

対角成分の逆数(前処理用)

その都度、除算をすると効率が悪い
ため、予め配列に格納。嘗ては除算と加
減乗算は10:1と言われていたが最近
はそれほどでもない。

solve_PCG (3/6)

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

do i= 1, N
  VAL= D(i)*X(i)
  do k= indexLU(i-1)+1, indexLU(i)
    VAL= VAL + AMAT(k)*X(itemLU(k))
  enddo
  W(i,R)= B(i) - VAL
enddo

BNRM2= 0.0D0
do i= 1, N
  BNRM2 = BNRM2 + B(i) **2
enddo

!C===
```

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

$$\text{BNRM2} = |\mathbf{b}|^2$$

あとで収束判定に使用

```

ITR= N
do L= 1, ITR

!C-- {z} = {M-1} {r} |
do i= 1, N
    W(i, Z) = W(i, R) * W(I, DD)
enddo

!C-- RHO = {r} {z} |
RHO = 0. d0
do i= 1, N
    RHO = RHO + W(i, R) * W(i, Z)
enddo

!C-- {p} = {z} if ITER=1
!C-- BETA = RHO / RHO1 otherwise

if ( L.eq.1 ) then
do i= 1, N
    W(i, P) = W(i, Z)
enddo
else
    BETA = RHO / RHO1
do i= 1, N
    W(i, P) = W(i, Z) + BETA * W(i, P)
enddo
endif
endif

```

solve_PCG (4/6)

```

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

```

```

!C-- {q} = [A] {p}

do i= 1, N
  VAL= D(i)*W(i, P)
  do k= indexLU(i-1)+1, indexLU(i)
    VAL= VAL + AMAT(k)*W(itemLU(k), P)
  enddo
  W(i, Q) = VAL
enddo

!C-- ALPHA= RHO / {p} {q}

C1= 0. d0
do i= 1, N
  C1= C1 + W(i, P)*W(i, Q)
enddo
ALPHA= RHO / C1

!C-- {x} = {x} + ALPHA*{p} |
!C-- {r} = {r} - ALPHA*{q} |

do i= 1, N
  X(i) = X(i) + ALPHA * W(i, P)
  W(i, R) = W(i, R) - ALPHA * W(i, Q)
enddo

```

solve_PCG (5/6)

```

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_PCG (6/6)

```

do L= 1, ITR
(....)
  DNRM2= 0. d0
  do i= 1, N
    DNRM2= DNRM2 + W(i, R)**2
  enddo

  ERR = dsqrt(DNRM2/BNRM2)
  if (ERR .lt. EPS) then
    IER = 0
    goto 900
  else
    RH01 = RH0
  endif

  enddo
  IER= 1

900 continue

```

```

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_PCG (6/6)

```

do L= 1, ITR
(…)
  DNRM2= 0. d0
  do i= 1, N
    DNRM2= DNRM2 + W
  enddo

  ERR = dsqrt(DNRM2/BNRM2)
  if (ERR .lt. EPS) then
    IER = 0
    goto 900
  else
    RH01 = RH0
  endif

enddo
IER= 1

900 continue

```

$$\begin{aligned}
 \mathbf{r} &= \mathbf{b} - [\mathbf{A}]\mathbf{x} \\
 \text{DNRM2} &= \|\mathbf{r}\|^2 \\
 \text{BNRM2} &= \|\mathbf{b}\|^2 \\
 \text{ERR} &= \|\mathbf{r}\| / \|\mathbf{b}\|
 \end{aligned}$$

$$\text{ERR} = \sqrt{\frac{\text{DNorm2}}{\text{BNorm2}}} = \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} = \frac{\|\mathbf{b} - \mathbf{A}\mathbf{x}\|}{\|\mathbf{b}\|} \leq \text{Eps}$$

```

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

```

else

$$\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$$

$$\mathbf{p}^{(i)} = \mathbf{z}^{(i-1)} + \beta_{i-1} \mathbf{p}^{(i-1)}$$

endif

$$\mathbf{q}^{(i)} = [\mathbf{A}]\mathbf{p}^{(i)}$$

$$\alpha_i = \rho_{i-1} / \mathbf{p}^{(i)} \mathbf{q}^{(i)}$$

$$\mathbf{x}^{(i)} = \mathbf{x}^{(i-1)} + \alpha_i \mathbf{p}^{(i)}$$

$$\mathbf{r}^{(i)} = \mathbf{r}^{(i-1)} - \alpha_i \mathbf{q}^{(i)}$$

check convergence $\|\mathbf{r}\|$

end

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

$$\mathbf{A}\mathbf{x} = \mathbf{b} \Rightarrow \alpha \mathbf{A}\mathbf{x} = \alpha \mathbf{b}$$

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x} \Rightarrow \mathbf{R} = \alpha \mathbf{b} - \alpha \mathbf{A}\mathbf{x} = \alpha \mathbf{r}$$