

並列アプリケーション開発法入門 (I)

有限体積法

2011年夏季集中講義

中島研吾

並列計算プログラミング(616-2057)・先端計算機演習I(616-4009)

本演習(P1)の目的

- 1CPU用シリアルアプリケーションを並列化する手順の習得。
- 並列アプリケーションを使用したシミュレーションの手順を一通り体験する。
 - データ生成(初期データ, 領域分割)
 - 計算
 - ポスト処理
- 対象
 - 有限体積法(または直接差分法)による熱伝導解析コード

並列計算の手順

- データ準備 (Pre-Processing)
 - 例: メッシュ生成, 領域分割
- 計算本体
- ポスト処理 (Post-Processing)
 - 例: 可視化, データマイニング

必要ファイルインストール: ECCS2008

```
$> cd <$E-EPS>
```

FORTRAN

```
$> cp /home03/skengon/Documents/class/EPSSummer/F/pla-f.tar .
```

C

```
$> cp /home03/skengon/Documents/class/EPSSummer/C/pla-c.tar .
```

```
$> tar xvf pla-f.tar (pla-c.tar)
```

```
$> ls
```

P1

```
$> cd P1
```

このディレクトリを<\$E-P1>と呼ぶ

```
$> ls
```

run/

実行ディレクトリ

serial/

1-CPUを使用した計算コード

T2Kにも同じものがある

```
$> cd <$T-EPS>
```

FORTRAN

```
$> cp /home/t00000/EPSSummer/F/pla-f.tar .
```

C

```
$> cp /home/t00000/EPSSummer/C/pla-c.tar .
```

```
$> tar xvf pla-f.tar (pla-c.tar)
```

```
$> ls
```

```
    P1
```

```
$> cd P1
```

このディレクトリを<\$T-P1>と呼ぶ

Lustreを使いたい人は:

```
$> cd /lustre/t50xxx
```

.bash_profile

```
$> cd
```

```
*****
```

```
$> cp /home03/skengon/.bash_profile .
```

または ~/.bash_profileに以下の一行を挿入

```
export PATH=/home03/skengon/bin:$PATH
```

```
*****
```

```
$> source .bash_profile
```

```
$> printenv
```

PATHに「home03/skengon/bin」が入っていることを確認

次回からはログイン時に自動的に起動

- **有限体積法の概要**
- 1CPU用プログラム `eps_fvm`
 - メッシュ生成, 形状データ
 - 計算実行例
 - プログラムの内容

対象とするアプリケーションの概要

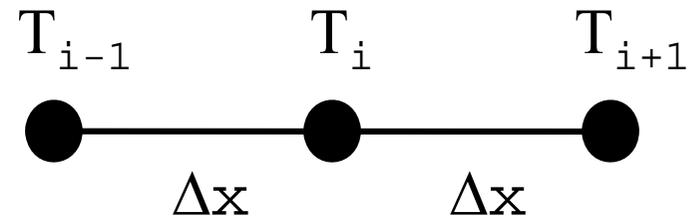
- 支配方程式: 三次元定常熱伝導方程式
 - 物性の温度依存性無し, 等方性

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

- 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
 - ディリクレ (温度固定), ノイマン (境界熱流束), 体積発熱
- 反復法による連立一次方程式解法
 - 共役勾配法 (CG) + 対角スケーリング (点ヤコビ) 前処理

一次元熱伝導方程式：中央差分 熱伝導率 λ 一定

$$\frac{d}{dx} \left(\lambda \frac{dT}{dx} \right)_i + Q = 0$$



$$\begin{aligned} \frac{d}{dx} \left(\lambda \frac{dT}{dx} \right)_i &= \frac{\left(\lambda \frac{dT}{dx} \right)_{i+1/2} - \left(\lambda \frac{dT}{dx} \right)_{i-1/2}}{\Delta x} = \frac{\lambda \frac{T_{i+1} - T_i}{\Delta x} - \lambda \frac{T_i - T_{i-1}}{\Delta x}}{\Delta x} \\ &= \lambda \left(\frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2} \right) \end{aligned}$$

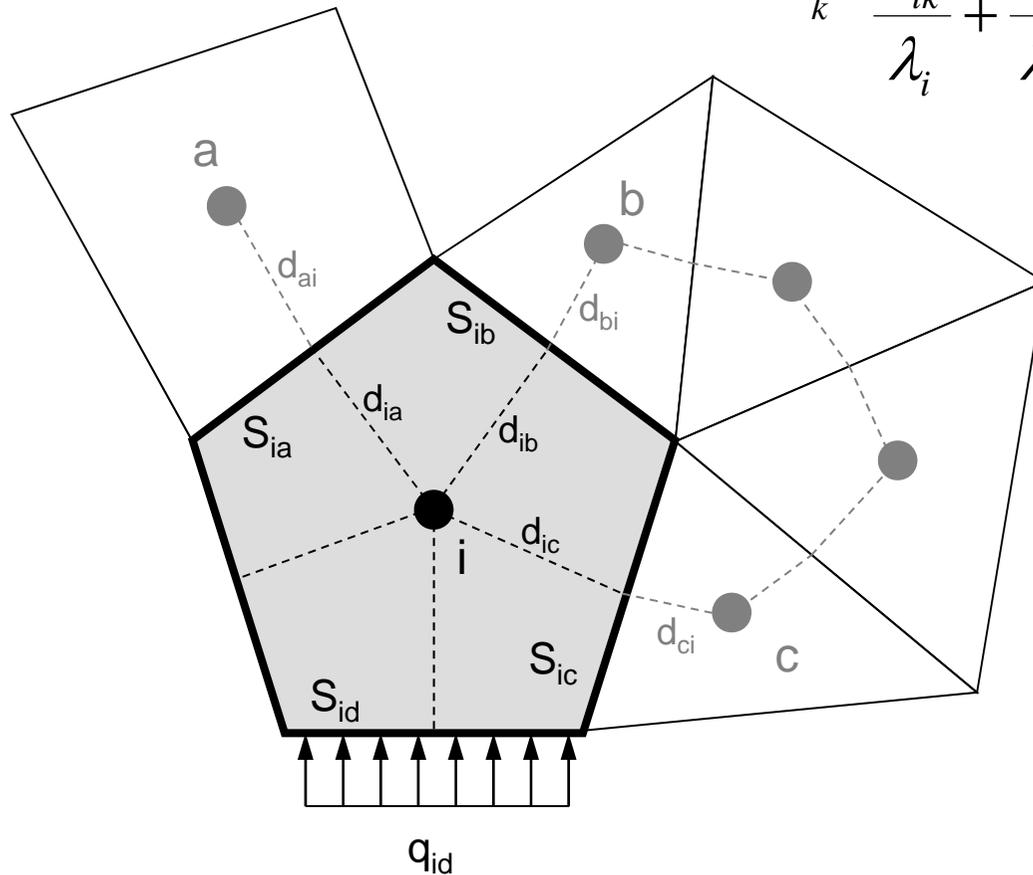
有限体積法による空間離散化

熱流束に関するつりあい式

隣接要素との熱伝導

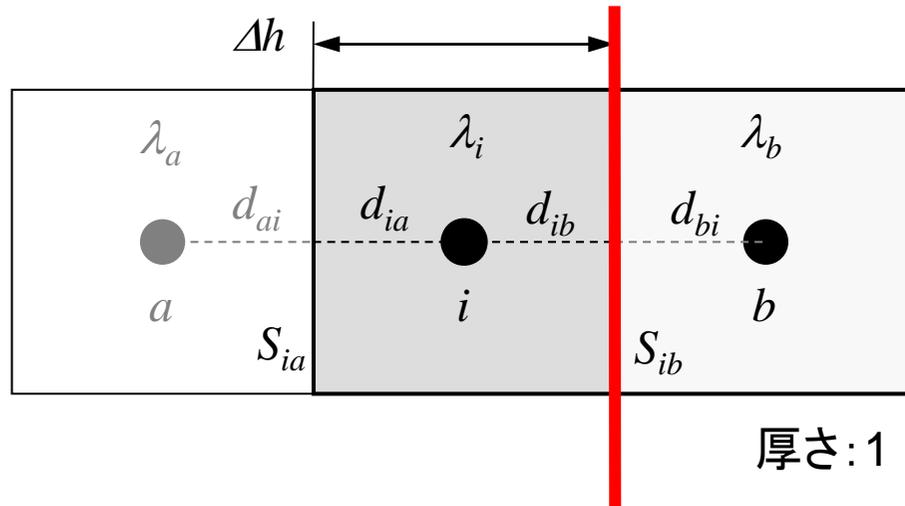
$$\sum_k \frac{S_{ik}}{d_{ik} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) + \sum_d S_{id} \dot{q}_{id} + V_i \dot{Q}_i = 0$$

要素境界面 体積発熱
通過熱流束



- λ : 熱伝導率
- 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 : 表面フラックス
- Q : 体積発熱

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

各メッシュの熱伝導率: λ_i

この面を通過する熱量: $Q_{S_{ib}}$

$$Q_{S_{ib}} = -\lambda \frac{T_b - T_i}{d_{ib} + d_{bi}} \cdot S_{ib} = -\frac{T_b - T_i}{\frac{d_{ib}}{\lambda} + \frac{d_{bi}}{\lambda}} \cdot S_{ib}$$

$$Q_{S_{ib}} = -\frac{T_b - T_i}{\frac{d_{ib}}{\lambda_i} + \frac{d_{bi}}{\lambda_b}} \cdot S_{ib}$$

熱伝導率が一樣の場合

フーリエの法則

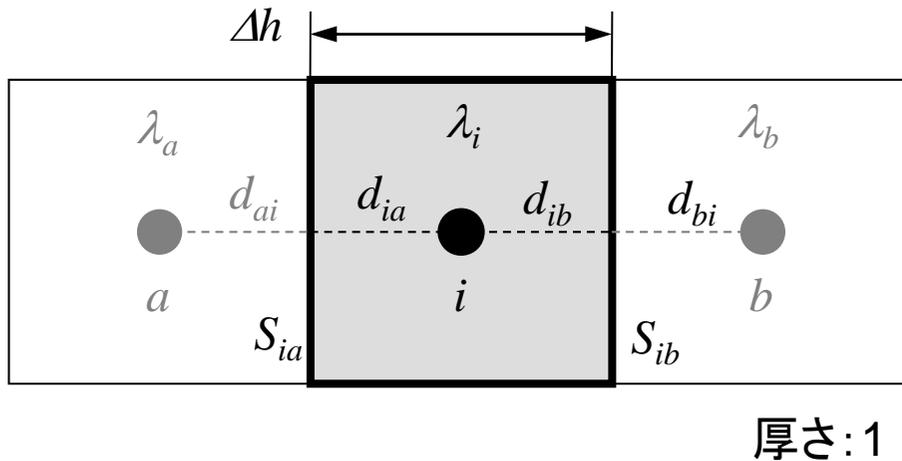
面を通過する熱量

= -(熱伝導率) × (温度勾配)

熱伝導率が要素ごとに

異なる場合: 調和平均

一次元差分法との比較(2/3)



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

各メッシュの熱伝導率: λ_i

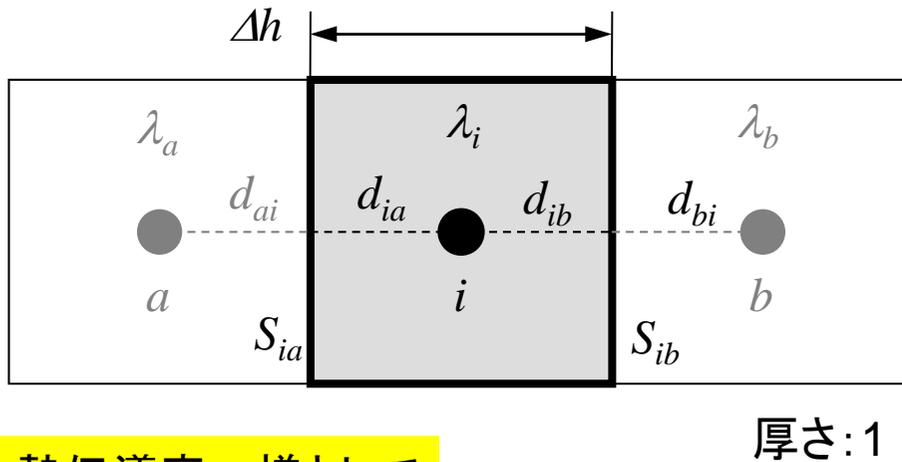
$$\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) + \sum_d S_{id} \dot{q}_{id} + V_i \dot{Q}_i = 0$$

両辺を V_i で割る:

$$\frac{1}{V_i} \sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) + \frac{1}{V_i} \sum_d S_{id} \dot{q}_{id} + \dot{Q}_i = 0$$

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

各メッシュの熱伝導率: λ_i

熱伝導率一様として

$$\begin{aligned}
 \frac{1}{V_i} \sum_k \frac{S_{ik}}{d_{ik} + d_{ki}} (T_k - T_i) &= \frac{1}{(\Delta h)^2} \sum_{k=a,b} \frac{\Delta h}{\frac{\Delta h}{2} + \frac{\Delta h}{2}} (T_k - T_i) \\
 &= \frac{1}{(\Delta h)^2} \sum_{k=a,b} \frac{\Delta h}{\frac{\Delta h}{2} + \frac{\Delta h}{2}} (T_k - T_i) = \frac{1}{(\Delta h)^2} \sum_{k=a,b} \frac{\Delta h}{\lambda} (T_k - T_i) = \frac{1}{(\Delta h)^2} \sum_{k=a,b} \lambda (T_k - T_i) \\
 &= \lambda \cdot \left[\frac{1}{(\Delta h)^2} (T_a - T_i) + \frac{1}{(\Delta h)^2} (T_b - T_i) \right] = \lambda \cdot \left[\frac{T_a - 2T_i + T_b}{(\Delta h)^2} \right]
 \end{aligned}$$

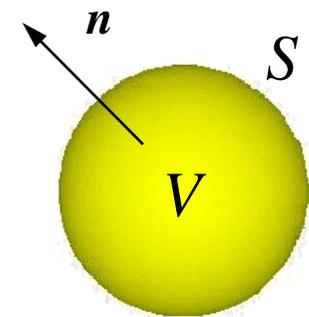
有限体積法

- セル中心型有限体積法
 - 「直接差分法」
 - ガウスグリーン型有限体積法
- 任意形状の要素を扱うことが可能
 - 均等メッシュでない場合は空間について一次精度
- 各要素の体積，要素間の接続関係（接触面積，重心（本当は「外心」）から接触面への距離）などを予め指定する必要があるため，メッシュ作成は多少面倒
- 境界面，ディリクレ境界条件（温度固定）の扱いが問題

ガウスの定理: Gauss's Theorem

$$\int_V \left(\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} + \frac{\partial W}{\partial z} \right) dV = \int_S (Un_x + Vn_y + Wn_z) dS$$

- 三次元デカルト座標 (x, y, z)
- 滑らかな閉曲面 S によって囲まれた V
- V 内で定義される, 3つの連続関数
 - $U(x, y, z), V(x, y, z), W(x, y, z)$
- 曲面 S 上で外向きに引いた法線ベクトル n
 - n_x, n_y, n_z : 方向余弦



グリーンの定理 (1/2)

- 以下のように仮定すると:

$$U = A \frac{\partial B}{\partial x}, \quad V = A \frac{\partial B}{\partial y}, \quad W = A \frac{\partial B}{\partial z}$$

- 以下が導かれる:

$$\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} + \frac{\partial W}{\partial z} = A \left(\frac{\partial^2 B}{\partial x^2} + \frac{\partial^2 B}{\partial y^2} + \frac{\partial^2 B}{\partial z^2} \right) + \left(\frac{\partial A}{\partial x} \frac{\partial B}{\partial x} + \frac{\partial A}{\partial y} \frac{\partial B}{\partial y} + \frac{\partial A}{\partial z} \frac{\partial B}{\partial z} \right)$$

- これを積分してガウスの定理を適用すると以下が得られる:

$$\begin{aligned} & \int_V A \left(\frac{\partial^2 B}{\partial x^2} + \frac{\partial^2 B}{\partial y^2} + \frac{\partial^2 B}{\partial z^2} \right) dV + \int_V \left(\frac{\partial A}{\partial x} \frac{\partial B}{\partial x} + \frac{\partial A}{\partial y} \frac{\partial B}{\partial y} + \frac{\partial A}{\partial z} \frac{\partial B}{\partial z} \right) dV \\ &= \int_S (Un_x + Vn_y + Wn_z) dS = \int_S A \left(\frac{\partial B}{\partial x} n_x + \frac{\partial B}{\partial y} n_y + \frac{\partial B}{\partial z} n_z \right) dS \end{aligned}$$

グリーンの定理 (2/2)

- (続き)

$$\int_S A \left(\frac{\partial B}{\partial x} n_x + \frac{\partial B}{\partial y} n_y + \frac{\partial B}{\partial z} n_z \right) dS = \int_S A \left(\frac{\partial B}{\partial x} \frac{\partial x}{\partial n} + \frac{\partial B}{\partial y} \frac{\partial y}{\partial n} + \frac{\partial B}{\partial z} \frac{\partial z}{\partial n} \right) dS$$

$$= \int_S A \frac{\partial B}{\partial n} dS \quad \frac{\partial B}{\partial n} : B \text{ の法線方向勾配}$$

- 結果として以下のようになる

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

ベクトル表記すると

- ガウスの定理

$$\int_V \nabla \cdot \mathbf{w} \, dV = \int_S \mathbf{w}^T \mathbf{n} \, dS$$

- グリーンの定理

$$\int_V v \Delta u \, dV = \int_S (v \nabla u)^T \mathbf{n} \, dS - \int_V (\nabla^T v)(\nabla u) \, dV$$

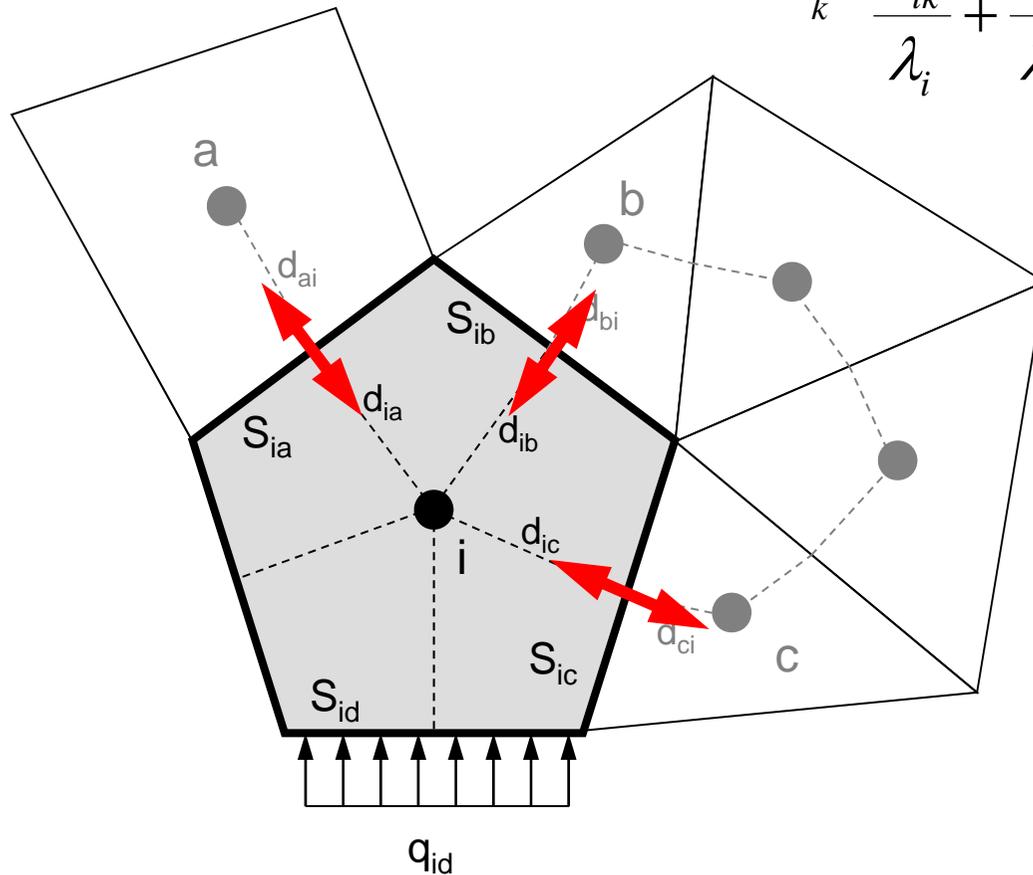
有限体積法による空間離散化

各要素の境界面を通過する熱量に着目

隣接要素との熱伝導

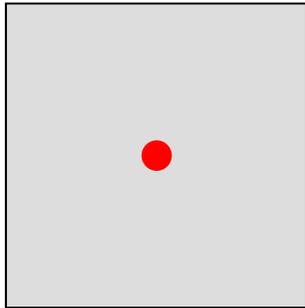
$$\sum_k \frac{S_{ik}}{d_{ik} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) + \sum_d S_{id} \dot{q}_{id} + V_i \dot{Q}_i = 0$$

要素境界面 体積発熱
通過熱流束

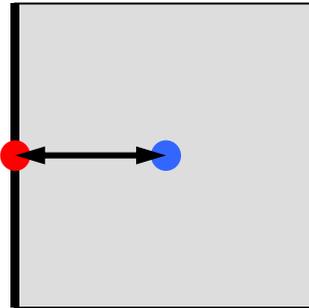


- λ : 熱伝導率
- V_i : 要素体積
- S : 表面面積
- d_{ij} : 要素中心から表面までの距離
- q : 表面フラックス
- Q : 体積発熱

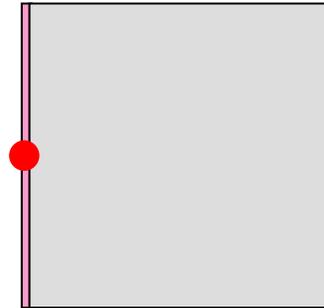
ディリクレ境界条件（境界温度固定）



単純に要素中心で指定
（精度悪化）

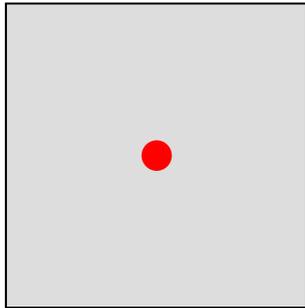


境界面からの等価な
フラックスを加算
（面積, 重心との距離必要）

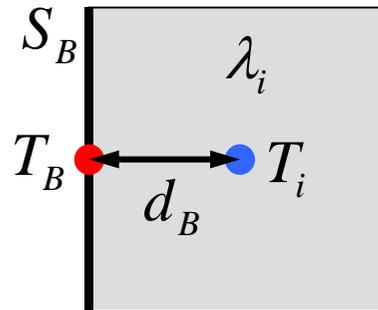


「薄い」要素を設定
（ $\sim 10^{-20}$ ）

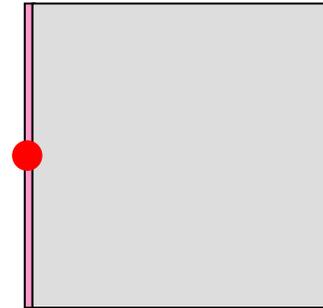
ディリクレ境界条件（境界温度固定）



単純に要素中心で指定
(精度悪化)



境界面からの等価な
フラックスを加算
(面積, 重心との距離必要)



「薄い」要素を設定
($\sim 10^{-20}$)

現在はこの方式を使用

$$q = \frac{S_B}{d_B / \lambda_i} (T_B - T_i)$$

有限体積法による空間離散化

熱流束に関するつりあい式: ディリクレ境界条件考慮

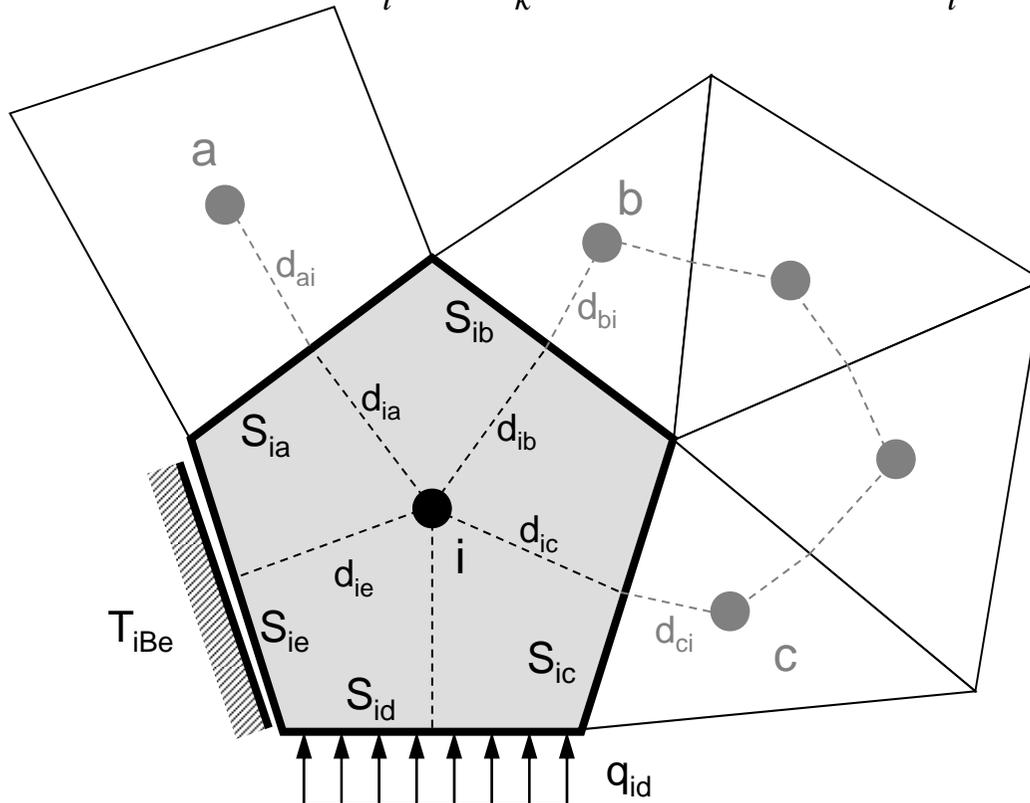
隣接要素との熱伝導

温度固定境界

$$\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) + \sum_e \frac{S_{ie}}{d_{ie}} (T_{iBe} - T_i) + \sum_d S_{id} \dot{q}_{id} + V_i \dot{Q}_i = 0$$

要素境界面
通過熱流束

体積発熱



- λ : 熱伝導率
- 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 : 表面フラックス
- Q : 体積発熱
- T_{iB} : 境界温度

有限体積法による空間離散化

隣接している要素の影響のみ受ける

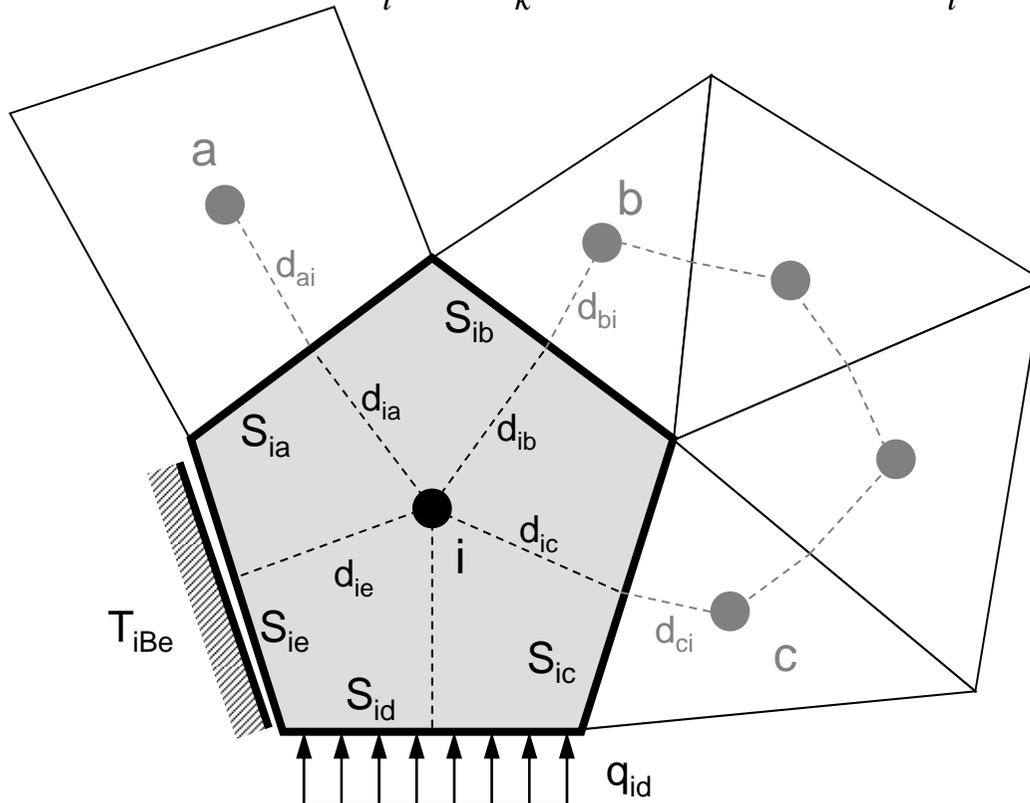
隣接要素との熱伝導

温度固定境界

$$\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) + \sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}} (T_{iBe} - T_i) + \sum_d S_{id} \dot{q}_{id} + V_i \dot{Q}_i = 0$$

要素境界面
通過熱流束

体積発熱



λ : 熱伝導率

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

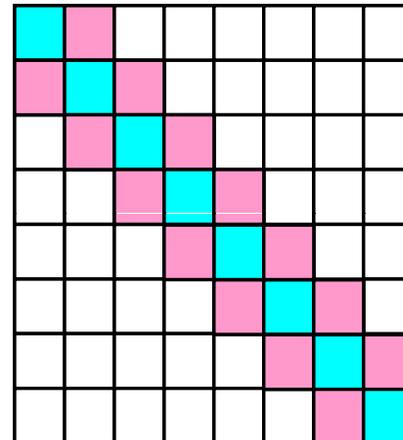
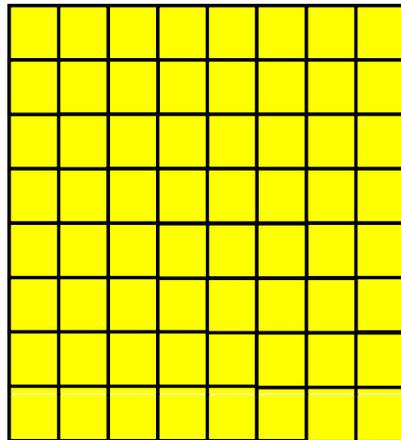
q : 表面フラックス

Q : 体積発熱

T_{iB} : 境界温度

係数行列は疎行列 (Sparse Matrix)

- 0が多い
- 通常「数値解析」などで扱うのは「密行列」
- 非ゼロ成分のみを記憶する方法
 - Compressed Row Storage, CRS (後述)
- 密行列と比較して取り扱いが困難

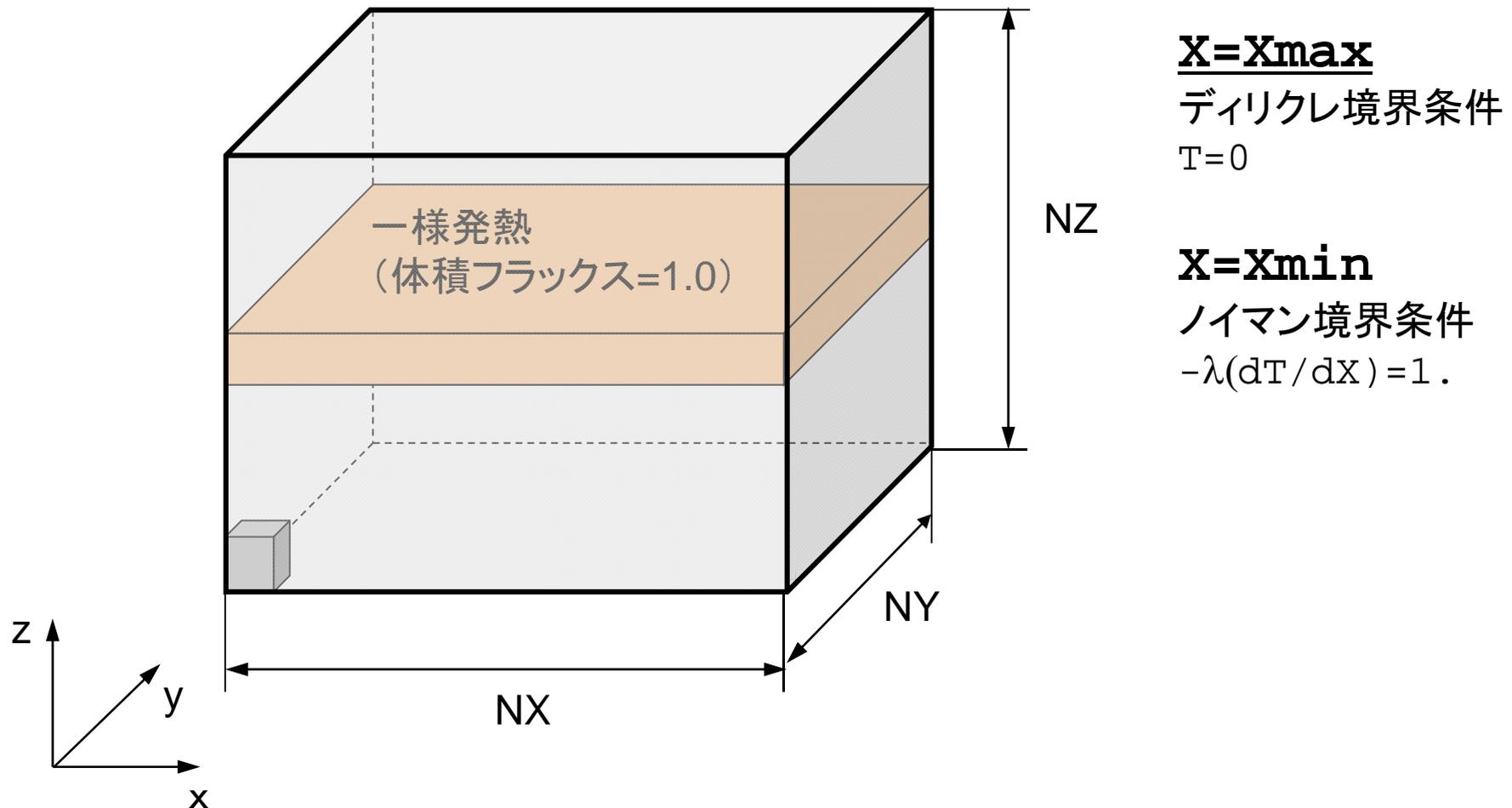


現行コードにおける制限

- 要素形状は六面体のみ。
 - 各要素は各辺長さ1の立方体
 - 全体形状, 境界条件等はメッシュジェネレータで規定
- 隣接要素数は要素各面に接する6面までとする。
 - プログラムを変更すれば増やすことは可能
- 境界条件
 - ディリクレ境界条件
 - ノイマン境界条件(面フラックス)
 - 体積発熱
- 要素番号は1から始まって連続していなければならない。

現在対象としている形状，境界条件

プログラムでは，後掲のデータ形式に従えば，任意の形状を扱うことができる



- 有限体積法の概要
- **1CPU用プログラム eps_fvm**
 - メッシュ生成, 形状データ
 - 計算実行例
 - プログラムの内容

メッシュ生成からやってみよう

```
$> cd <${E-P1}>/run
$> cat fvmmg.ctrl
      2  2  3
$> eps_fvm_mg

$> ls fvm_entire*
fvm_entire_mesh.dat
fvm_entire_mesh.inp
fvm_entire_mesh.inp_geo
```

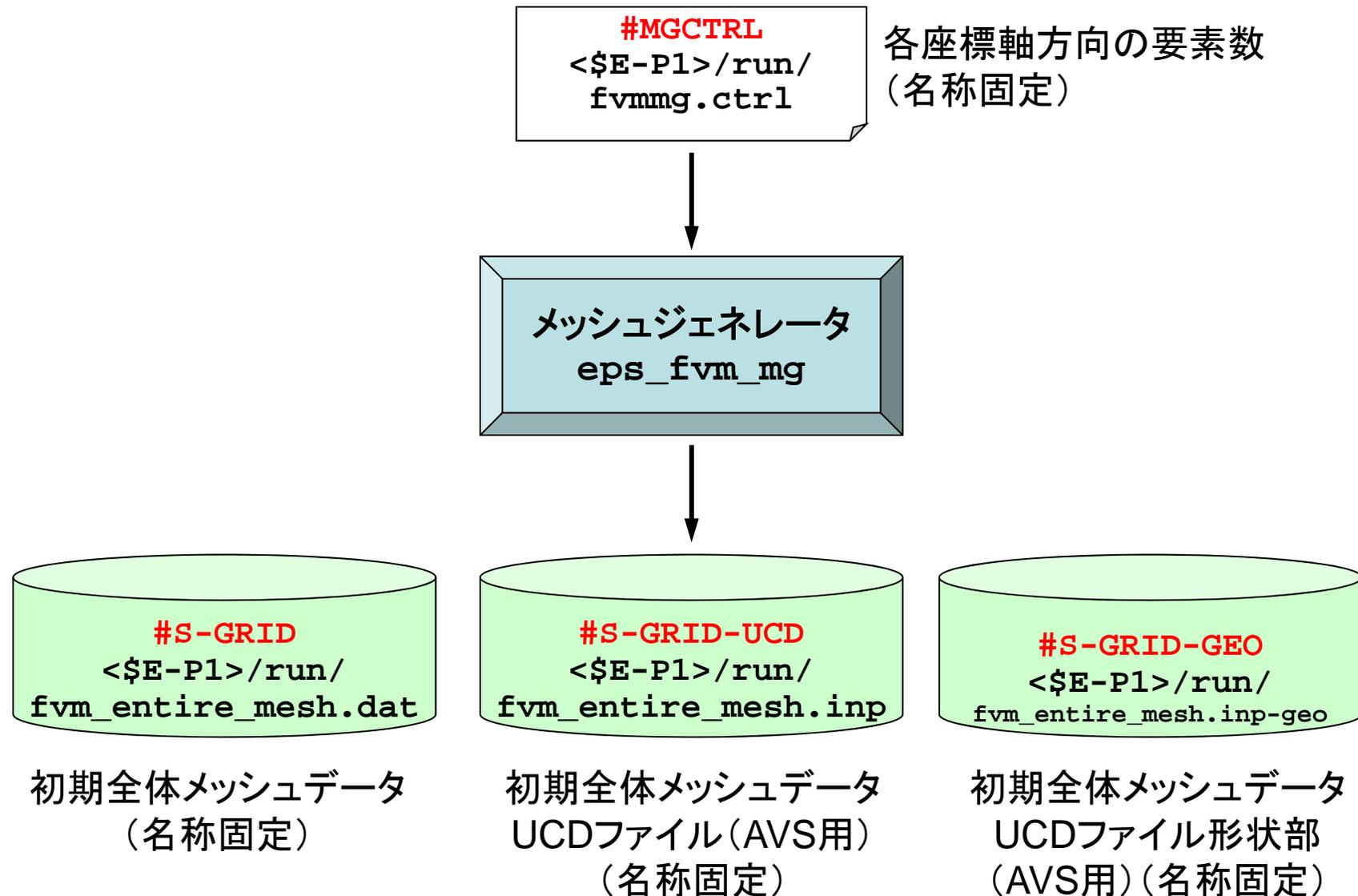
- 入力

- `fvmmg.ctrl`: 各座標軸方向の要素数 (NX,NY,NZ) **#MGCTRL**
 - この場合 $2 \times 2 \times 3 = 12$ 個の要素が生成される

- 出力

- `fvm_entire_mesh.dat`: 初期全体メッシュデータ **#S-GRID**
- `fvm_entire_mesh.inp`: AVS表示用ファイル **#S-GRID-UCD**
 - メッシュ形状をmicroAVSで表示するためのファイル
- `fvm_entire_mesh.inp-geo`: AVS表示用形状ファイル **#S-GRID-GEO**

メッシュ生成: 専用ツール使用



初期全体メッシュデータ(例)(1/6)

http://nkl.cc.u-tokyo.ac.jp/tutorial/part_tutorial

12						
1	1.000000E+00	1.000000E+00	5.000000E-01	5.000000E-01	5.000000E-01	各要素情報
2	1.000000E+00	1.000000E+00	1.500000E+00	5.000000E-01	5.000000E-01	
3	1.000000E+00	1.000000E+00	5.000000E-01	1.500000E+00	5.000000E-01	
4	1.000000E+00	1.000000E+00	1.500000E+00	1.500000E+00	5.000000E-01	
5	1.000000E+00	1.000000E+00	5.000000E-01	5.000000E-01	1.500000E+00	
6	1.000000E+00	1.000000E+00	1.500000E+00	5.000000E-01	1.500000E+00	
7	1.000000E+00	1.000000E+00	5.000000E-01	1.500000E+00	1.500000E+00	
8	1.000000E+00	1.000000E+00	1.500000E+00	1.500000E+00	1.500000E+00	
9	1.000000E+00	1.000000E+00	5.000000E-01	5.000000E-01	2.500000E+00	
10	1.000000E+00	1.000000E+00	1.500000E+00	5.000000E-01	2.500000E+00	
11	1.000000E+00	1.000000E+00	5.000000E-01	1.500000E+00	2.500000E+00	
12	1.000000E+00	1.000000E+00	1.500000E+00	1.500000E+00	2.500000E+00	
20						要素間コネクティビティ面に隣接する要素
1	2	1.000000E+00	5.000000E-01	5.000000E-01		
1	3	1.000000E+00	5.000000E-01	5.000000E-01		
1	5	1.000000E+00	5.000000E-01	5.000000E-01		
2	4	1.000000E+00	5.000000E-01	5.000000E-01		
2	6	1.000000E+00	5.000000E-01	5.000000E-01		
3	4	1.000000E+00	5.000000E-01	5.000000E-01		
3	7	1.000000E+00	5.000000E-01	5.000000E-01		
4	8	1.000000E+00	5.000000E-01	5.000000E-01		
5	6	1.000000E+00	5.000000E-01	5.000000E-01		
5	7	1.000000E+00	5.000000E-01	5.000000E-01		
5	9	1.000000E+00	5.000000E-01	5.000000E-01		
6	8	1.000000E+00	5.000000E-01	5.000000E-01		
6	10	1.000000E+00	5.000000E-01	5.000000E-01		
7	8	1.000000E+00	5.000000E-01	5.000000E-01		
7	11	1.000000E+00	5.000000E-01	5.000000E-01		
8	12	1.000000E+00	5.000000E-01	5.000000E-01		
9	10	1.000000E+00	5.000000E-01	5.000000E-01		
9	11	1.000000E+00	5.000000E-01	5.000000E-01		
10	12	1.000000E+00	5.000000E-01	5.000000E-01		
11	12	1.000000E+00	5.000000E-01	5.000000E-01		
6						ディリクレ境界条件 温度固定
2	1.000000E+00	5.000000E-01	0.000000E+00			
4	1.000000E+00	5.000000E-01	0.000000E+00			
6	1.000000E+00	5.000000E-01	0.000000E+00			
8	1.000000E+00	5.000000E-01	0.000000E+00			
10	1.000000E+00	5.000000E-01	0.000000E+00			
12	1.000000E+00	5.000000E-01	0.000000E+00			
6						ノイマン境界条件 境界熱流束
1	1.000000E+00	1.000000E+00				
3	1.000000E+00	1.000000E+00				
5	1.000000E+00	1.000000E+00				
7	1.000000E+00	1.000000E+00				
9	1.000000E+00	1.000000E+00				
11	1.000000E+00	1.000000E+00				
4						体積発熱
5	1.000000E+00					
6	1.000000E+00					
7	1.000000E+00					
8	1.000000E+00					

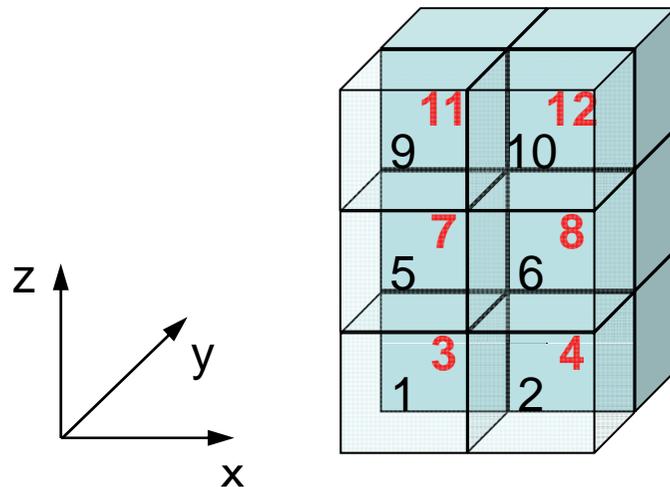
変数表

変数名	型	配列サイズ	内容
NODE_tot	I	-	総要素数
NODE_VOL(:)	R	NODE_tot	要素体積
NODE_COND(:)	R	NODE_tot	要素熱伝導率
NODE_XYZ(:)	R	3*NODE_tot	要素重心座標(3次元)
CONN_tot	I	-	コネクティビティ総数
CONN_node(:)	I	2*CONN_tot	コネクティビティ構成要素
CONN_COEF(:)	R	CONN_tot	コネクティビティ係数
FIX_NODE_tot	I	-	ディリクレ境界条件適用要素数
FIX_NODE_ID(:)	I	FIX_NODE_tot	ディリクレ境界条件適用要素番号
FIX_NODE_COEF(:)	R	FIX_NODE_tot	ディリクレ境界条件係数
FIX_NODE_VAL(:)	R	FIX_NODE_tot	ディリクレ境界条件値
SURF_NODE_tot	I	-	ノイマン境界条件適用要素数
SURF_NODE_ID(:)	I	SURF_NODE_tot	ノイマン境界条件適用要素番号
SURF_NODE_FLUX(:)	R	SURF_NODE_tot	ノイマン境界条件フラックス
BODY_NODE_tot	I	-	体積発熱境界条件適用要素数
BODY_NODE_ID(:)	I	BODY_NODE_tot	体積発熱境界条件適用要素番号
BODY_NODE_FLUX(:)	R	BODY_NODE_tot	体積発熱境界条件フラックス

初期全体メッシュデータ(例)(2/6)

各要素

要素番号	要素体積 NODE_VOL(i)	要素熱伝導率 NODE_COND(i)	要素中心x座標 NODE_XYZ(3*i-2)	要素中心y座標 NODE_XYZ(3*i-1)	要素中心z座標 NODE_XYZ(3*i)
12	要素数: NODE_tot				
1	1.000000E+00	1.000000E+00	5.000000E-01	5.000000E-01	5.000000E-01
2	1.000000E+00	1.000000E+00	1.500000E+00	5.000000E-01	5.000000E-01
3	1.000000E+00	1.000000E+00	5.000000E-01	1.500000E+00	5.000000E-01
4	1.000000E+00	1.000000E+00	1.500000E+00	1.500000E+00	5.000000E-01
5	1.000000E+00	1.000000E+00	5.000000E-01	5.000000E-01	1.500000E+00
6	1.000000E+00	1.000000E+00	1.500000E+00	5.000000E-01	1.500000E+00
7	1.000000E+00	1.000000E+00	5.000000E-01	1.500000E+00	1.500000E+00
8	1.000000E+00	1.000000E+00	1.500000E+00	1.500000E+00	1.500000E+00
9	1.000000E+00	1.000000E+00	5.000000E-01	5.000000E-01	2.500000E+00
10	1.000000E+00	1.000000E+00	1.500000E+00	5.000000E-01	2.500000E+00
11	1.000000E+00	1.000000E+00	5.000000E-01	1.500000E+00	2.500000E+00
12	1.000000E+00	1.000000E+00	1.500000E+00	1.500000E+00	2.500000E+00



```

read (IUNIT, '(10i10)') NODE_tot
do i= 1, NODE_tot
  read (IUNIT, '(i10,5e16.6)')
    ii, NODE_VOL(i), NODE_COND(i),
    (NODE_XYZ(3*i-3+k), k=1, 3)
enddo

```

初期全体メッシュデータ(例)(3/6)

要素間コネクティビティ:ある面に接続する2要素

```

20 コネクティビティ総数: CONN_tot
  1      2      1.000000E+00      5.000000E-01      5.000000E-01
  1      3      1.000000E+00      5.000000E-01      5.000000E-01
  1      5      1.000000E+00      5.000000E-01      5.000000E-01
  2      4      1.000000E+00      5.000000E-01      5.000000E-01
  2      6      1.000000E+00      5.000000E-01      5.000000E-01
  3      4      1.000000E+00      5.000000E-01      5.000000E-01
  3      7      1.000000E+00      5.000000E-01      5.000000E-01
  4      8      1.000000E+00      5.000000E-01      5.000000E-01
  5      6      1.000000E+00      5.000000E-01      5.000000E-01
  5      7      1.000000E+00      5.000000E-01      5.000000E-01
  5      9      1.000000E+00      5.000000E-01      5.000000E-01
  6      8      1.000000E+00      5.000000E-01      5.000000E-01
  6     10      1.000000E+00      5.000000E-01      5.000000E-01
  7      8      1.000000E+00      5.000000E-01      5.000000E-01
  7     11      1.000000E+00      5.000000E-01      5.000000E-01
  8     12      1.000000E+00      5.000000E-01      5.000000E-01
  9     10      1.000000E+00      5.000000E-01      5.000000E-01
  9     11      1.000000E+00      5.000000E-01      5.000000E-01
 10     12      1.000000E+00      5.000000E-01      5.000000E-01
 11     12      1.000000E+00      5.000000E-01      5.000000E-01

```

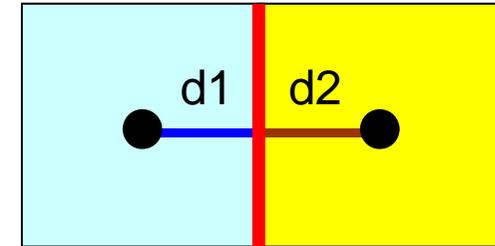
E1 E2 s:要素境界面積

d1:E1重心～
境界面距離

d2:E2重心～
境界面距離

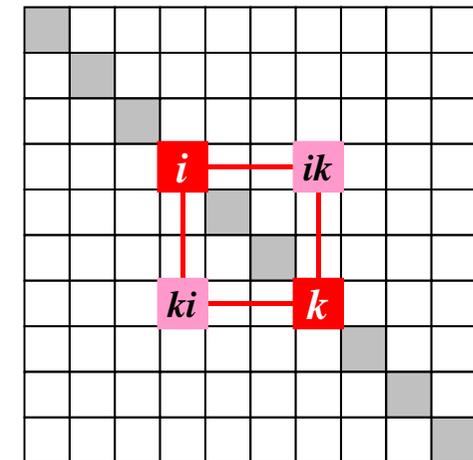
E1= CONN_NODE(2*ic-1)

E2= CONN_NODE(2*ic)



E1

E2



係数行列の非対角成分

```

read (IUNIT,'(10i10)') CONN_tot
do i= 1, CONN_tot
  read (IUNIT,'( 2i10, 3e16.6)')      &
    (CONN_NODE(2*i-2+k), k= 1, 2),    &
    AREA, d1, d2
enddo

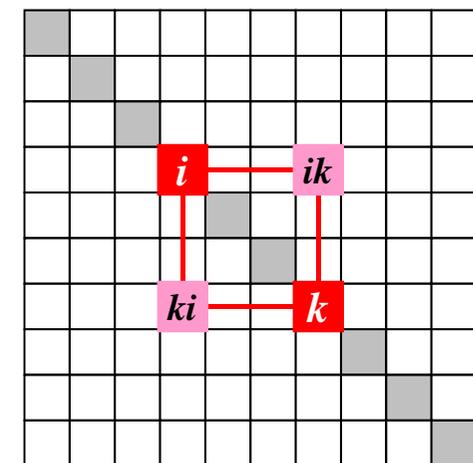
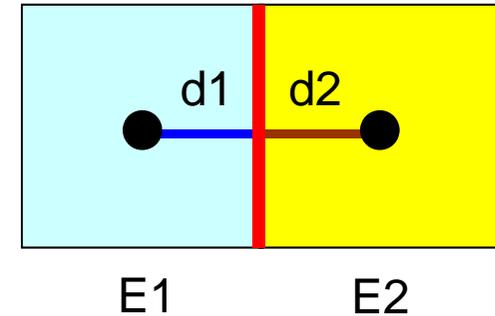
```

初期全体メッシュデータ(例)(3/6)

要素間コネクティビティ:5番要素に着目

1	5	1.0E+00	5.0E-01	5.0E-01
5	6	1.0E+00	5.0E-01	5.0E-01
5	7	1.0E+00	5.0E-01	5.0E-01
5	9	1.0E+00	5.0E-01	5.0E-01

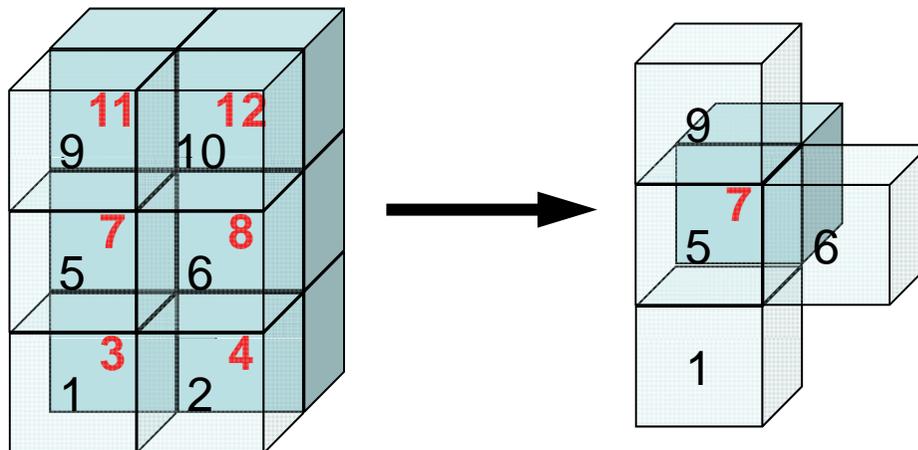
E1 **E2** **s:要素境界面積** **d1:E1重心~境界面距離** **d2:E2重心~境界面距離**



係数行列の非対角成分

各要素は一辺の長さ1の立方体:

- 要素境界面積は $1 \times 1 = 1$
- 各要素重心から境界面までの距離は0.50



初期全体メッシュデータ(例)(4/6)

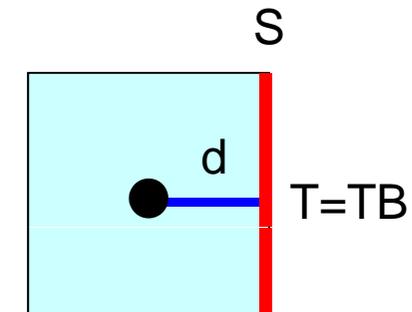
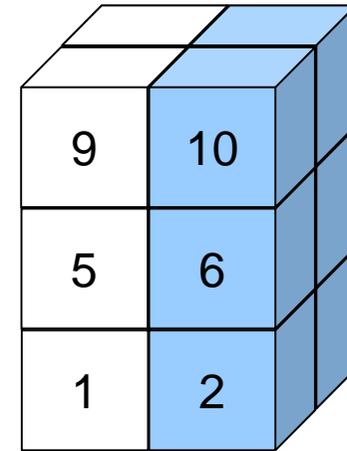
ディリクレ境界条件(表面温度固定)

6 **ディリクレ境界条件を与える要素数: FIX_NODE_tot**

2	1.000000E+00	5.000000E-01	0.000000E+00
4	1.000000E+00	5.000000E-01	0.000000E+00
6	1.000000E+00	5.000000E-01	0.000000E+00
8	1.000000E+00	5.000000E-01	0.000000E+00
10	1.000000E+00	5.000000E-01	0.000000E+00
12	1.000000E+00	5.000000E-01	0.000000E+00

要素番号 **境界面面積s** **境界面と要素重心距離d** **境界値TB**
FIX_NODE_ID(ib) **FIX_NODE_VAL(ib)**

```
read (IUNIT,'(10i10)') FIX_NODE_tot
do i= 1, FIX_NODE_tot
  read (IUNIT, '(i10, 3e16.6)')      &
    FIX_NODE_ID(i), S, d,           &
    FIX_NODE_VAL(i)
enddo
```



1つの「要素」に複数のディリクレ境界条件を与えることも実は可能

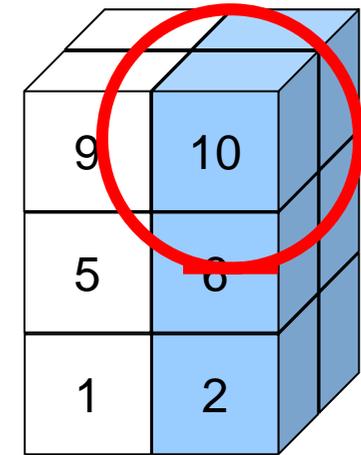
7 **ディリクレ境界条件を与える要素数: FIX_NODE_tot**

2	1.000000E+00	5.000000E-01	0.000000E+00
4	1.000000E+00	5.000000E-01	0.000000E+00
6	1.000000E+00	5.000000E-01	0.000000E+00
8	1.000000E+00	5.000000E-01	0.000000E+00
10	1.000000E+00	5.000000E-01	0.000000E+00
12	1.000000E+00	5.000000E-01	0.000000E+00
10	1.000000E+00	5.000000E-01	1.000000E+01

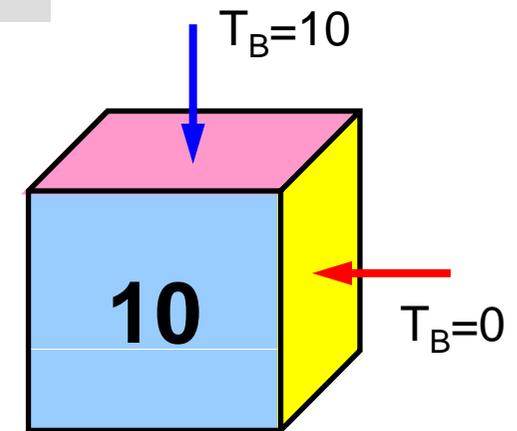
要素番号 境界面面積s 境界面と要素重心距離d 境界値 T_B

FIX_NODE_ID(ib)

FIX_NODE_VAL(ib)



ある表面の温度を規定するだけであって、その要素の温度自体(要素中心)を規定するわけではない。「要素数」というよりは「面数」という方がより正確。



有限体積法による空間離散化

熱流束に関するつりあい式

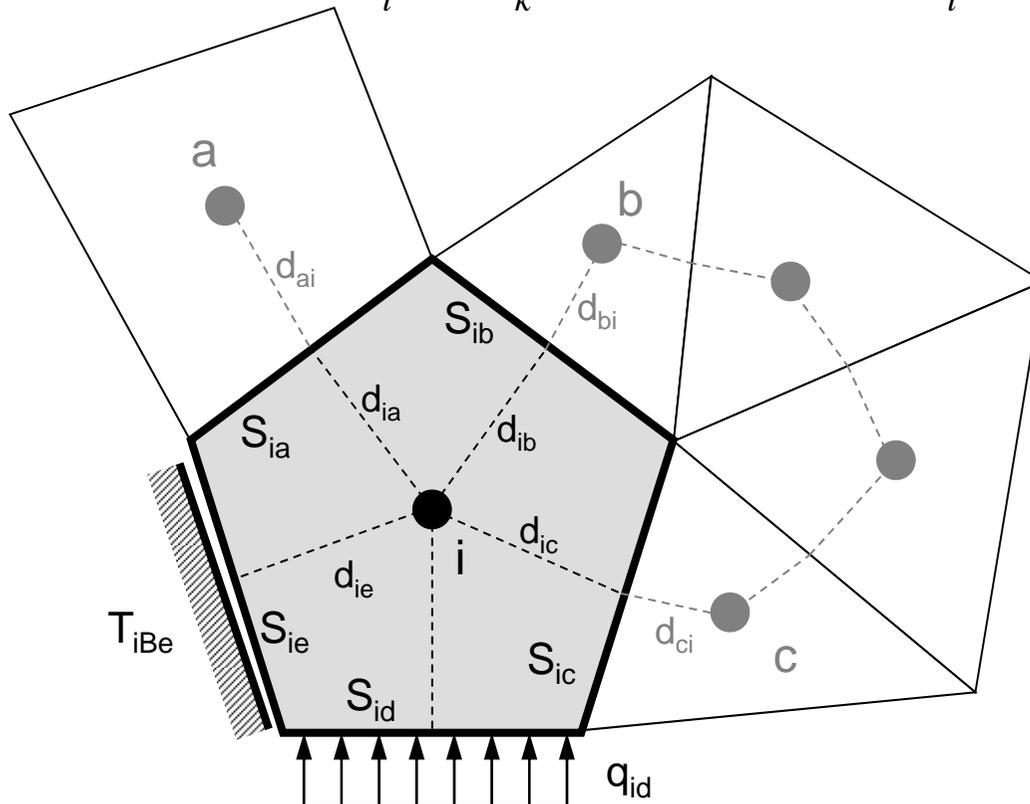
隣接要素との熱伝導

温度固定境界

$$\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) + \sum_e \frac{S_{ie}}{d_{ie}} (T_{iBe} - T_i) + \sum_d S_{id} \dot{q}_{id} + V_i \dot{Q}_i = 0$$

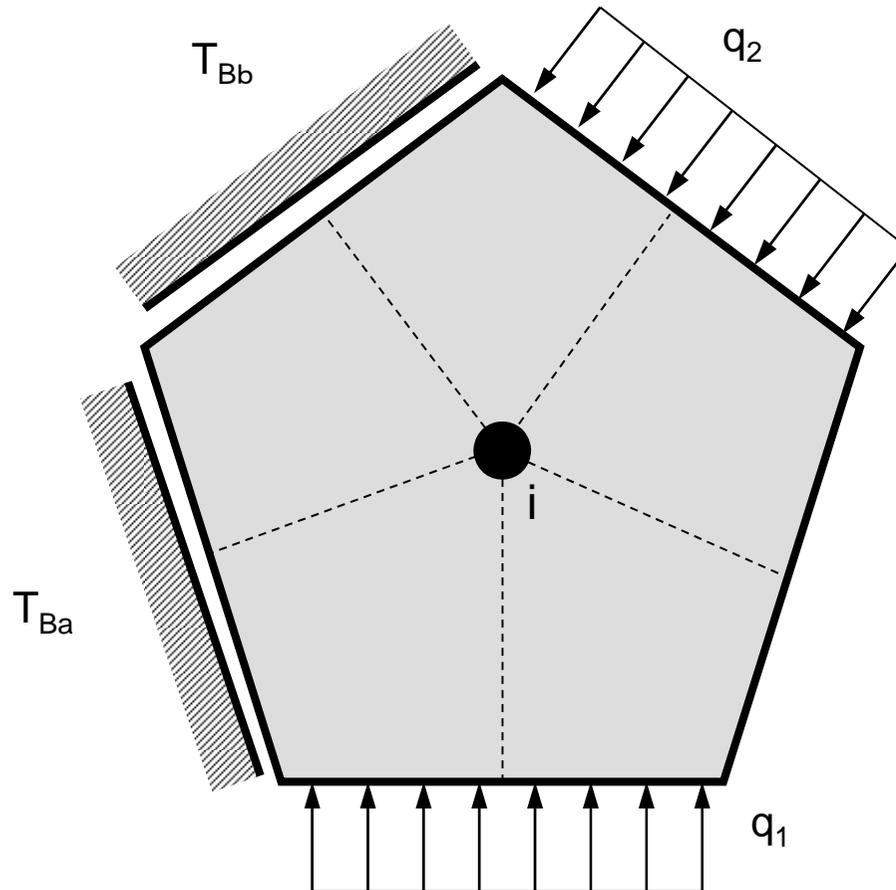
要素境界面
通過熱流束

体積発熱



- λ : 熱伝導率
- 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 : 表面フラックス
- Q : 体積発熱
- T_{iB} : 境界温度

各要素に適用できる境界条件の数に制限は無い



初期全体メッシュデータ(例)(5/6)

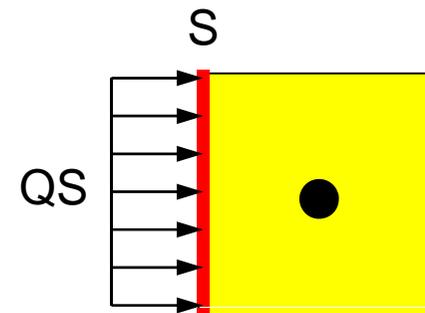
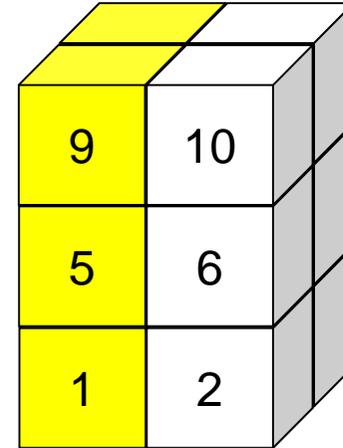
ノイマン境界条件(表面熱フラックス: W/m^2)

6 ノイマン境界条件を与える要素数: SURF_NODE_tot

1	1.000000E+00	1.000000E+00
3	1.000000E+00	1.000000E+00
5	1.000000E+00	1.000000E+00
7	1.000000E+00	1.000000E+00
9	1.000000E+00	1.000000E+00
11	1.000000E+00	1.000000E+00

要素番号 境界表面積s 表面フラックスQS
SURF_NODE_ID(ib)

```
read (IUNIT,'(10i10)') SURF_NODE_tot
do i= 1, SURF_NODE_tot
  read (IUNIT, '(i10, 3e16.6)')      &
    SURF_NODE_ID(i), S, QS
enddo
```



1つの要素に複数のノイマン境界条件を適用することは可能(複数の面に異なった熱流束を与える)。

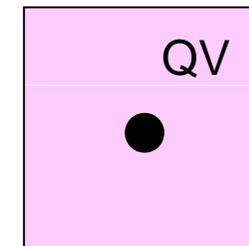
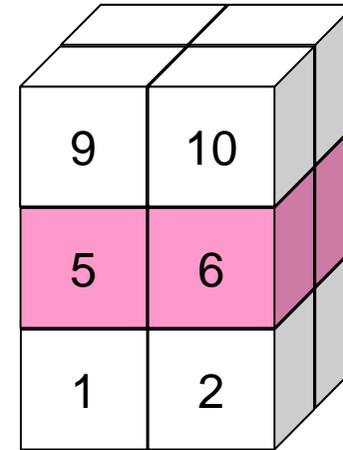
初期全体メッシュデータ(例)(6/6)

体積発熱(W/m³)

```
4  体積発熱を与える要素数
5      1.000000E+00
6      1.000000E+00
7      1.000000E+00
8      1.000000E+00
要素番号 体積フラックスqv
```

```
read (IUNIT,'(10i10)') BODY_NODE_tot
do i= 1, BODY_NODE_tot
  read (IUNIT, '(i10, 3e16.6)') icel, qv
enddo
```

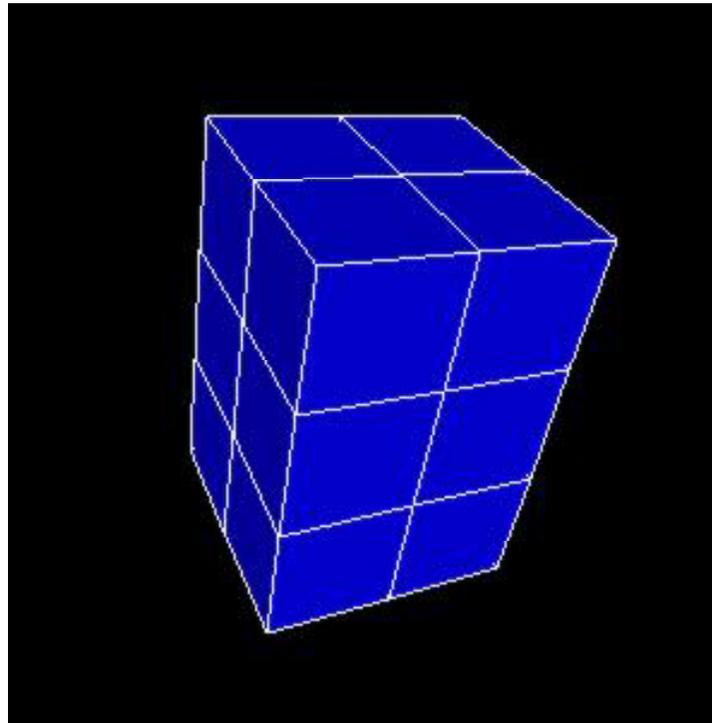
同じ要素に対して、複数回体積発熱を指定した場合には、一番最後に指定した値が適用される。



MicroAVSによる表示

#S-GRID-UCD:

`<$FVM>/run/fvm_entire_mesh.inp`



- AVS, microAVSのUCDファイルフォーマット (Unstructured Cell Data)を使用。
 - <http://www.kgt.co.jp/>

MicroAVSの使用

- 手順は以下
 - <http://nkl.cc.u-tokyo.ac.jp/10e/mavs/>

UCDフォーマットについて(1/4)

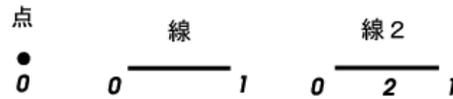
MicroAVSで扱うことのできる要素形状

要素の種類

キーワード

点

pt

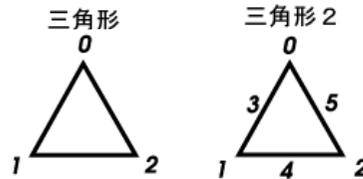


線

line

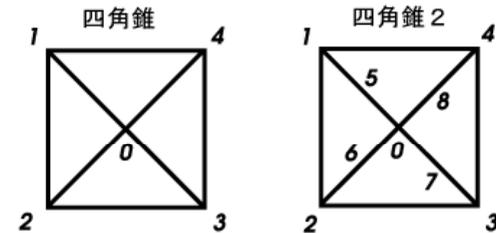
三角形

tri



四角形

quad

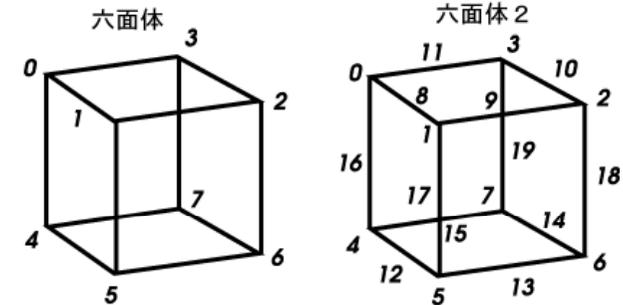


四面体

tet

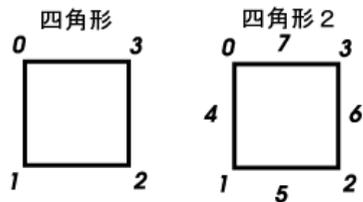
角錐

pyr



三角柱

prism



六面体

hex

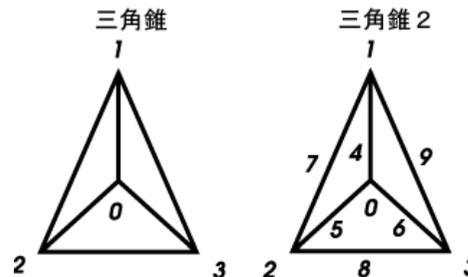
二次要素

line2

線2

三角形2

tri2



四角形2

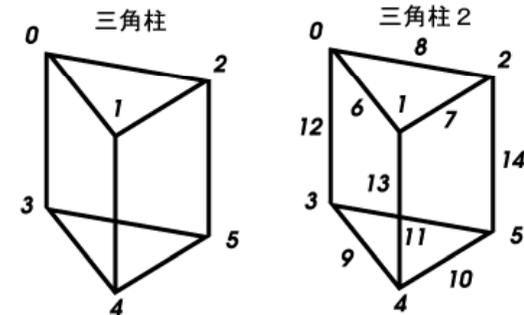
quad2

四面体2

tet2

角錐2

pyr2



三角柱2

prism2

六面体2

hex2

UCDフォーマットについて(2/4)

書式の概要

- ファイルの拡張子
 - データファイルの拡張子は “.inp”
- 書式
 - アスキーファイル
 - 時系列(複数ステップ)に対応したものが標準フォーマット
 - MicroAVS Ver.6.0(現在はVer.13.0)まで使用していた旧フォーマット(単一ステップデータのための書式)も読み込むことは可能
 - 実は一部これを使用している

UCDフォーマットについて(4/3)

書式の概要

(コメント行)
 (ステップ数)
 (データの繰り返しタイプ)
 (ステップ番号1) (コメント)
 (全節点数) (全要素数)
 (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)

・
 ・
 ・

(各節点のデータ数) (各要素のデータ数)
 (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) ……

・
 ・
 ・

(ステップ番号2) (コメント) (全節点数) (全要素数)

・
 ・

UCDフォーマットについて(4/4)

旧フォーマット

(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) ⋯⋯

⋮

UCDデータの例

新フォーマット

```

#
1
data
step1
36 12
 1 0.0 0.0 0.0
 2 1.0 0.0 0.0
 3 2.0 0.0 0.0
 4 0.0 1.0 0.0
 5 1.0 1.0 0.0
 6 2.0 1.0 0.0
...
31 0.0 1.0 3.0
32 1.0 1.0 3.0
33 2.0 1.0 3.0
34 0.0 2.0 3.0
35 1.0 2.0 3.0
36 2.0 2.0 3.0
 1 1 hex 1 2 5 4 10 11 14 13
 2 1 hex 2 3 6 5 11 12 15 14
 3 1 hex 4 5 8 7 13 14 17 16
...
10 1 hex 20 21 24 23 29 30 33 32
11 1 hex 22 23 26 25 31 32 35 34
12 1 hex 23 24 27 26 32 33 36 35
0 1
1 1
Mesh,
 1 1.00
 2 1.00
 3 1.00
 4 1.00
...
10 1.00
11 1.00
12 1.00

```

旧フォーマット

```

36 12 0 1 0
 1 0.0 0.0 0.0
 2 1.0 0.0 0.0
 3 2.0 0.0 0.0
 4 0.0 1.0 0.0
 5 1.0 1.0 0.0
 6 2.0 1.0 0.0
...
31 0.0 1.0 3.0
32 1.0 1.0 3.0
33 2.0 1.0 3.0
34 0.0 2.0 3.0
35 1.0 2.0 3.0
36 2.0 2.0 3.0
 1 1 hex 1 2 5 4 10 11 14 13
 2 1 hex 2 3 6 5 11 12 15 14
 3 1 hex 4 5 8 7 13 14 17 16
...
10 1 hex 20 21 24 23 29 30 33 32
11 1 hex 22 23 26 25 31 32 35 34
12 1 hex 23 24 27 26 32 33 36 35
1 1
Mesh,
 1 1.00
 2 1.00
 3 1.00
 4 1.00
...
10 1.00
11 1.00
12 1.00

```

- 有限体積法の概要
- 1CPU用プログラム `eps_fvm`
 - メッシュ生成, 形状データ
 - **計算実行例**
 - プログラムの内容

「eps_fvm」計算の実行

```
$> cd <$E-P1>/serial
$> make
...
$> cd ../run
$> ls eps_fvm
    eps_fvm

$> ./eps_fvm
```

ロードモジュールの生成確認

- 実行形式 `eps_fvm`は`<$FVM>/run` に生成される。
- 入力
 - `<$E-P1>/run/fvmmg.ctrl` **#MGCTRL**
 - `<$E-P1>/run/fvm_entire_mesh.dat` **#S-GRID**
 - ファイルを`<$T-P1>/run`に予め生成しておく必要がある。
- 出力
 - `<$E-P1>/run/fvm_entire_mesh_results.inp`(名称固定)
 - MicroAVS用結果出力 **#S-GRID-R-UCD**

シミュレーションコード: eps_fvm

```
#MGCTRL  
<$E-P1>/run/  
fvmmg.ctrl
```

各座標軸方向の要素数
(名称固定)

```
#S-GRID  
<$E-P1>/run/  
fvm_entire_mesh.dat
```

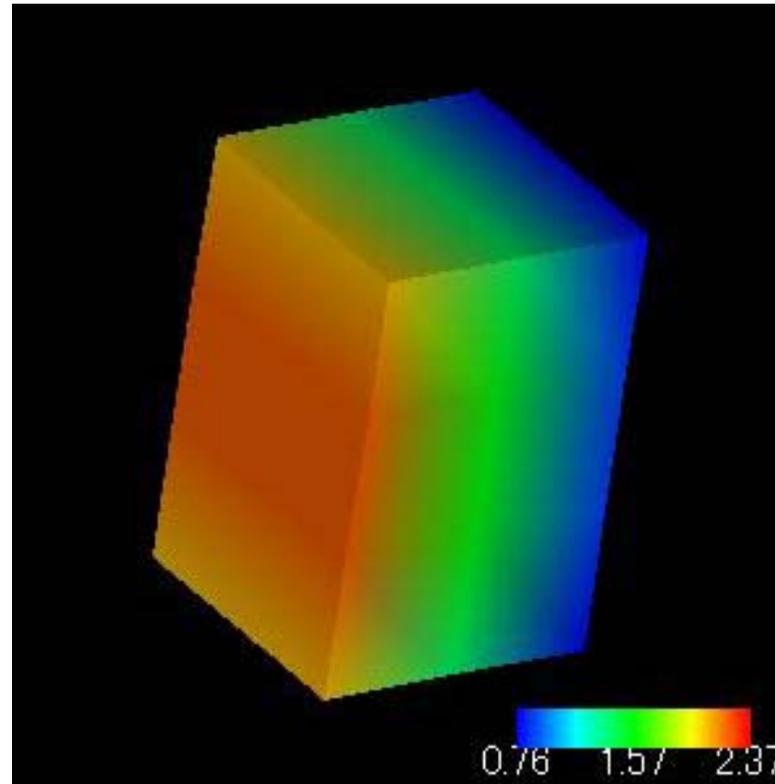
初期全体メッシュデータ
(名称固定)

eps_fvm

```
#S-GRID-R-UCD  
<$E-P1>/run/  
fvm_entire_mesh  
_results.inp
```

結果ファイルAVS表示用
(名称固定)

計算結果



- 「`fvm_entire_mesh_results.inp`」をMicroAVSによって処理することができる。

- 有限体積法の概要
- 1CPU用プログラム `eps_fvm`
 - メッシュ生成, 形状データ
 - 計算実行例
 - **プログラムの内容**

プログラムの概要

```
$> cd <$E-P1>/serial
$> cat eps_fvm.f
```

メインプログラム

```

    program eps_fvm

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

    call eps_fvm_input_grid
    call poi_gen
    call eps_fvm_solve

    call output_ucd

    end program eps_fvm
```

「eps_fvm」処理: メイン (1/2)

eps_fvm_all

```
program eps_fvm
  use eps_fvm_all

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

  call eps_fvm_input_grid
  call poi_gen
  call eps_fvm_solver
  call output_ucd

  end program eps_fvm
```

変数の内容について記載したモジュールブロック(コモンブロックのようなもの)。

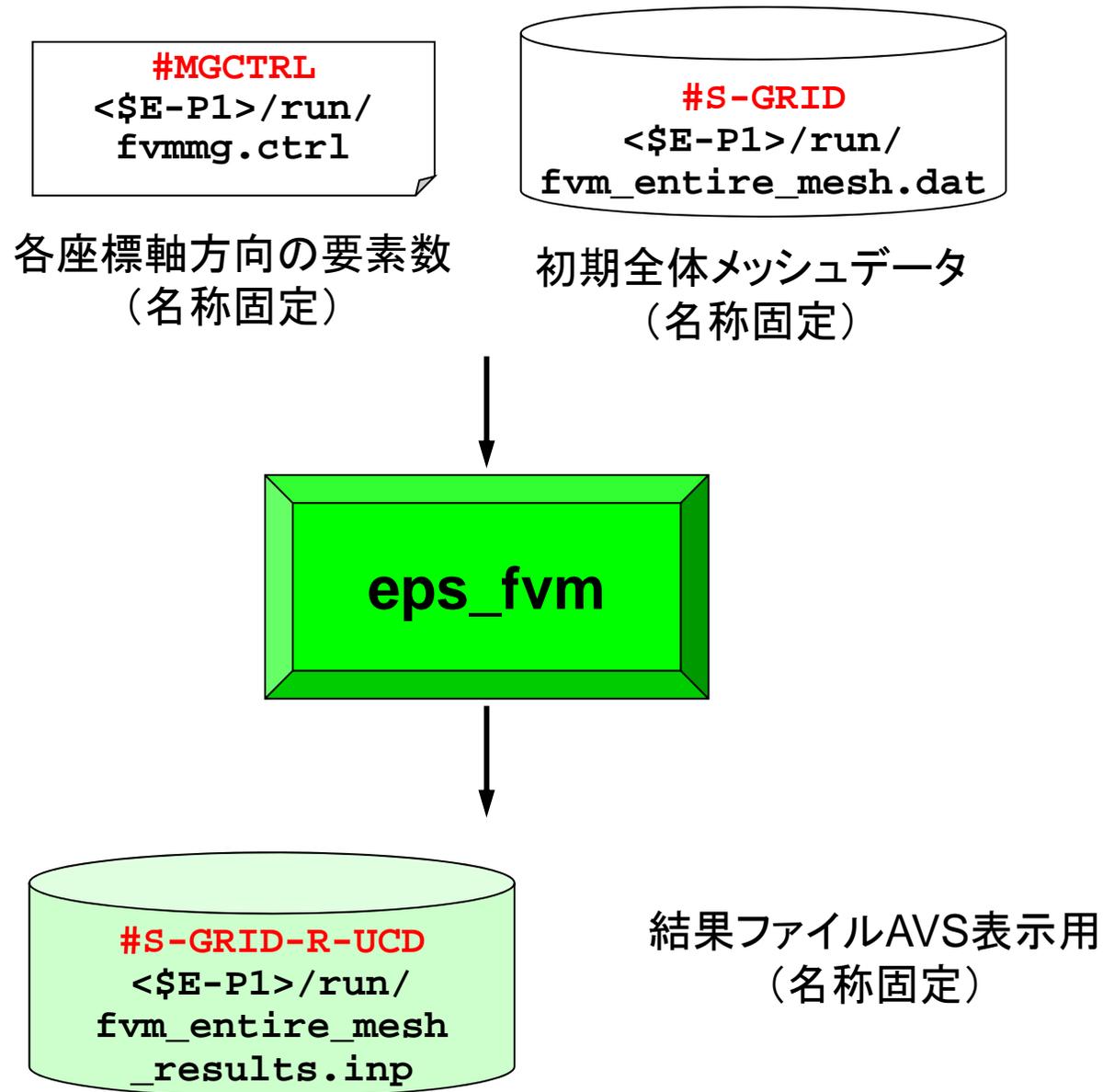
詳細はチュートリアル参照

http://nkl.cc.u-tokyo.ac.jp/tutorial/eps_fvm_tutorial/

http://nkl.cc.u-tokyo.ac.jp/tutorial/eps_fvm_tutorial.tar

```
module eps_fvm_all
  use eps_fvm_util
  use eps_fvm_pcg
  use appl_cntl
end module eps_fvm_all
```

eps_fvm の処理



「eps_fvm」処理：メイン(2/2)

その他

```
program eps_fvm
use eps_fvm_all

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

```
call eps_fvm_input_grid
call poi_gen
call eps_fvm_solver
call output_ucd
```

メッシュ読み込み(#S-GRID)

マトリクス生成

線形ソルバー

AVS用結果ファイル書き出し(S-GRID-R-UCD)

```
end program eps_fvm
```

```
program eps_fvm
use eps_fvm_all

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

```
call eps_fvm_input_grid
```

```
call poi_gen
```

```
call eps_fvm_solver
```

```
call output_ucd
```

```
end program eps_fvm
```

メッシュ読み込み(#S-GRID)

マトリクス生成

線形ソルバー

AVS用結果ファイル書き出し(S-GRID-R-UCD)

メッシュ関連変数 (eps_fvm_util)

変数名	型	配列サイズ	内容
NODE_tot	I	-	要素数
NODE_VOL(:)	R	NODE_tot	要素体積
NODE_COND(:)	R	NODE_tot	要素熱伝導率
NODE_XYZ(:)	R	3*NODE_tot	要素重心座標(3次元)
CONN_tot	I	-	コネクティビティ総数
CONN_node(:)	I	2*CONN_tot	コネクティビティ構成要素
CONN_COEF(:)	R	CONN_tot	コネクティビティ係数
FIX_NODE_tot	I	-	ディリクレ境界条件適用要素数
FIX_NODE_ID(:)	I	FIX_NODE_tot	ディリクレ境界条件適用要素番号
FIX_NODE_COEF(:)	R	FIX_NODE_tot	ディリクレ境界条件係数
FIX_NODE_VAL(:)	R	FIX_NODE_tot	ディリクレ境界条件値
SURF_NODE_tot	I	-	ノイマン境界条件適用要素数
SURF_NODE_ID(:)	I	SURF_NODE_tot	ノイマン境界条件適用要素番号
SURF_NODE_FLUX(:)	R	SURF_NODE_tot	ノイマン境界条件フラックス
BODY_NODE_tot	I	-	体積発熱境界条件適用要素数
BODY_NODE_ID(:)	I	BODY_NODE_tot	体積発熱境界条件適用要素番号
BODY_NODE_FLUX(:)	R	BODY_NODE_tot	体積発熱境界条件フラックス

「eps_fvm」処理: メッシュ読み込み (1/5)

```

subroutine eps_fvm_input_grid
use hpcmw_eps_fvm_all
implicit REAL*8 (A-H,O-Z)

character(len=NAME_LEN) :: member
character(len=80       ) :: LINE

!C
!C +-----+
!C | MESH INPUT |
!C +-----+
!C===
      open (22, file='fvmmg.ctrl', status='unknown')
      read (22,*) NX, NY, NZ
      close (22)

      IUNIT= 11
      open (IUNIT,file= 'fvm_entire_mesh.dat', status='unknown')

!C
!C-- NODE
      read (IUNIT, '(10i10)') NODE_tot

      if (NODE_tot.ne.NX*NY*NZ) then
        write (*,'(a)') "incosistent test.grid and test.mesh !!!"
        call hpcmw_eps_fvm_abort
      endif

      allocate (NODE_VOL(NODE_tot), NODE_COND(NODE_tot), &
&              NODE_XYZ(3*NODE_tot))

      do i= 1, NODE_tot
        read (IUNIT,'(i10,5e16.6)') ii, NODE_VOL(i), NODE_COND(i), &
&              (NODE_XYZ(3*i-3+k), k=1, 3)
      enddo

```

各座標軸方向の
要素数, AVS出力
用に必要

各要素情報の読み
込み

NODE_tot:
総要素数

「eps_fvm」処理: メッシュ読み込み(1/5)

```
!C
!C-- NODE
  read (IUNIT, '(10i10)') NODE_tot

  if (NODE_tot.ne.NX*NY*NZ) then
    write (*,'(a)') "incosistent test.grid and test.mesh !!!"
    call hpcmw_eps_fvm_abort
  endif

  allocate (NODE_VOL(NODE_tot), NODE_COND(NODE_tot), &
&          NODE_XYZ(3*NODE_tot))

  do i= 1, NODE_tot
    read (IUNIT,'(i10,5e16.6)') ii, NODE_VOL(i), NODE_COND(i), &
&          (NODE_XYZ(3*i-3+k), k=1, 3)
  enddo
```

要素番号	要素体積 NODE_VOL(i)	要素熱伝導率 NODE_COND(i)	要素中心x座標 NODE_XYZ(3*i-2)	要素中心y座標 NODE_XYZ(3*i-1)	要素中心z座標 NODE_XYZ(3*i)
12	要素数: NODE_tot				
1	1.000000E+00	1.000000E+00	5.000000E-01	5.000000E-01	5.000000E-01
2	1.000000E+00	1.000000E+00	1.500000E+00	5.000000E-01	5.000000E-01
3	1.000000E+00	1.000000E+00	5.000000E-01	1.500000E+00	5.000000E-01
4	1.000000E+00	1.000000E+00	1.500000E+00	1.500000E+00	5.000000E-01
5	1.000000E+00	1.000000E+00	5.000000E-01	5.000000E-01	1.500000E+00
6	1.000000E+00	1.000000E+00	1.500000E+00	5.000000E-01	1.500000E+00
7	1.000000E+00	1.000000E+00	5.000000E-01	1.500000E+00	1.500000E+00
8	1.000000E+00	1.000000E+00	1.500000E+00	1.500000E+00	1.500000E+00
9	1.000000E+00	1.000000E+00	5.000000E-01	5.000000E-01	2.500000E+00
10	1.000000E+00	1.000000E+00	1.500000E+00	5.000000E-01	2.500000E+00
11	1.000000E+00	1.000000E+00	5.000000E-01	1.500000E+00	2.500000E+00
12	1.000000E+00	1.000000E+00	1.500000E+00	1.500000E+00	2.500000E+00

「eps_fvm」処理: メッシュ読み込み (2/5)

コネクティビティ情報

```
!C
!C-- CONNECTION
      read (IUNIT,'(10i10)') CONN_tot
      allocate (CONN_NODE(2*CONN_tot), CONN_COEF(CONN_tot))
      do i= 1, CONN_tot
        read (IUNIT,'( 2i10, 3e16.6)') in1, in2, AREA, D1, D2
        CONN_NODE(2*i-1)= in1
        CONN_NODE(2*i  )= in2
        C1 = NODE_COND(in1)
        C2 = NODE_COND(in2)
        CONN_COEF(i)= AREA / ( D1/C1 + D2/C2 )
      enddo
```

CONN tot:
総コネクティビティ数

```
20 コネクティビティ総数: CONN_tot
  1      2      1.000000E+00      5.000000E-01      5.000000E-01
  1      3      1.000000E+00      5.000000E-01      5.000000E-01
  1      5      1.000000E+00      5.000000E-01      5.000000E-01
  2      4      1.000000E+00      5.000000E-01      5.000000E-01
...
  6     10      1.000000E+00      5.000000E-01      5.000000E-01
  7      8      1.000000E+00      5.000000E-01      5.000000E-01
  7     11      1.000000E+00      5.000000E-01      5.000000E-01
  8     12      1.000000E+00      5.000000E-01      5.000000E-01
  9     10      1.000000E+00      5.000000E-01      5.000000E-01
  9     11      1.000000E+00      5.000000E-01      5.000000E-01
 10     12      1.000000E+00      5.000000E-01      5.000000E-01
 11     12      1.000000E+00      5.000000E-01      5.000000E-01
```

E1 **E2** **s:要素境界面積**

**d1:E1重心～
境界面距離**

**d2:E2重心～
境界面距離**

E1= CONN_NODE(2*ic-1)

E2= CONN_NODE(2*ic)

有限体積法による空間離散化

熱流束に関するつりあい式

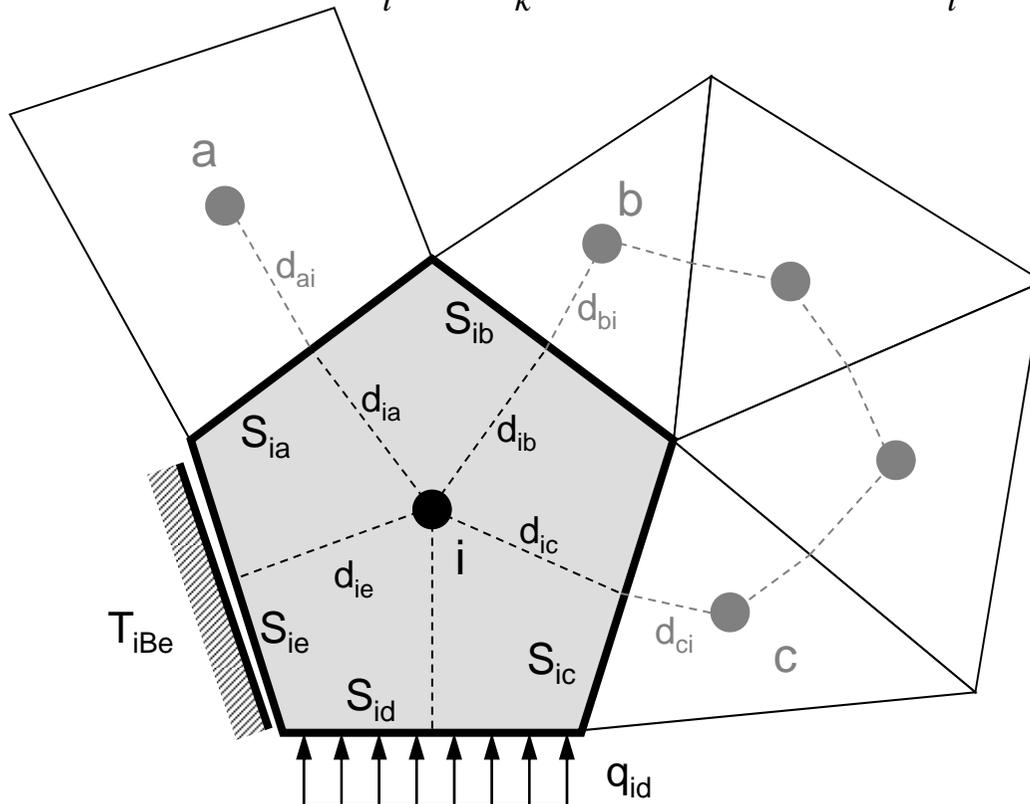
隣接要素との熱伝導

温度固定境界

$$\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) + \sum_e \frac{S_{ie}}{d_{ie}} (T_{iBe} - T_i) + \sum_d S_{id} \dot{q}_{id} + V_i \dot{Q}_i = 0$$

要素境界面
通過熱流束

体積発熱



- λ : 熱伝導率
- 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 : 表面フラックス
- Q : 体積発熱
- T_{iB} : 境界温度

「eps_fvm」処理：メッシュ読み込み(2/5)

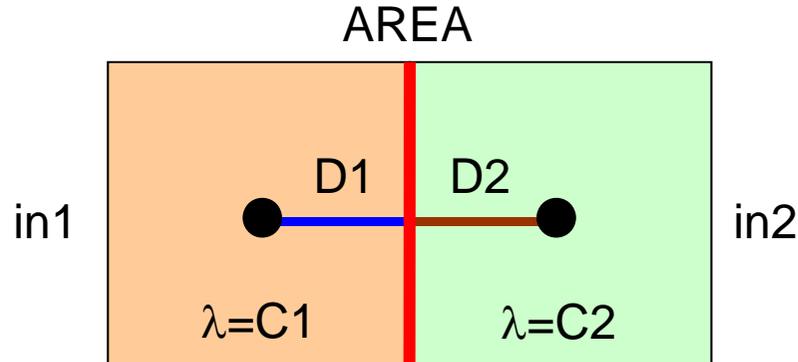
コネクティビティ情報

```
!C
!C-- CONNECTION
  read (IUNIT,'(10i10)') CONN_tot
  allocate (CONN_NODE(2*CONN_tot), CONN_COEF(CONN_tot))
  do i= 1, CONN_tot
    read (IUNIT,'( 2i10, 3e16.6)') in1, in2, AREA, D1, D2
    CONN_NODE(2*i-1)= in1
    CONN_NODE(2*i )= in2
    C1 = NODE_COND(in1)
    C2 = NODE_COND(in2)
    CONN_COEF(i)= AREA / ( D1/C1 + D2/C2 )
  enddo
```

CONN tot:

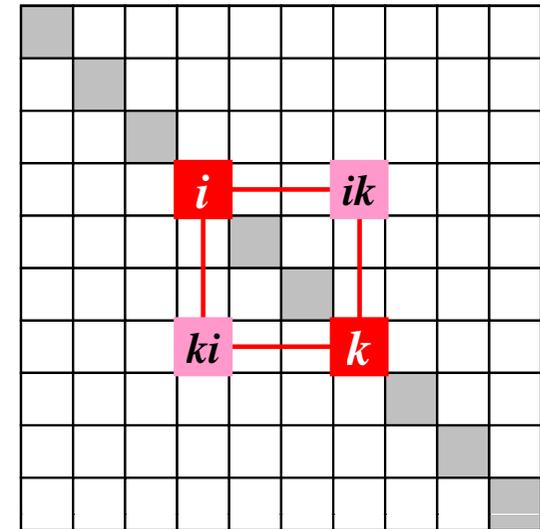
総コネクティビティ数
(これを2倍すると非対角成分総数)

非対角成分の計算を実施している



$$\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) + \sum_e \frac{S_{ie}}{d_{ie}} (T_{iBe} - T_i) + \sum_d S_{id} \dot{q}_{id} + V_i \dot{Q}_i = 0$$

CONN_COEF



係数行列の非対角成分

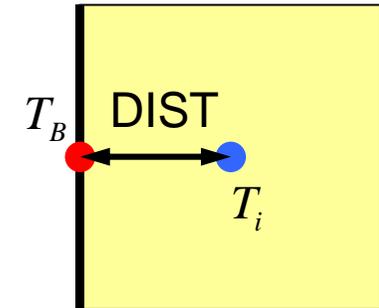
「eps_fvm」処理: メッシュ読み込み (3/5)

ディリクレ境界条件

```
!C
!C-- DIRICHLET
  read (IUNIT,'(10i10)') FIX_NODE_tot
  allocate (FIX_NODE_ID(FIX_NODE_tot), FIX_NODE_COEF(FIX_NODE_tot))
  allocate (FIX_NODE_VAL(FIX_NODE_tot))

  do i= 1, FIX_NODE_tot
    read (IUNIT, '(i10, 3e16.6)')
    &   FIX_NODE_ID(i), AREA, DIST, FIX_NODE_VAL(i): T_k
    icel= FIX_NODE_ID(i)
    COND= NODE_COND(icel)
    FIX_NODE_COEF(i)= AREA / (DIST/COND)
  enddo
```

AREA



6 ディリクレ境界条件を与える要素数: FIX_NODE_tot

2	1.000000E+00	5.000000E-01	0.000000E+00
4	1.000000E+00	5.000000E-01	0.000000E+00
6	1.000000E+00	5.000000E-01	0.000000E+00
8	1.000000E+00	5.000000E-01	0.000000E+00
10	1.000000E+00	5.000000E-01	0.000000E+00
12	1.000000E+00	5.000000E-01	0.000000E+00

要素番号	境界面積s	境界面と要素重心距離d	境界値T _B
<code>FIX_NODE_ID(ib)</code>			<code>FIX_NODE_VAL(ib)</code>

$$\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) + \sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}} (T_{iBe} - T_i) + \sum_d S_{id} \dot{q}_{id} + V_i \dot{Q}_i = 0$$

FIX_NODE_COEF

$$\sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}} (T_{iBe} - T_i)$$

FIX_NODE_VAL

「eps_fvm」処理: メッシュ読み込み(4/5)

ノイマン境界条件

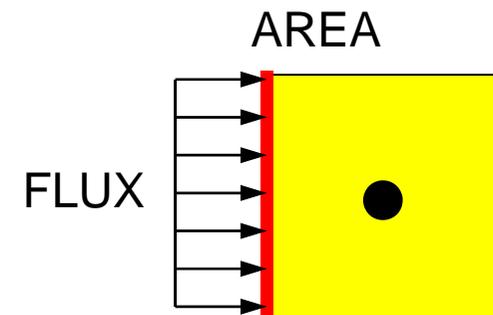
```
!C
!C-- NEUMANN
  read (IUNIT,'(10i10)') SURF_NODE_tot
  allocate
  & (SURF_NODE_ID (SURF_NODE_tot), SURF_NODE_FLUX(SURF_NODE_tot)) &

  do i= 1, SURF_NODE_tot
    read (IUNIT, '(i10, 3e16.6)') SURF_NODE_ID(i), AREA, FLUX
    SURF_NODE_FLUX(i)= AREA*FLUX
  enddo
```

6 ノイマン境界条件を与える要素数: SURF_NODE_tot

1	1.000000E+00	1.000000E+00
3	1.000000E+00	1.000000E+00
5	1.000000E+00	1.000000E+00
7	1.000000E+00	1.000000E+00
9	1.000000E+00	1.000000E+00
11	1.000000E+00	1.000000E+00

要素番号 境界表面積s 表面フラックスqs
SURF_NODE_ID(ib)



$$\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) + \sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}} (T_{iBe} - T_i) + \sum_d \boxed{S_{id} \dot{q}_{id}} + V_i \dot{Q}_i = 0$$

SURF_NODE_FLUX

「eps_fvm」処理: メッシュ読み込み (5/5)

体積発熱

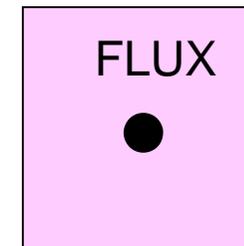
```
!C
!C-- BODY FLUX
  read (IUNIT,'(10i10)') BODY_NODE_tot

  allocate (BODY_NODE_FLUX(NODE_tot))
  do i= 1, BODY_NODE_tot
    read (IUNIT, '(i10, 3e16.6)') icel, FLUX
    BODY_NODE_FLUX(icel)= FLUX * NODE_VOL(icel)
  enddo
```

要素番号	体積フラックスQV
4	体積発熱を与える要素数
5	1.000000E+00
6	1.000000E+00
7	1.000000E+00
8	1.000000E+00

$$\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) + \sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}} (T_{iBe} - T_i) + \sum_d S_{id} \dot{q}_{id} + \boxed{V_i \dot{Q}_i} = 0$$

BODY_NODE_FLUX



```
program eps_fvm
use eps_fvm_all

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

call eps_fvm_input_grid
call poi_gen
call eps_fvm_solver
call output_ucd

end program eps_fvm
```

メッシュ読み込み (#S-GRID)
マトリクス生成
線形ソルバー
AVS用結果ファイル書き出し (S-GRID-R-UCD)

マトリクス関連変数表 (eps_fvm_pcg)

変数名	型	サイズ	内容
NPLU	I	-	連立一次方程式係数マトリクス非対角成分総数
D(:)	R	NODE_tot	連立一次方程式係数マトリクス対角成分
PHI(:)	R	NODE_tot	連立一次方程式未知数ベクトル
BFORCE(:)	R	NODE_tot	連立一次方程式右辺ベクトル
index(:)	I	0:NODE_tot	係数マトリクス非対角成分要素番号用一次元圧縮配列(非対角成分数)
item(:)	I	NPLU	係数マトリクス非対角成分要素番号用一次元圧縮配列(非対角成分要素(列)番号)
AMAT(:)	R	NPLU	係数マトリクス非対角成分要素番号用一次元圧縮配列(非対角成分)

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

行列ベクトル積への適用

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

D (i) 対角成分(実数, $i=1, N$)
index(i) 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) 非対角成分の要素(列)番号
(整数, $k=1, \text{index}(N)$)
AMAT(k) 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
```

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	$index(0) = 0$
②	3.6 ②	4.3 ①	2.5 ④	3.7 ⑥	9.1 ⑧	4	$index(1) = 2$
③	5.7 ③	1.5 ⑤	3.1 ⑦			2	$index(2) = 6$
④	9.8 ④	4.1 ②	2.5 ⑤	2.7 ⑥		3	$index(3) = 8$
⑤	11.5 ⑤	3.1 ①	9.5 ②	10.4 ③	4.3 ⑦	4	$index(4) = 11$
⑥	12.4 ⑥	6.5 ③	9.5 ⑦			2	$index(5) = 15$
⑦	23.1 ⑦	6.4 ②	2.5 ③	1.4 ⑥	13.1 ⑧	4	$index(6) = 17$
⑧	51.3 ⑧	9.5 ②	1.3 ③	9.6 ④	3.1 ⑥	4	$index(7) = 21$
							$index(8) = 25$

NPLU= 25
(=index(N))

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

Compressed Row Storage (CRS)

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

NPLU = 25
(=index(N))

$index(i-1)+1 \sim index(i)$ 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:

$\text{item}(7) = 5, \text{AMAT}(7) = 1.5$

$\text{item}(19) = 3, \text{AMAT}(19) = 2.5$

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

「eps_fvm」処理：マトリクス生成(1/5)

```
!C
!C***
!C*** POI_GEN
!C***
!C
!C   generate COEF. MATRIX for POISSON equations
!C
!C   subroutine POI_GEN
!C
!C       use eps_fvm_all
!C       implicit REAL*8 (A-H,O-Z)
!C       integer, pointer :: IWKX(:, :)
!C
!C   +-----+
!C   | INIT. |
!C   +-----+
!C===
!C
!C-- MATRIX
!C   nn = NODE_tot
!C
!C   allocate (BFORCE(nn), D(nn), PHI(nn))
!C   allocate (index(0:nn))
!C
!C   BFORCE= 0.d0
!C   PHI= 0.d0
!C   D= 0.d0
!C
!C   index= 0
```

「eps_fvm」処理: マトリクス生成 (2/5)

```
!C
!C-- ETC.
  allocate (IWKX(NODE_tot,6))
  IWKX= 0

  do ic= 1, CONN_tot
    in1= CONN_NODE(2*ic-1)
    in2= CONN_NODE(2*ic )

    ik1= index(in1) + 1
    IWKX (in1,ik1)= ic
    index(in1      )= ik1

    ik2= index(in2) + 1
    IWKX (in2,ik2)= ic
    index(in2      )= ik2
  enddo

  do i= 1, nn
    index(i)= index(i-1) + index(i)
  enddo

  NPLU= index(nn)

  allocate (item(NPLU), AMAT(NPLU))
```

(非零)非対角成分算出
(非零)非対角成分抽出

index(in):
各要素の非対角成分数
(通し番号)

IWKX(in,1-6):
各要素の非対角成分の
コネクティビティID
(CONN_NODE)

非対角要素数を増やした場合は
IWKXのサイズを増やせばよい
(現在は隣接要素数が6までに限
定されている)

「eps_fvm」処理: マトリクス生成 (2/5)

```
!C
!C-- ETC.
  allocate (IWKX(NODE_tot,6))
  IWKX= 0

  do ic= 1, CONN_tot
    in1= CONN_NODE(2*ic-1)
    in2= CONN_NODE(2*ic )

    ik1= index(in1) + 1
    IWKX (in1,ik1)= ic
    index(in1      )= ik1

    ik2= index(in2) + 1
    IWKX (in2,ik2)= ic
    index(in2      )= ik2
  enddo

  do i= 1, nn
    index(i)= index(i-1) + index(i)
  enddo

  NPLU= index(nn)

  allocate (item(NPLU), AMAT(NPLU))
```

index(in):
各要素の非対角成分数
(通し番号)

NPLU:
(非零)非対角成分の総数

「eps_fvm」処理: マトリクス生成 (3/5)

```

do i= 1, NODE_tot
  do j= 1, index(i)-index(i-1)
    k= index(i-1) + j

    ic = IWKX(i,j)
    in1= CONN_NODE(2*ic-1)
    in2= CONN_NODE(2*ic )

    if (in1.eq.i) then
      item(k)= in2
    else
      item(k)= in1
    endif
  enddo
enddo

!C===

!C
!C +-----+
!C | INTERIOR NODEs + BODY FLUX |
!C +-----+
!C===

do icel= 1, NODE_tot
  BFORCE(icel)= BFORCE(icel) + BODY_NODE_FLUX(icel)
enddo

do i= 1, NODE_tot
  do j= index(i-1)+1, index(i)
    icon= IWKX(i,j-index(i-1))
    AMAT(j)= -CONN_COEF(icon)
    D (i)= D(i) + CONN_COEF(icon)
  enddo
enddo
deallocate (IWKX)

!C===

```

index(in):
各要素の非対角成分数
(通し番号)

IWKX(in,1-6):
各要素の非対角成分の
コネクティビティID
(CONN_NODE)

item(k):
非対角成分(列番号)

有限体積法による空間離散化

熱流束に関するつりあい式

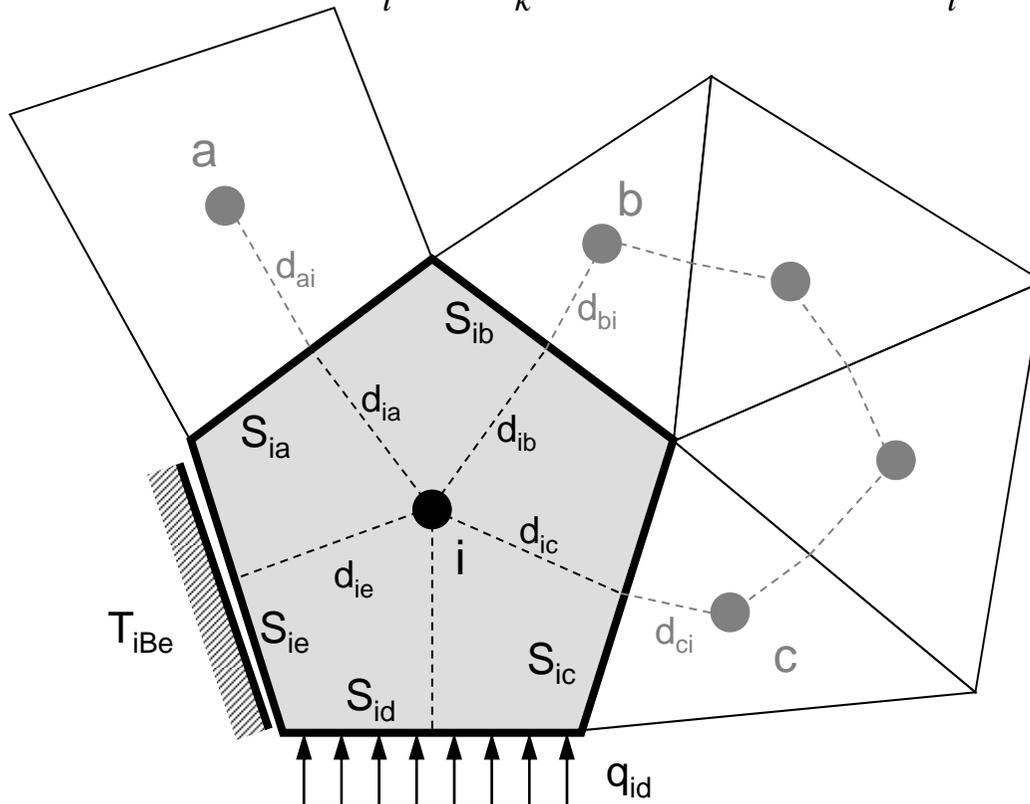
隣接要素との熱伝導

温度固定境界

$$\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) + \sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}} (T_{iBe} - T_i) + \sum_d S_{id} \dot{q}_{id} + V_i \dot{Q}_i = 0$$

要素境界面
通過熱流束

体積発熱



- λ : 熱伝導率
- 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 : 表面フラックス
- Q : 体積発熱
- T_{iB} : 境界温度

全体マトリクスの生成

要素*i*に関する釣り合い

$$\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) + \sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}} (T_{iBe} - T_i) + \sum_d S_{id} \dot{q}_{id} + V_i \dot{Q}_i = 0$$

隣接要素との熱伝導

温度固定境界

要素境界面
通過熱流束

体積
発熱

全体マトリクスの生成

要素*i*に関する釣り合い

$$-\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) - \sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}} (T_{iBe} - T_i) - \sum_d S_{id} \dot{q}_{id} - V_i \dot{Q}_i = 0$$

隣接要素との熱伝導

温度固定境界

要素境界面
通過熱流束

体積
発熱

全体マトリクスの生成

要素*i*に関する釣り合い

$$-\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) - \sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}} (T_{iBe} - T_i) - \sum_d S_{id} \dot{q}_{id} - V_i \dot{Q}_i = 0$$

$$-\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} T_k + \sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} T_i - \sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}} T_{iBe} + \sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}} T_i = \sum_d S_{id} \dot{q}_{id} + V_i \dot{Q}_i$$

定数項: 右辺へ移項

全体マトリクスの生成

要素*i*に関する釣り合い

$$-\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} (T_k - T_i) - \sum_e \frac{S_{ie}}{d_{ie}} (T_{iBe} - T_i) - \sum_d S_{id} \dot{q}_{id} - V_i \dot{Q}_i = 0$$

$$-\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} T_k + \sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} T_i - \sum_e \frac{S_{ie}}{d_{ie}} T_{iBe} + \sum_e \frac{S_{ie}}{d_{ie}} T_i = \sum_d S_{id} \dot{q}_{id} + V_i \dot{Q}_i$$

$$\left[\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} + \sum_e \frac{S_{ie}}{d_{ie}} \right] T_i - \left[\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} T_k \right] = \sum_d S_{id} \dot{q}_{id} + V_i \dot{Q}_i + \sum_e \frac{S_{ie}}{d_{ie}} T_{iBe}$$

D(対角成分)

AMAT(非対角成分)

BFORCE(右辺)

「eps_fvm」処理: マトリクス生成 (3/5)

体積発熱

```

do i= 1, NODE_tot
  do j= 1, index(i)-index(i-1)
    k= index(i-1) + j

    ic = IWKX(i,j)
    in1= CONN_NODE(2*ic-1)
    in2= CONN_NODE(2*ic )

    if (in1.eq.i) then
      item(k)= in2
    else
      item(k)= in1
    endif
  enddo
enddo !C===

```

```

!C
!C +-----+
!C | INTERIOR NODEs + BODY FLUX |
!C +-----+
!C===

```

```

do icel= 1, NODE_tot
  BFORCE(icel)= BFORCE(icel) + BODY_NODE_FLUX(icel)
enddo

```

```

do i= 1, NODE_tot
  do j= index(i-1)+1, index(i)
    icon= IWKX(i,j-index(i-1))
    AMAT(j)= -CONN_COEF(icon)
    D (i)= D(i) + CONN_COEF(icon)
  enddo
enddo
deallocate (IWKX)

```

```
!C===
```

$$\left[\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} + \sum_e \frac{S_{ie}}{\lambda_i} \right] T_i - \left[\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} T_k \right]$$

$$= \sum_d S_{id} \dot{q}_{id} + \underline{V_i \dot{Q}_i} + \sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}} T_{iBe}$$

BFORCE (右辺)

BODY_NODE_FLUX

「eps_fvm」処理: マトリクス生成 (3/5)

内部熱伝導

```

do i= 1, NODE_tot
  do j= 1, index(i)-index(i-1)
    k= index(i-1) + j

    ic = IWKX(i,j)
    in1= CONN_NODE(2*ic-1)
    in2= CONN_NODE(2*ic )

    if (in1.eq.i) then
      item(k)= in2
    else
      item(k)= in1
    endif
  enddo
enddo !C===

```

```

!C
!C +-----+
!C | INTERIOR NODEs + BODY FLUX |
!C +-----+
!C===

```

```

do icel= 1, NODE_tot
  BFORCE(icel)= BFORCE(icel) + BODY_NODE_FLUX(icel)
enddo

```

```

do i= 1, NODE_tot
  do j= index(i-1)+1, index(i)
    icon= IWKX(i,j-index(i-1))
    AMAT(j)= -CONN_COEF(icon)
    D (i)= D(i) + CONN_COEF(icon)
  enddo
enddo
deallocate (IWKX)

```

```
!C===
```

$$\begin{aligned}
 & \left[\sum_k \frac{S_{ik}}{\lambda_i} + \sum_e \frac{S_{ie}}{\lambda_i} \right] T_i - \left[\sum_k \frac{S_{ik}}{\lambda_i + \lambda_k} T_k \right] \\
 & \quad \text{CONN_COEF} \qquad \qquad \qquad \text{CONN_COEF} \\
 & = \sum_d S_{id} \dot{q}_{id} + V_i \dot{Q}_i + \sum_e \frac{S_{ie}}{\lambda_i} T_{iBe}
 \end{aligned}$$

「eps_fvm」処理：マトリクス生成(4/5)

ディリクレ境界条件(表面温度固定)

```

!C
!C +-----+
!C | DIRICHLET |
!C +-----+
!C===
do i= 1, FIX_NODE_tot
  icel= FIX_NODE_ID(i)
  D      (icel)= D      (icel) + FIX_NODE_COEF(i)
  BFORCE(icel)= BFORCE(icel) + FIX_NODE_COEF(i)*FIX_NODE_VAL(i)
enddo
!C===

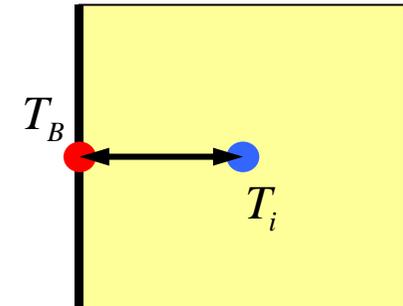
```

$$\left[\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} + \sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}} \right] T_i - \left[\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} T_k \right]$$

FIX_NODE_COEF

$$= \sum_d S_{id} \dot{q}_{id} + V_i \dot{Q}_i + \sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}} T_{iBe}$$

FIX_NODE_COEF * FIX_NODE_VAL



「eps_fvm」処理: マトリクス生成 (5/5)

ノイマン境界条件 (表面熱流束)

```

!C
!C +-----+
!C | SURFACE FLUX |
!C +-----+
!C===
      do i= 1, SURF_NODE_tot
         icel= SURF_NODE_ID(i)
         BFORCE(icel)= BFORCE(icel) + SURF_NODE_FLUX(i)
      enddo
!C===
return
end

```

$$\left[\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} + \sum_e \frac{S_{ie}}{\lambda_i} \right] T_i - \left[\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i} + \frac{d_{ki}}{\lambda_k}} T_k \right]$$

$$= \underbrace{\sum_d S_{id} \dot{q}_{id}}_{\text{SURF_NODE_FLUX}} + V_i \dot{Q}_i + \sum_e \frac{S_{ie}}{\lambda_i} T_{iBe} \quad \text{BFORCE(右辺)}$$

```
program eps_fvm
use eps_fvm_all

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

call eps_fvm_input_grid
call poi_gen
call eps_fvm_solver
call output_ucd

end program eps_fvm
```

メッシュ読み込み (#S-GRID)

マトリクス生成

線形ソルバー

AVS用結果ファイル書き出し (S-GRID-R-UCD)

「eps_fvm」処理: ソルバー

```
subroutine eps_fvm_solver

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

EPS = 1.d-8
ITR = NODE_tot

call eps_fvm_solver_CG                                &
& ( NODE_tot, NPLU, D, BFORCE, PHI, EPS,             &
&   ITR, IER, index, item, AMAT, COMMtime)
ISET= ISET + 1

open (11, file='fvmmg.ctrl', status='unknown')
  read (11,*) NX, NY, NZ
close (11)

iS= NX*NY*NZ/2 + NX*NY/2
do i= iS+1, iS+NX
  write (*,'(i8,3(1p16.6))') i, PHI(i)
enddo

end subroutine eps_fvm_solver
```

代表的な反復法：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 > 0 と同値
 - (ガラーキソ法) 熱伝導, 弾性, ねじり: 本コードの場合も SPD
- 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, $\mathbf{p}^{(i)}$: 探索ベクトル, 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)

共役勾配法のアルゴリズム

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

- 行列ベクトル積
- ベクトル内積
- ベクトル定数倍の加減

$x^{(i)}$: ベクトル

α_i : スカラー

前処理付き共役勾配法のアルゴリズム

```
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
```

前処理付共役勾配法

Preconditioned Conjugate Gradient Method (PCG)

```

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

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

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。

「eps_fvm」処理: ソルバー(1/7)

```
!C
!C***
!C*** CG
!C***
!C
  subroutine eps_fvm_solver_CG                                &          &
&      ( N, NPLU, D, B, X, EPS, ITR, IER,                    &
&      index, item, COEF)

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

  real(kind=kreal), dimension(N) :: D
  real(kind=kreal), dimension(N) :: B
  real(kind=kreal), dimension(N) :: X

  integer          , dimension(0:N) :: index
  integer          , dimension(NPLU):: item
  real (kind=kreal), dimension(NPLU):: COEF

  real(kind=kreal) :: EPS

  integer :: ITR, IER
  integer :: P, Q, R, Z, DD

  real(kind=kreal), dimension(:,,:), allocatable, save :: W
```

「eps_fvm」処理: ソルバー (2/7)

```

!C
!C +-----+
!C |  INIT.  |
!C +-----+
!C===
      if (.not.allocated(W)) then
          allocate (W(N,4))
      endif

      X= 0.d0
      W= 0.d0

      R = 1
      Z = 2
      Q = 2
      P = 3
      DD= 4

      do i= 1, N
          W(i,DD)= 1.0D0 / D(i)
      enddo

```

$W(i, 1) = W(i, R) \Rightarrow \{r\}$
 $W(i, 2) = W(i, Z) \Rightarrow \{z\}$
 $W(i, 2) = W(i, Q) \Rightarrow \{q\}$
 $W(i, 3) = W(i, P) \Rightarrow \{p\}$

$W(i, 4) = W(i, DD) \Rightarrow 1/DIAG$

```

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

```

「eps_fvm」処理: ソルバー (2/7)

```

!C
!C +-----+
!C |  INIT.  |
!C +-----+
!C===
      if (.not.allocated(W)) then
        allocate (W(NP,4))
      endif

      X= 0.d0
      W= 0.d0

      R = 1
      Z = 2
      Q = 2
      P = 3
      DD= 4

      do i= 1, N
        W(i,DD)= 1.0D0 / D(i)
      enddo

      IER = 0
      Tcomm= 0.d0
!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

```

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

```

「eps_fvm」処理: ソルバー (3/7)

```

!C
!C +-----+
!C | {r0} = {b} - [A]{xini} |
!C +-----+
!C===
      do i= 1, N
        W(i,R) = D(i)*X(i)
        do j= index(i-1)+1, index(i)
          W(i,R) = W(i,R) + COEF(j) * X(item(j))
        enddo
      enddo

      BNRM2= 0.0D0
      do i= 1, N
        BNRM2= BNRM2 + B(i) **2
        W(i,R)= B(i) - W(i,R)
      enddo
!C===

```

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

```

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) + αi p(i)
  r(i) = r(i-1) - αi q(i)
  check convergence |r|
end

```

「eps_fvm」処理: ソルバー (3/7)

```
!C
!C +-----+
!C | {r0} = {b} - [A]{xini} |
!C +-----+
!C===
do i= 1, N
  W(i,R) = D(i)*X(i)
  do j= index(i-1)+1, index(i)
    W(i,R) = W(i,R) + COEF(j) * X(item(j))
  enddo
enddo

BNRM2= 0.0D0
do i= 1, N
  BNRM2= BNRM2 + B(i) **2
  W(i,R)= B(i) - W(i,R)
enddo
!C===
```

$BNRM2 = |b|^2$
あとで収束判定に使用

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) + αi p(i)
  r(i) = r(i-1) - αi q(i)
  check convergence |r|
end
```

「eps_fvm」処理: ソルバー (4/7)

```

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

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

```

```

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

```

「eps_fvm」処理: ソルバー (5/7)

```

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

!C
!C +-----+
!C | {q}= [A]{p} |
!C +-----+
!C===
      do i= 1, N
        W(i,Q) = D(i) * W(i,P)
        do j= index(i-1)+1, index(i)
          W(i,Q) = W(i,Q) + COEF(j) * W(item(j),P)
        enddo
      enddo
!C===

```

```

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

```

「eps_fvm」処理: ソルバー (6/7)

```

!C
!C +-----+
!C | ALPHA= RHO / {p}{q} |
!C +-----+
!C===
      C1= 0.d0
      do i= 1, N
        C1= C1 + W(i,P)*W(i,Q)
      enddo
      ALPHA= RHO / C1
!C===

!C +-----+
!C | {x}= {x} + ALPHA*{p} |
!C | {r}= {r} - ALPHA*{q} |
!C +-----+
!C===
      do i= 1, N
        X(i) = X(i) + ALPHA * W(i,P)
        W(i,R)= W(i,R) - ALPHA * W(i,Q)
      enddo

```

```

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

```

「eps_fvm」処理: ソルバー (7/7)

```

DNRM2 = 0.0
do i= 1, N
  DNRM2= DNRM2 + W(i,R)**2
enddo

RESID= dsqrt(DNRM2/BNRM2)

if (my_rank.eq.0) write (*, 1000) L, RESID
1000 format (i5, lpe16.6)

if ( RESID.le.EPS) goto 900
RHO1 = RHO

enddo
IER = 1

900 continue

ITR= L
EPS= RESID

return

end subroutine hpcmw_eps_fvm_solver_CG

```

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

```

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

```

いろいろなメッシュでやってみよう

- MicroAVSの使い方に慣れる。