並列アプリケーション開発法入門（I）

2007年6月20日

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）

スケジュール

- 6月20日（水）: 並列アプリケーション開発入門（I）
  - 有限体積法
  - 1CPU用計算プログラム **eps_fvm**
  - 課題S3解説
- 7月4日（水）: 並列アプリケーション開発入門（II）
  - 領域分割手法
  - 並列分散メッシュデータ
- 7月11日（水）: 並列アプリケーション開発入門（III）
  - 並列FVMコードの開発

- 7月18日（水）: 並列アプリケーション開発入門（IV）
  - 密行列を係数行列とするアプリケーションの並列化

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

- 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

必要ファイルインストール

```
> cd <$S07>

FORTRAN
> cp /home/nakajima/class/2007summer/F/p1a-f.tar .
C
> cp /home/nakajima/class/2007summer/C/p1a-c.tar .

> tar xvf p1a-f.tar (p1a-c.tar)
> ls
      P1

> cd P1                 このディレクトリを<$P1>と呼ぶ
> ls
      mesh/             メッシュ生成
      run/              実行ディレクトリ
      serial/           1-CPUを使用した計算コード
```

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）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
- Parallel，Serial

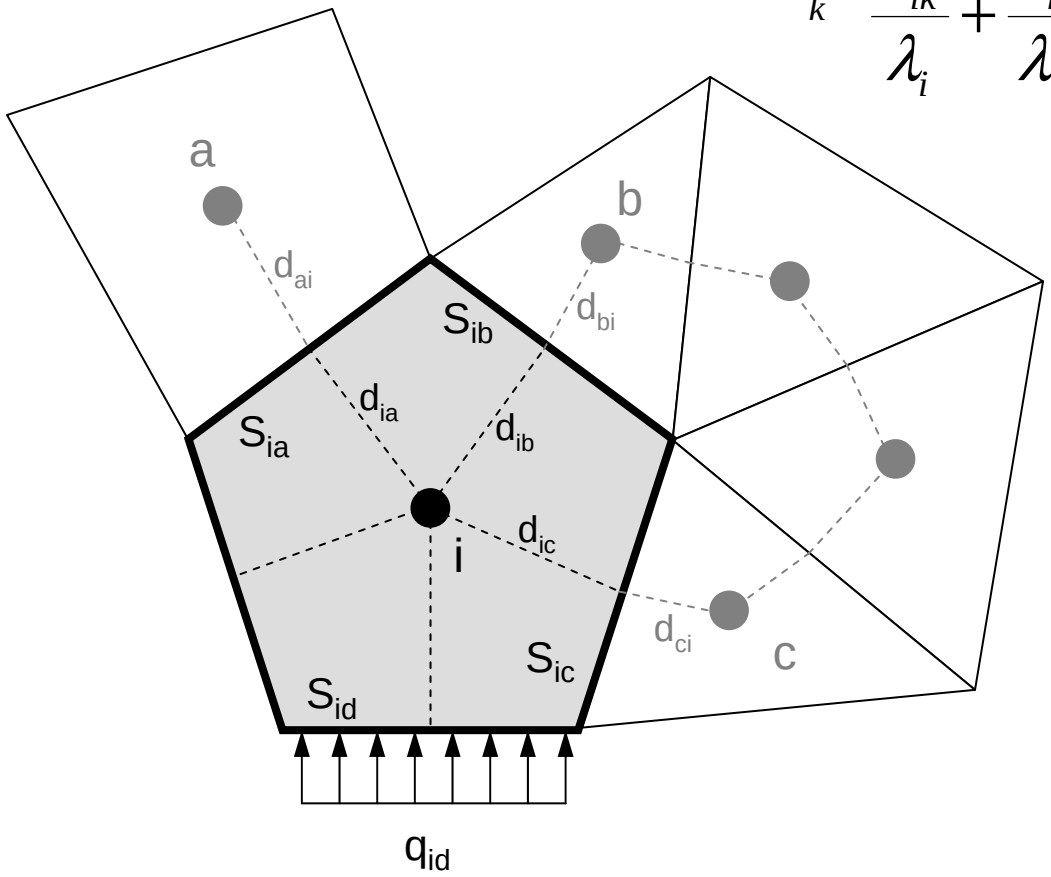
有限体積法による空間離散化

熱流束に関するつりあい式

隣接要素との熱伝導

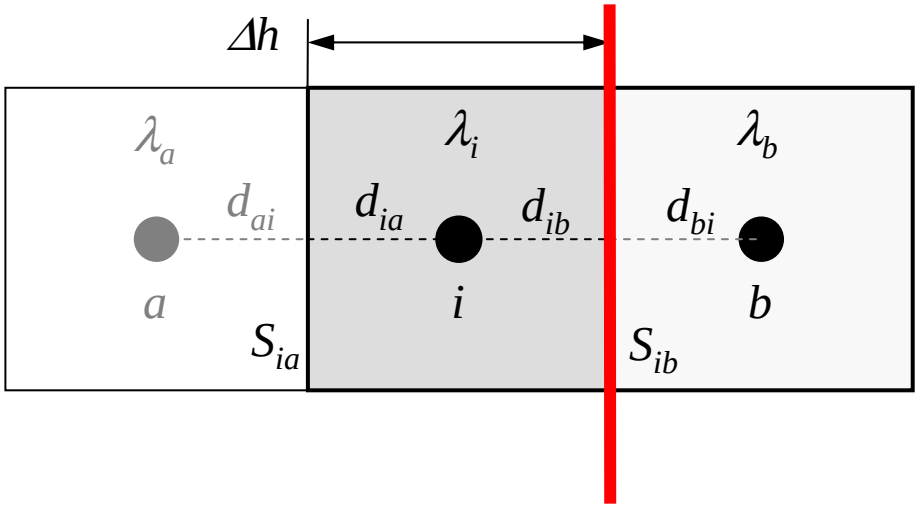
$$\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i}+\frac{d_{ki}}{\lambda_k}}\left(T_k - T_i\right) + \sum_d S_{id}\dot{q}_{id} + V_i\dot{Q}_i = 0$$

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: λi

この面を通過する熱量: Qs_{ib}

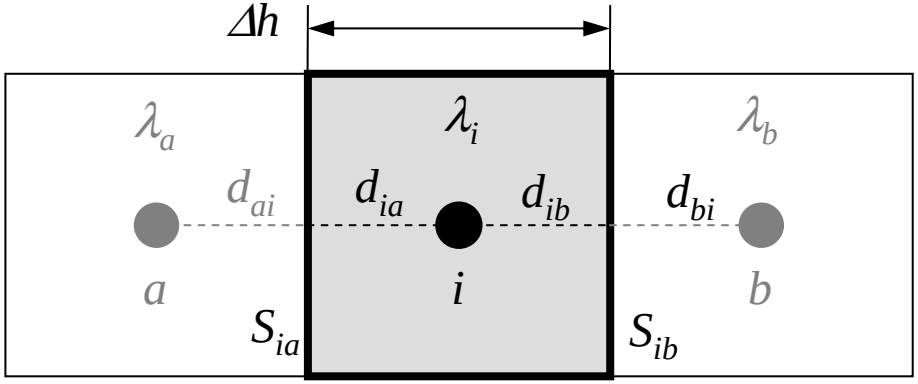
$$Qs_{ib} = -\lambda \frac{T_b - T_i}{d_{ib} + d_{bi}} \cdot S_{ib} = -\frac{T_b - T_i}{\frac{d_{ib}}{\lambda} + \frac{d_{bi}}{\lambda}} \cdot S_{ib}$$

熱伝導率が一様の場合
フーリエの法則

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

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: λi

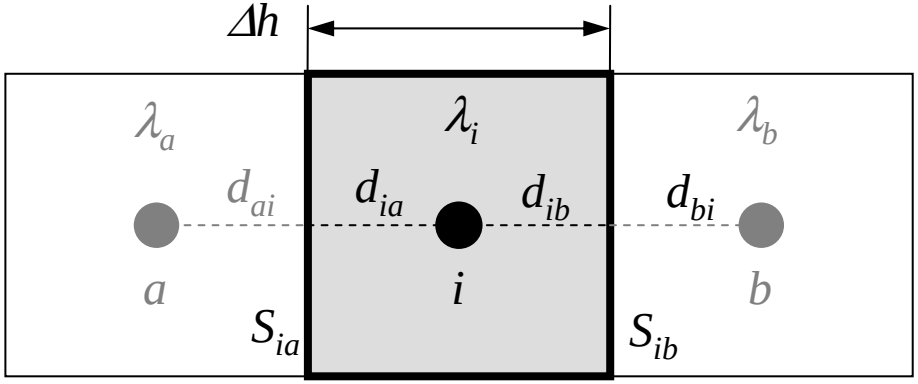
$$\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i}+\frac{d_{ki}}{\lambda_k}}\left(T_k - T_i\right) + \sum_d S_{id}\dot{q}_{id} + V_i\dot{Q}_i = 0$$

両辺をV_iで割る:

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

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: λi

熱伝導率一様として

$$\frac{1}{V_i}\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i}+\frac{d_{ki}}{\lambda_k}}(T_k - T_i) = \frac{1}{(\Delta h)^2}\sum_{k=a,b}\frac{\Delta h}{\frac{\Delta h/2}{\lambda}+\frac{\Delta h/2}{\lambda}}(T_k - T_i)$$

$$= \frac{1}{(\Delta h)^2}\sum_{k=a,b}\frac{\Delta h}{\frac{\Delta h}{2\lambda}+\frac{\Delta h}{2\lambda}}(T_k - T_i) = \frac{1}{(\Delta h)^2}\sum_{k=a,b}\frac{\Delta h}{\frac{\Delta h}{\lambda}}(T_k - T_i) = \frac{1}{(\Delta h)^2}\sum_{k=a,b}\lambda(T_k - T_i)$$

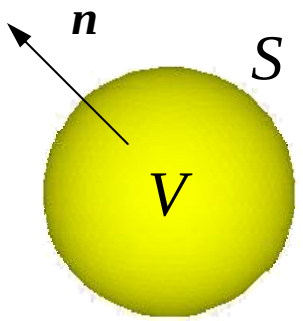
$$= \lambda\cdot\left[\frac{1}{(\Delta h)^2}(T_a - T_i)+\frac{1}{(\Delta h)^2}(T_b - T_i)\right] = \lambda\cdot\left[\frac{T_a - 2T_i + T_b}{(\Delta h)^2}\right]$$

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

$$\int_V \left(\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} + \frac{\partial W}{\partial z} \right) dV = \int_S \left(U n_x + V n_y + W n_z \right) dS$$



- 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$\boldsymbol{n}$
  - 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:

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

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

グリーンの定理(2/2)

- （続き）

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

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

- 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

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

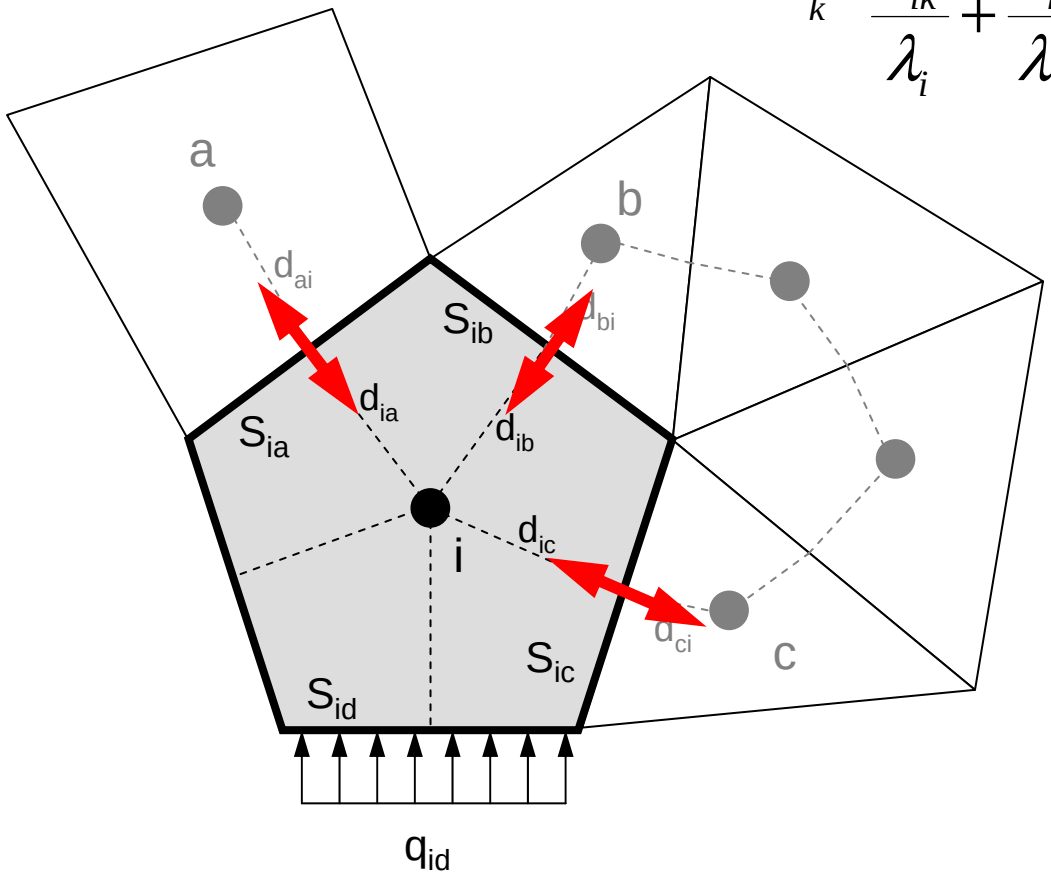
有限体積法による空間離散化

各要素の境界面を通貨する熱量に着目

隣接要素との熱伝導

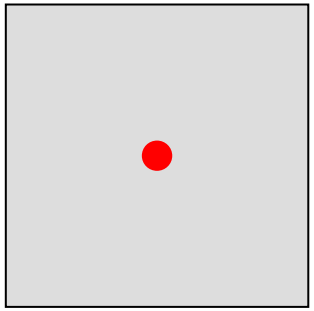
$$\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i}+\frac{d_{ki}}{\lambda_k}}\left(T_k - T_i\right) + \sum_d S_{id}\dot{q}_{id} + V_i\dot{Q}_i = 0$$

要素境界面　体積発熱
通過熱流束

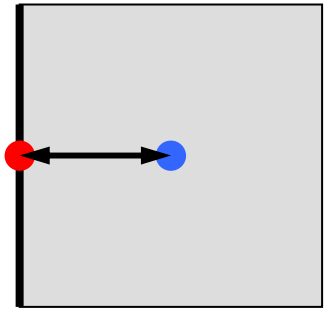


λ　：熱伝導率
V_i ：要素体積
S　：表面面積
d_{ij} ：要素中心から表面までの距離
q　：表面フラックス
Q　：体積発熱

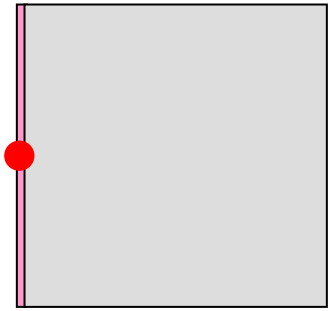
ディリクレ境界条件（境界温度固定）



単純に要素中心で指定
（精度悪化）

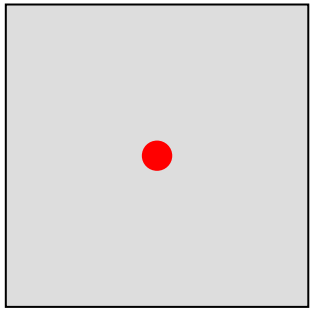


境界面からの等価な
フラックスを加算
（面積，重心との距離必要）

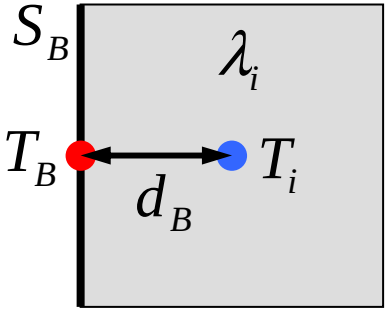


「薄い」要素を設定
（～10^{-20}）

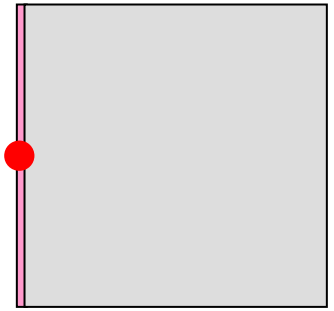
ディリクレ境界条件（境界温度固定）



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
（～10^{-20}）

現在はこの方式を使用

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

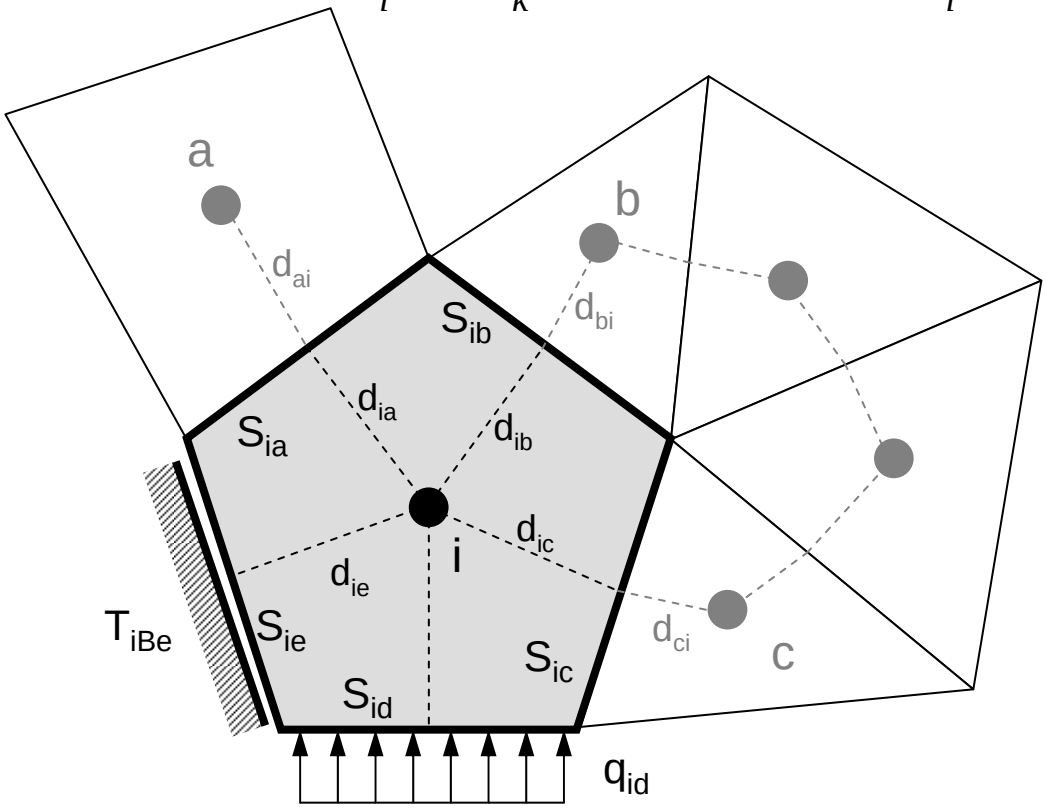
有限体積法による空間離散化

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

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

要素境界面通過熱流束　　体積発熱



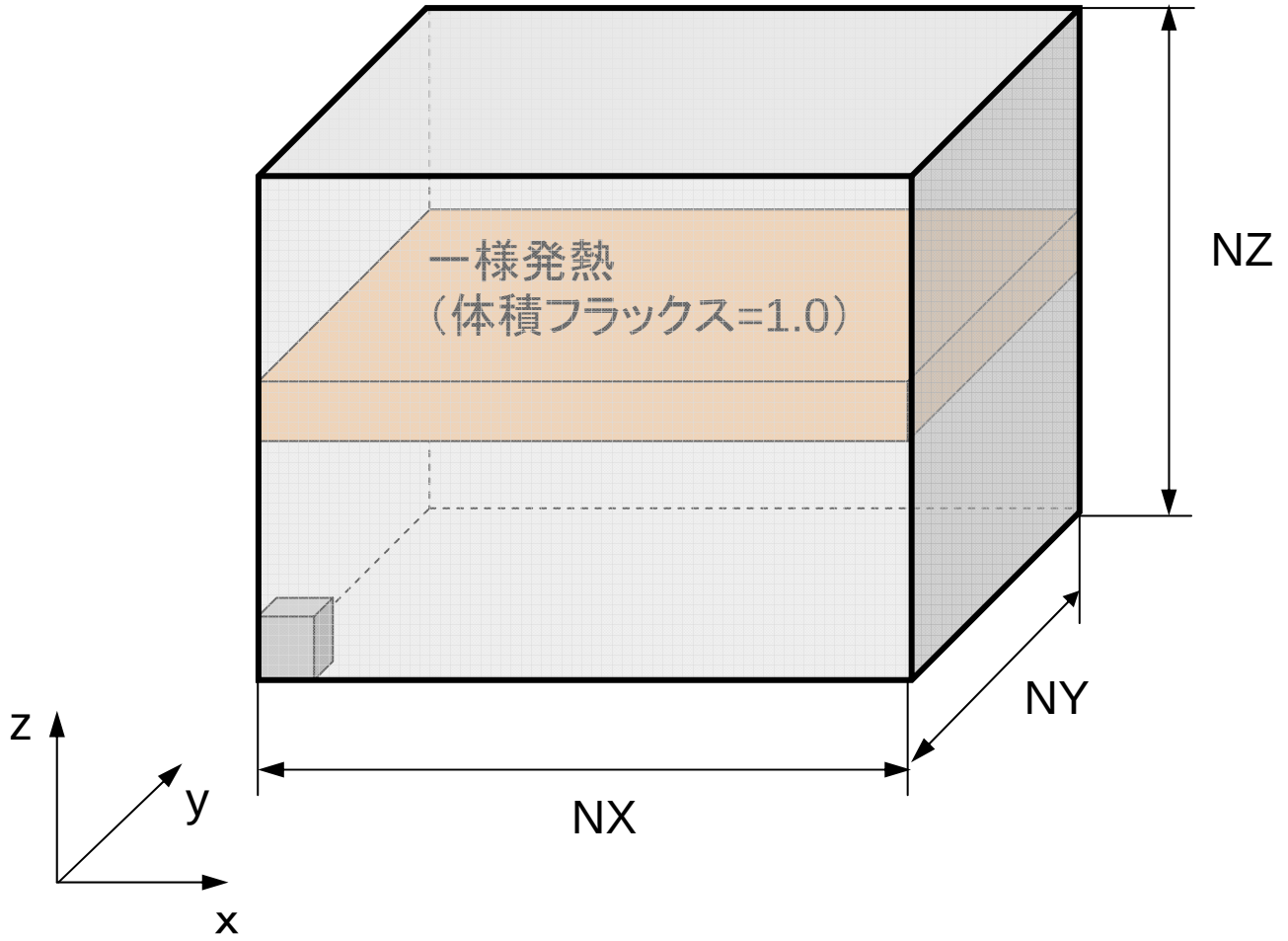
λ　：熱伝導率
V_i　：要素体積
S　：表面面積
d_{ij}　：要素中心から表面までの距離
q　：表面フラックス
Q　：体積発熱
T_{iB}　：境界温度

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。
  - この手法の面白さは，例えば，遠く離れた要素との相互関係（例えば熱輻射）を容易に定義できることにあるのだが・・・
  - 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



X=Xmax

ディリクレ境界条件

T=0

X=Xmin

ノイマン境界条件

-λ(dT/dX)=1.

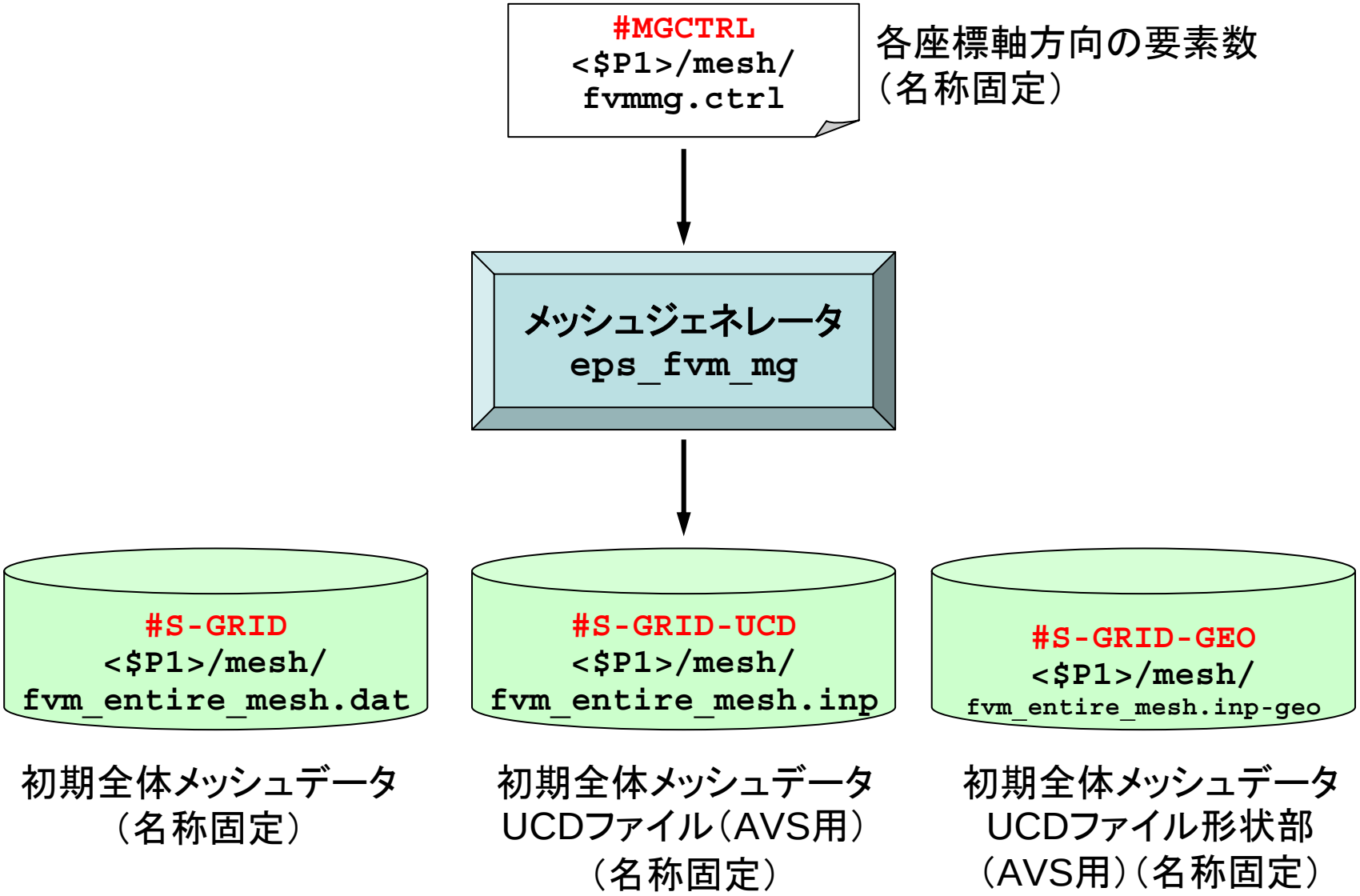
- 概要
- 1CPU用プログラム **eps_fvm**
  - –メッシュ生成，形状データ
  - –計算実行例
  - –プログラムの内容

メッシュ生成からやってみよう

```
$> cd <$P1>/mesh
$> 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2×2×3=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


#MGCTRL
<$P1>/mesh/
fvmmg.ctrl
各座標軸方向の要素数
（名称固定）
メッシュジェネレータ
eps_fvm_mg
#S-GRID
<$P1>/mesh/
fvm_entire_mesh.dat
#S-GRID-UCD
<$P1>/mesh/
fvm_entire_mesh.inp
#S-GRID-GEO
<$P1>/mesh/
fvm_entire_mesh.inp-geo
初期全体メッシュデータ
（名称固定）
初期全体メッシュデータ
UCDファイル（AVS用）
（名称固定）
初期全体メッシュデータ
UCDファイル形状部
（AVS用）（名称固定）

初期全体メッシュデータ（例）（1/6）

```
12
 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
20
 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
 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
 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
 5    1.000000E+00
 6    1.000000E+00
 7    1.000000E+00
 8    1.000000E+00
```

各要素情報

要素間コネクティビティ
面に隣接する要素

ディリクレ境界条件
温度固定

ノイマン境界条件
境界熱流束

体積発熱

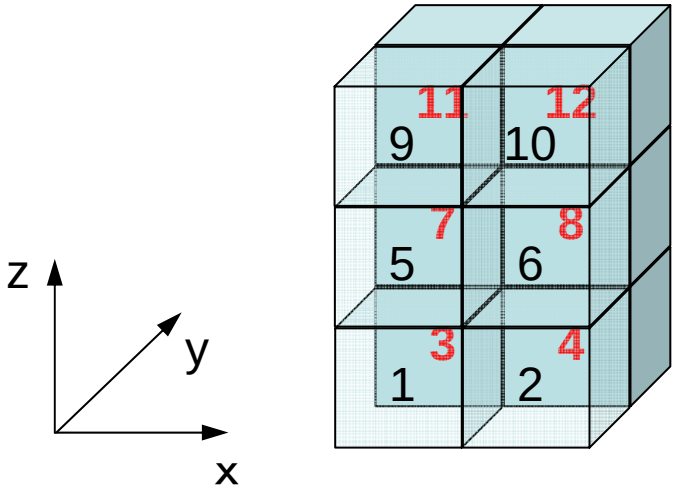
変数表

変数名	型	配列サイズ	内容
NODE_tot	I	-	内点数＋外点数
intNODE_tot	I	-	内点数
NODE_GLOBAL(:)	I	NODE_tot	グローバル要素番号
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

```
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
要素番号  要素体積        要素熱伝導率     要素中心x座標    要素中心y座標    要素中心z座標
          NODE_VOL(i)     NODE_COND(i)    NODE_XYZ(3*i-2) NODE_XYZ(3*i-1) NODE_XYZ(3*i)
```

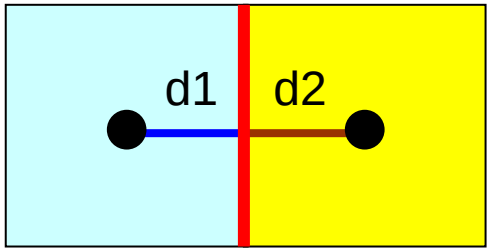


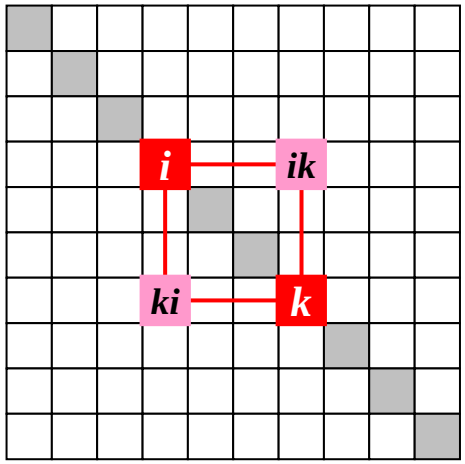

```
read (IUNIT, '(10i10)') NODE_tot
do i= 1, NODE_tot
  read (IUNIT,'(i10,5e16.6)')                    &
        ii, NODE_VOL(i), NODE_COND(i),           &
        (NODE_XYZ(3*i-3+k), k=1, 3)
enddo
```

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～      d2:E2重心～
                                    境界面距離       境界面距離

E1= CONN_NODE(2*ic-1)
E2= CONN_NODE(2*ic)
```





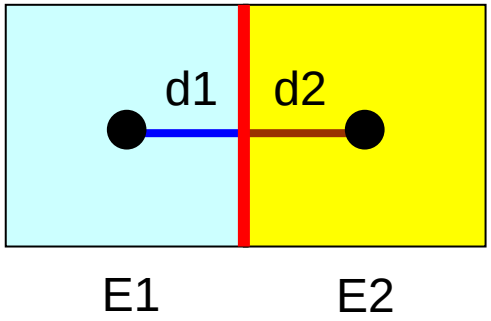
係数行列の非対角成分

```
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)

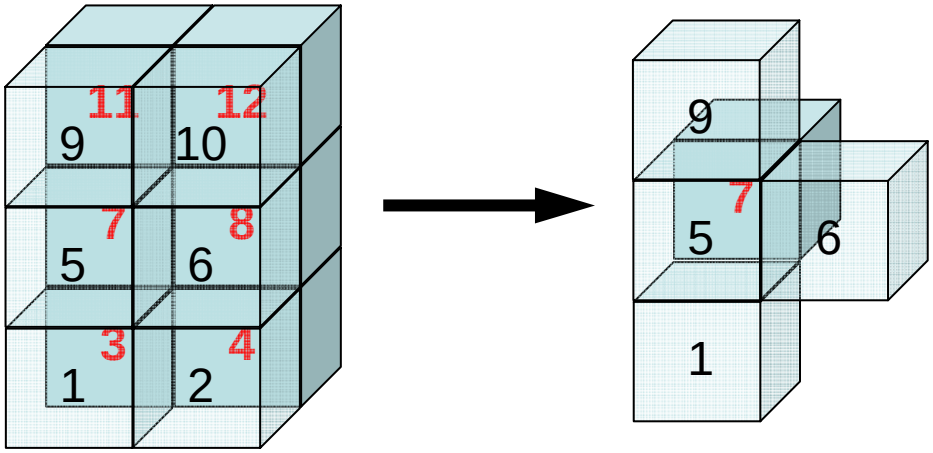
要素間コネクティビティ:5番要素に着目

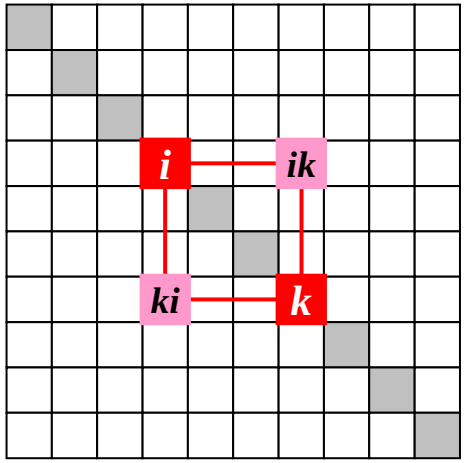
```
 1    5     1.0E+00      5.0E-01      5.0E-01
 5    6     1.0E+00      5.0E-01      5.0E-01
 5    7     1.0E+00      5.0E-01      5.0E-01
 5    9     1.0E+00      5.0E-01      5.0E-01
E1   E2   S:要素境界面積 d1:E1重心～    d2:E2重心～
                          境界面距離     境界面距離
```



各要素は一辺の長さ1の立方体:

- 要素境界面積は1×1=1
- 各要素重心から境界面までの距離は0.50





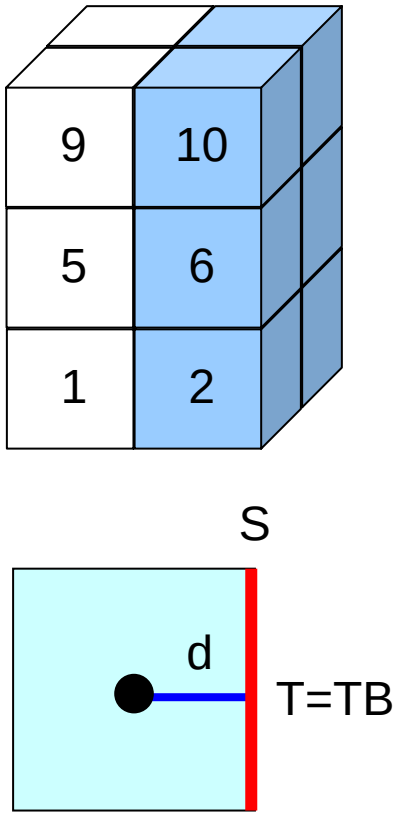
係数行列の非対角成分

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）

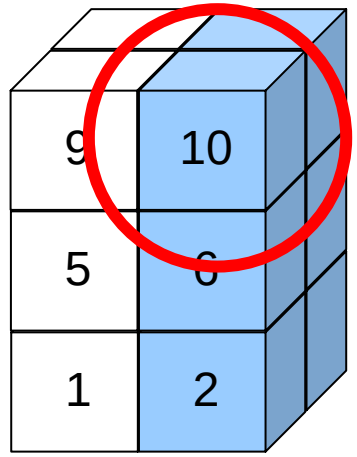
```
 6   ディリクレ境界条件を与える要素数: FIX_NODE_tot
 2       1.000000E+00    5.000000E-01    0.000000E+00
 4       1.000000E+00    5.000000E-01    0.000000E+00
 6       1.000000E+00    5.000000E-01    0.000000E+00
 8       1.000000E+00    5.000000E-01    0.000000E+00
10       1.000000E+00    5.000000E-01    0.000000E+00
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
```

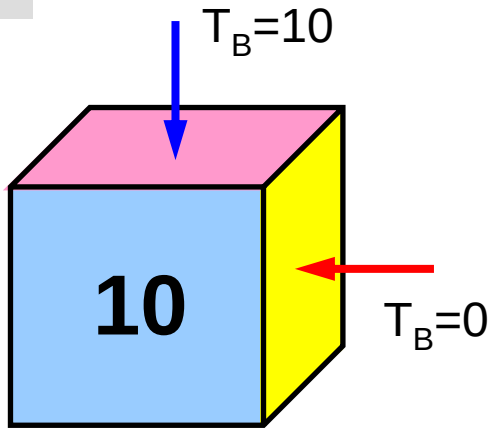


1つの「要素」に複数のディリクレ境界条件を与えることも実は可能

```
 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TB
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。



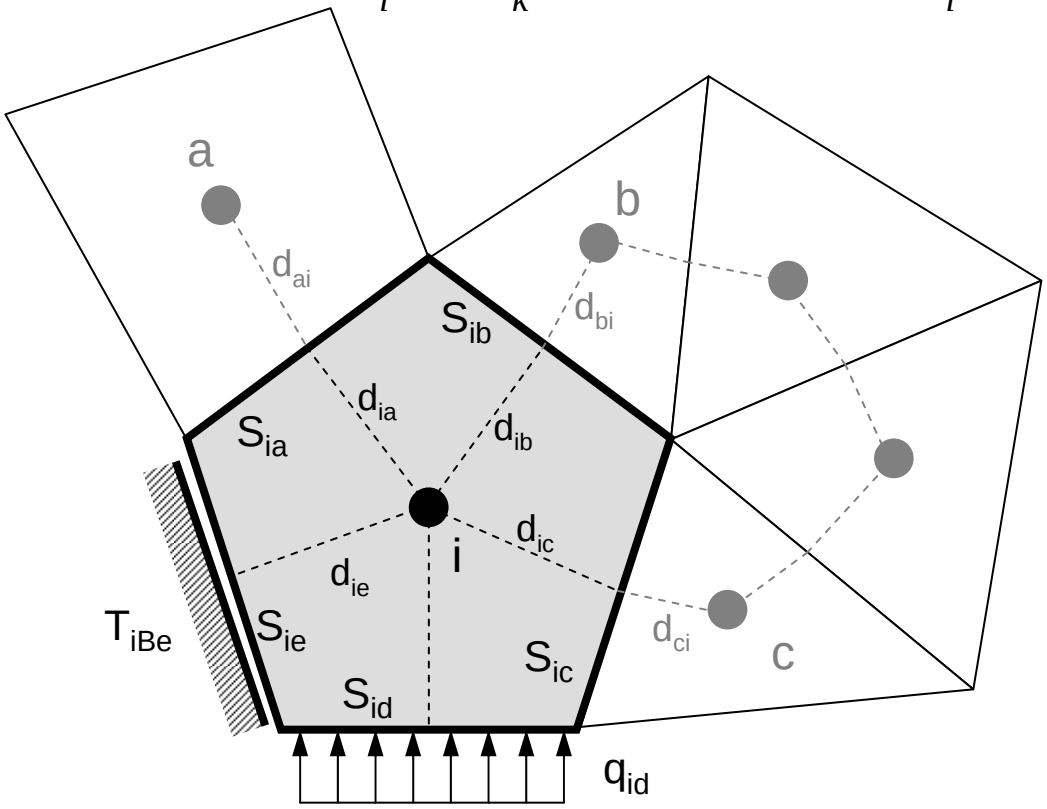
有限体積法による空間離散化

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

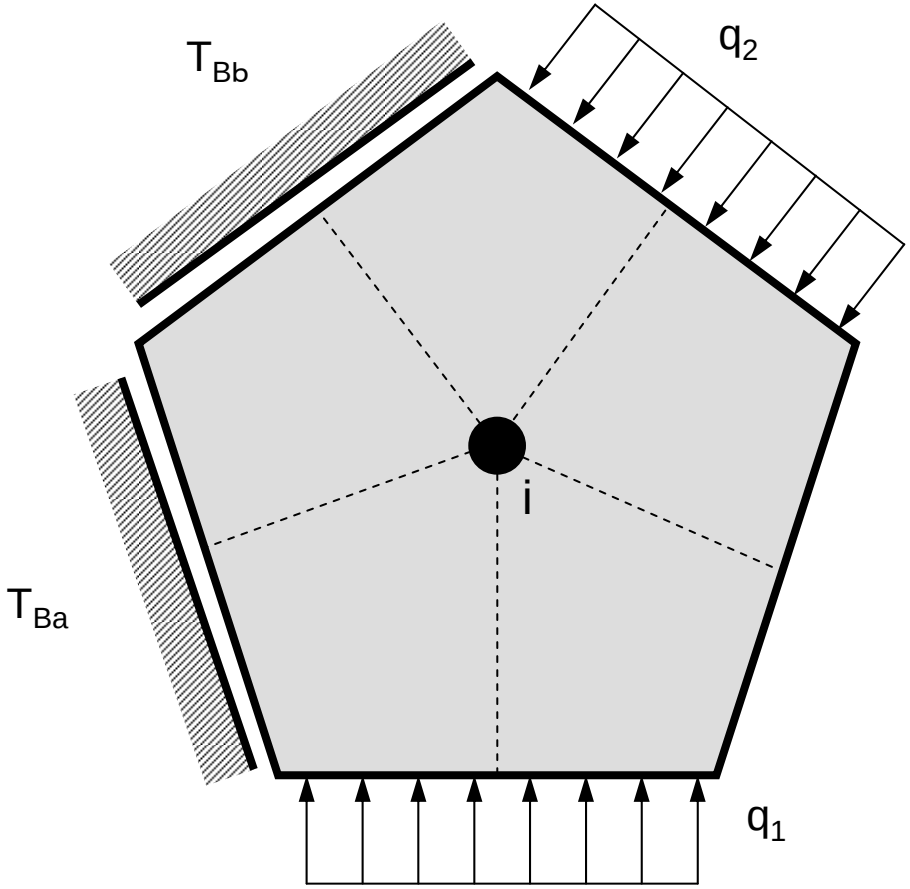
$$\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i}+\frac{d_{ki}}{\lambda_k}}\left(T_k - T_i\right) + \sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}}\left(T_{iBe} - T_i\right) + \sum_d S_{id}\dot{q}_{id} + V_i\dot{Q}_i = 0$$

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

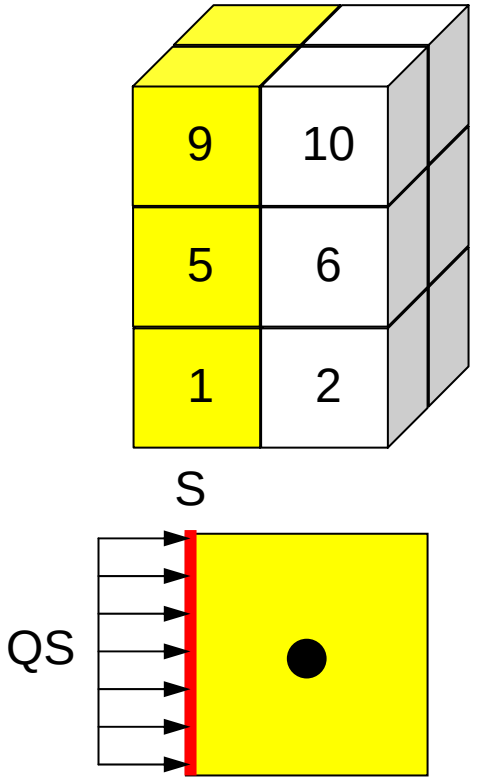


初期全体メッシュデータ（例）（5/6）

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
```



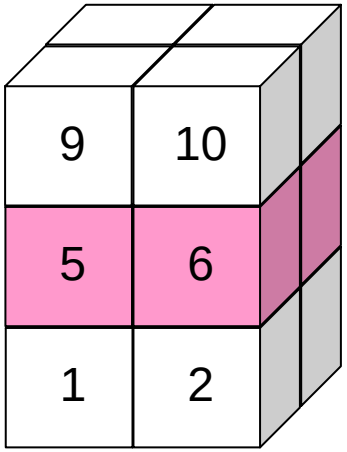
1つの要素に複数のノイマン境界条件を適用することは可能（複数の面に異なった熱流束を与える）。

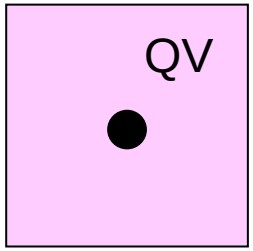
初期全体メッシュデータ（例）（6/6）

体積発熱（W/m^3）

```
 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
```

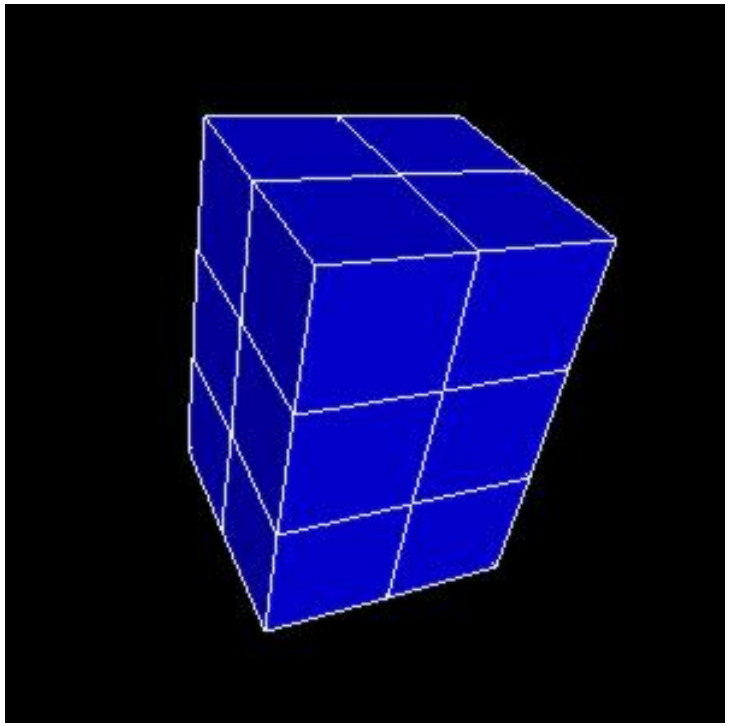




同じ要素に対して，複数回体積発熱を指定した場合には，一番最後に指定した値が適用される。

MicroAVSによる表示

#S-GRID-UCD:<$P1>/mesh/fvm_entire_mesh.inp



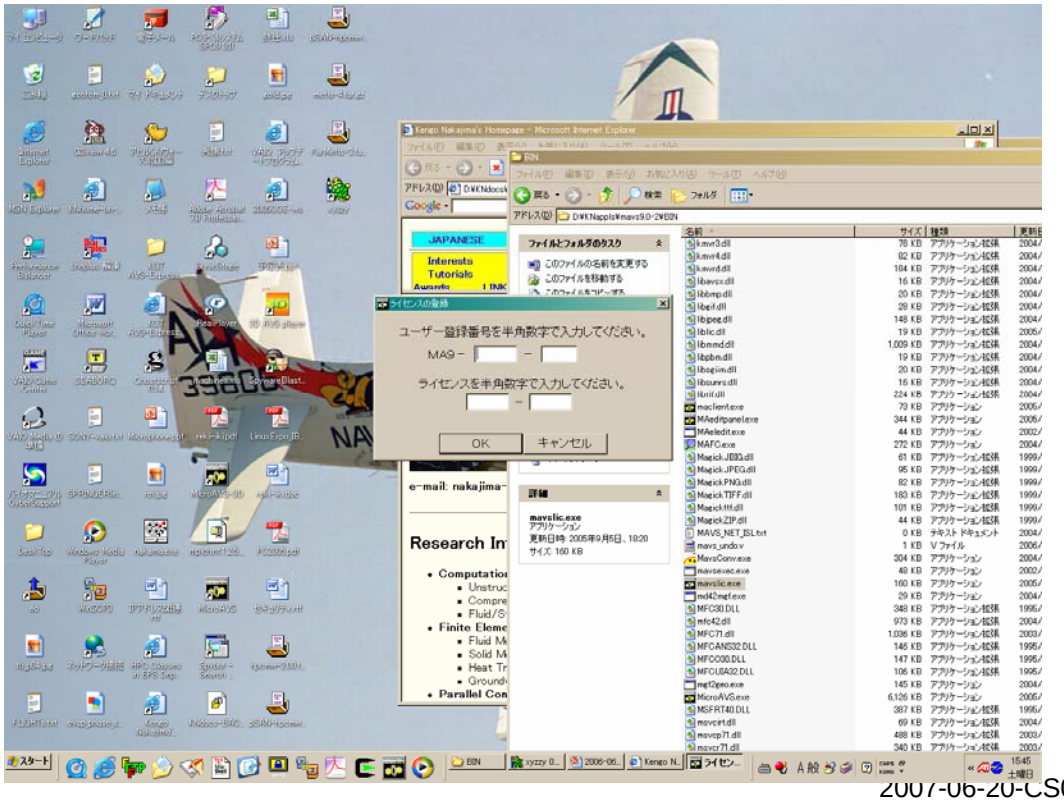
- AVS，microAVSのUCDファイルフォーマット（Unstructured Cell Data）を使用。
  - http://www.kgt.co.jp/

MicroAVSの使用

- 手順は以下
  - http://www-solid.eps.s.u-tokyo.ac.jp/~nakajima/04s/mavs/
- 評価版のダウンロード，インストール（WINDOWS版のみ）
  - http://www.kgt.co.jp/feature/microavs/#evaluation
- WINDOWS環境の無い人は理1-837号室のマシンを使用する
  - 使いたい人は中島まで

ライセンス

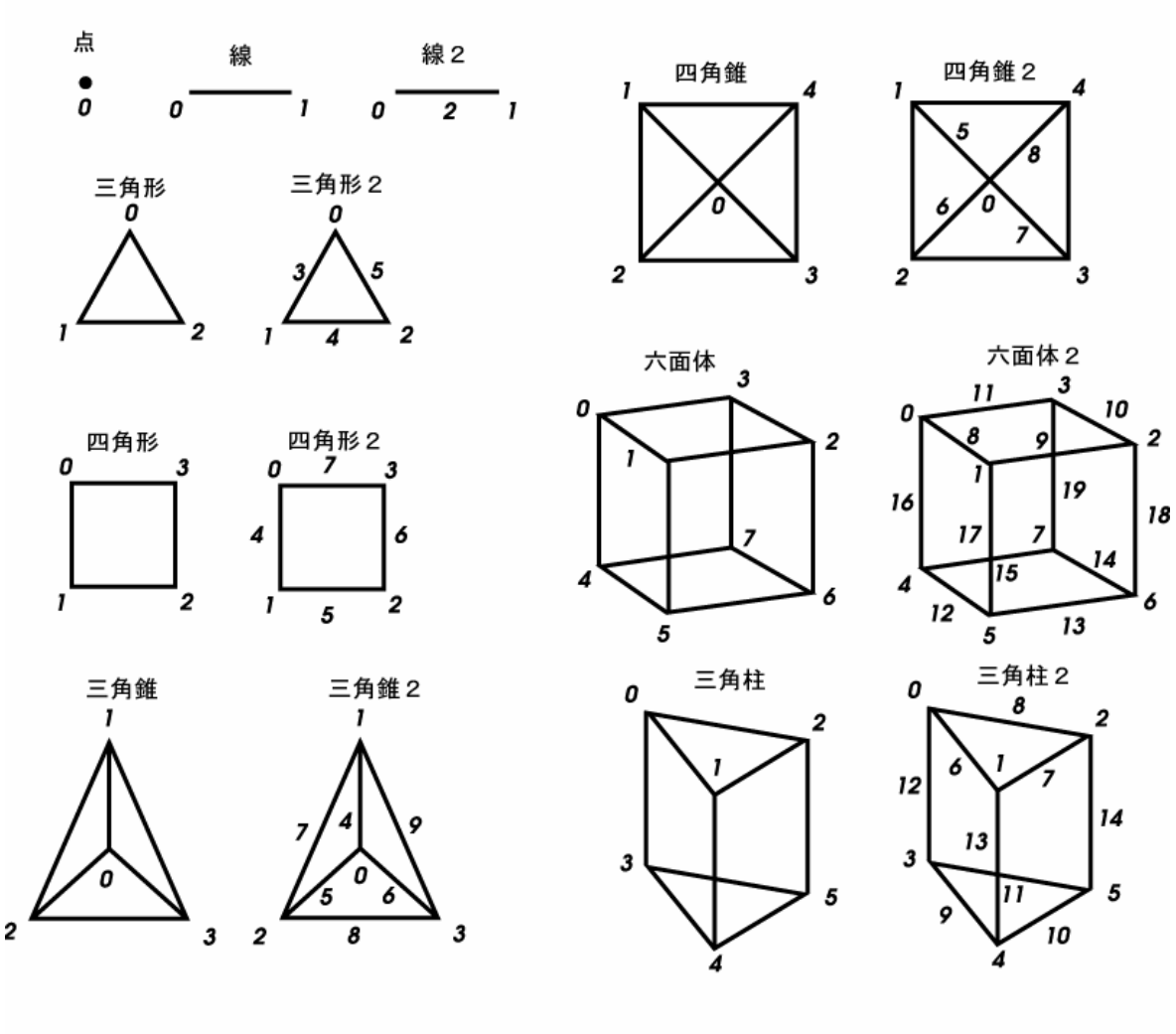
- インストールしたディレクトリの下の「BIN」にある「mavslic.exe」を起動するとライセンスのIDを訊いてくるので，記入する。
- これをやらないとインストールから30日で使えなくなるので注意。
- 1年間有効
- 学術的な目的以外（例えばアルバイト）に使わないこと。
  - 受講者以外の人にライセンスIDを教えないこと：応相談



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	
線2	line2
三角形2	tri2
四角形2	quad2
四面体2	tet2
角錐2	pyr2
三角柱2	prism2
六面体2	hex2



UCD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.10.0）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
```

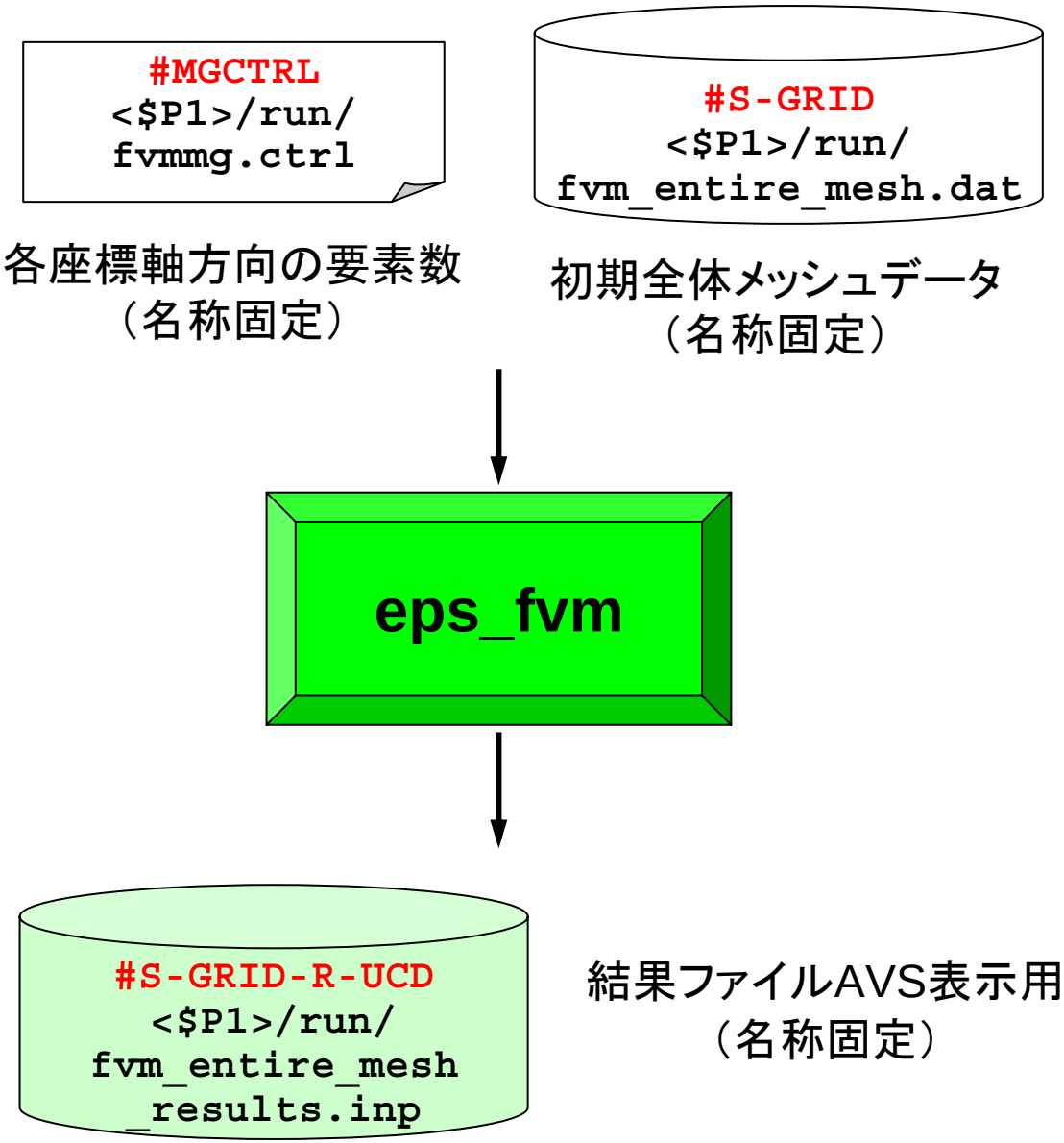
- 概要
- 1CPU用プログラム **eps_fvm**
  - –メッシュ生成，形状データ
  - –計算実行例
  - –プログラムの内容

「eps_fvm」計算の実行

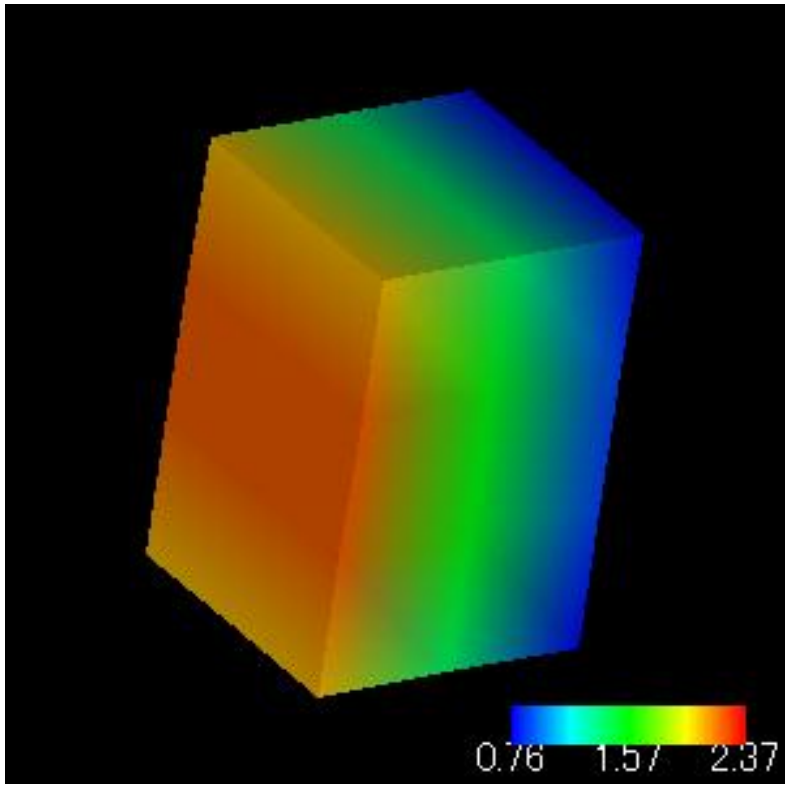
```
$> cd <$P1>/serial
$> make
…
$> cd ../run
$> ls eps_fvm                                  ロードモジュールの生成確認
     eps_fvm
$> cp ../mesh/fvmmg.ctrl .                    必要ファイルのコピー
$> cp ../mesh/fvm_entire_mesh.dat .
$> ./eps_fvm
```

- 実行形式**eps_fvm**は**<$P1>/run**というディレクトリに生成される。
- 入力
  - **<$P1>/run/fvmmg.ctrl** **#MGCTRL**
  - **<$P1>/run/fvm_entire_mesh.dat** **#S-GRID**
  - ファイルを**<$P1>/mesh**からコピーする必要がある。
- 出力
  - **<$P1>/run/fvm_entire_mesh_results.inp**(名称固定)
    - AVS用結果出力 **#S-GRID-R-UCD**

シミュレーションコード：eps_fvm



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。

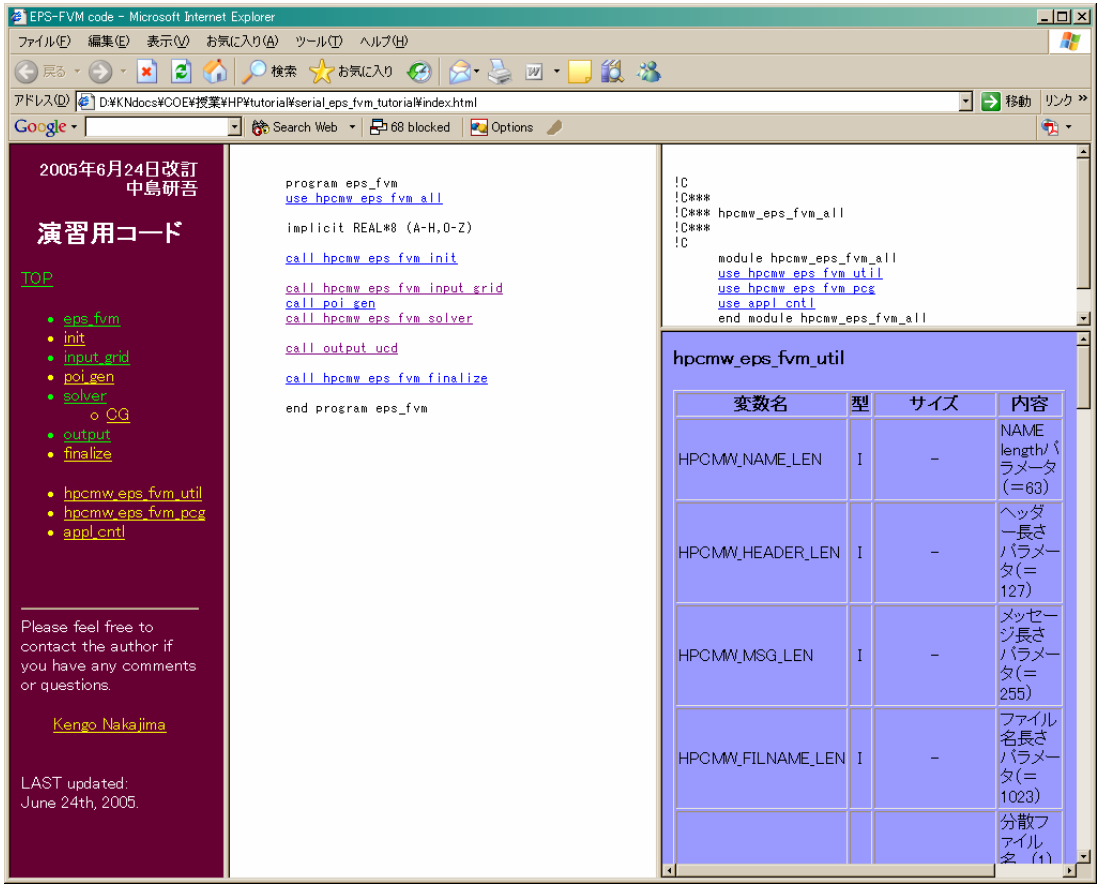
- 概要
- 1CPU用プログラム **eps_fvm**
  - –メッシュ生成，形状データ
  - –計算実行例
  - –プログラムの内容

プログラム内容に関するチュートリアル FORTRAN版

http://www-solid.eps.s.u-tokyo.ac.jp/~nakajima/07s/CS08/index.html

http://www-solid.eps.s.u-tokyo.ac.jp/~nakajima/tutorial/serial_eps_fvm_tutorial/index.html

MACへ持ってきて展開してもできる



プログラムの概要

$> cd <$P1>/serial
$> cat eps_fvm.f メインプログラム

```
      program eps_fvm

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

      call hpcmw_eps_fvm_init
      call hpcmw_eps_fvm_input_grid

      call poi_gen
      call hpcmw_eps_fvm_solve
      call output_ucd

      call hpcmw_eps_fvm_finalize
      end program eps_fvm
```

「eps_fvm」処理：メイン（1/3）
hpcmw_eps_fvm_init/finalize

```
program eps_fvm
use hpcmw_eps_fvm_all

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

call hpcmw_eps_fvm_init
call hpcmw_eps_fvm_input_grid
call poi_gen
call hpcmw_eps_fvm_solver
call output_ucd

call hpcmw_eps_fvm_finalize

end program eps_fvm
```

これらは実際はMPIのサブルーチン（MPI_INIT, MPI_FINALIZE）を呼んで初期化，後処理を実施している。

「eps_fvm」はシリアル処理であるが，「MPI_WTIME」等を利用するために，MPIのライブラリをリンクしている。

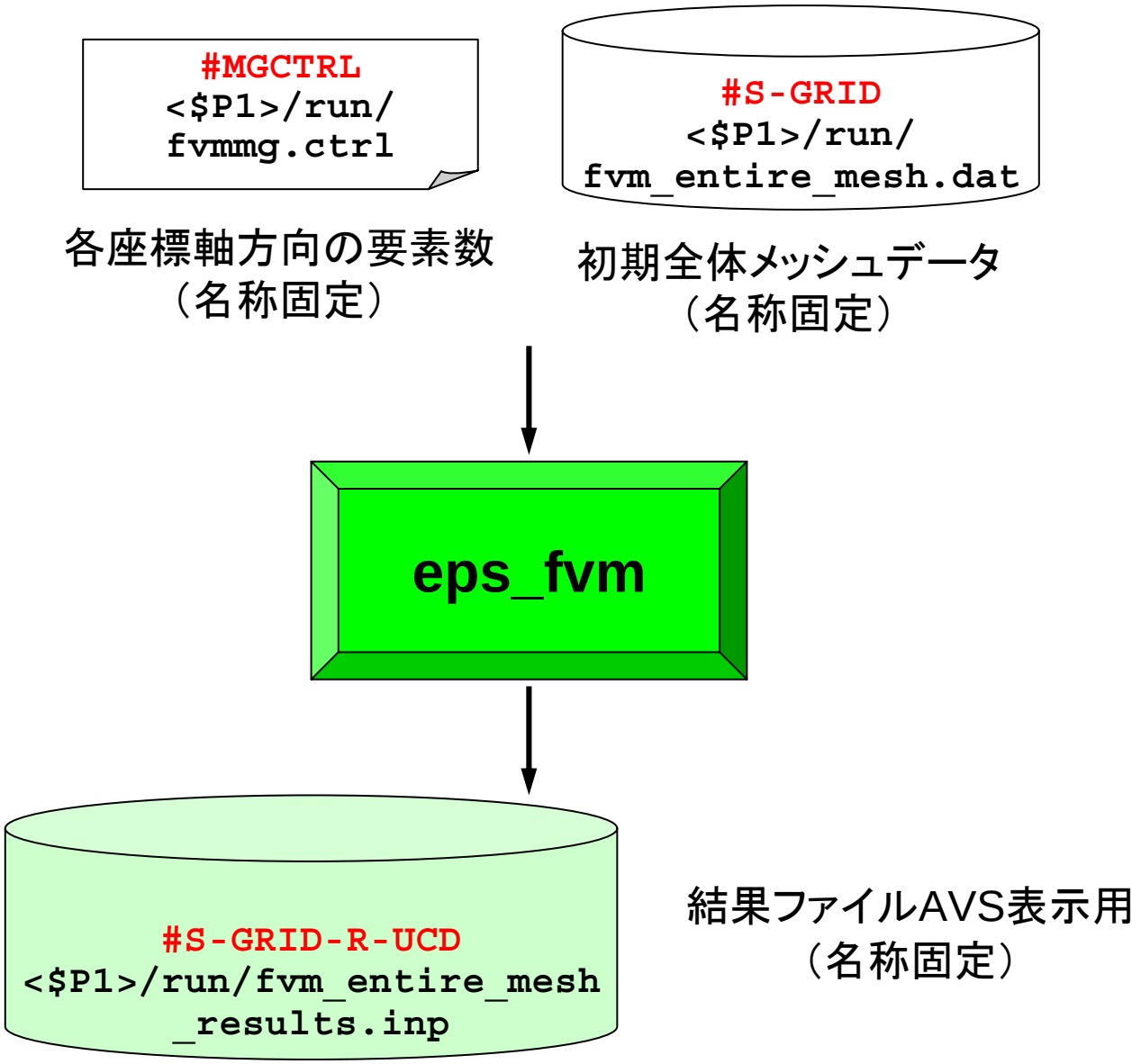
「eps_fvm」処理：メイン（2/3）
hpcmw_eps_fvm_all

```
program eps_fvm
use hpcmw_eps_fvm_all

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

call hpcmw_eps_fvm_init
call hpcmw_eps_fvm_input_grid
call poi_gen
call hpcmw_eps_fvm_solver
call output_ucd

call hpcmw_eps_fvm_finalize

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

eps_fvm の処理


#MGCTRL
<$P1>/run/
fvmmg.ctrl
#S-GRID
<$P1>/run/
fvm_entire_mesh.dat
各座標軸方向の要素数
（名称固定）
初期全体メッシュデータ
（名称固定）
eps_fvm
#S-GRID-R-UCD
<$P1>/run/fvm_entire_mesh
_results.inp
結果ファイルAVS表示用
（名称固定）

「eps_fvm」処理：メイン（3/3）
その他

```
program eps_fvm
use hpcmw_eps_fvm_all

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

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

call hpcmw_eps_fvm_finalize

end program eps_fvm
```

```
program eps_fvm
use hpcmw_eps_fvm_all

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

call hpcmw_eps_fvm_init
call hpcmw_eps_fvm_input_grid
call poi_gen
call hpcmw_eps_fvm_solver
call output_ucd

call hpcmw_eps_fvm_finalize

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：メッシュ読み込み（1/6）

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

      character(len=HPCMW_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（1/6）

```
!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
```

```
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
要素番号 要素体積        要素熱伝導率     要素中心x座標    要素中心y座標    要素中心z座標
         NODE_VOL(i)     NODE_COND(i)    NODE_XYZ(3*i-2) NODE_XYZ(3*i-1) NODE_XYZ(3*i)
```

「eps_fvm」処理：メッシュ読み込み（2/6）

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
```

```
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～       d2:E2重心～
                                    境界面距離        境界面距離
E1= CONN_NODE(2*ic-1)
E2= CONN_NODE(2*ic)
```

CONN_tot:
総コネクティビティ数

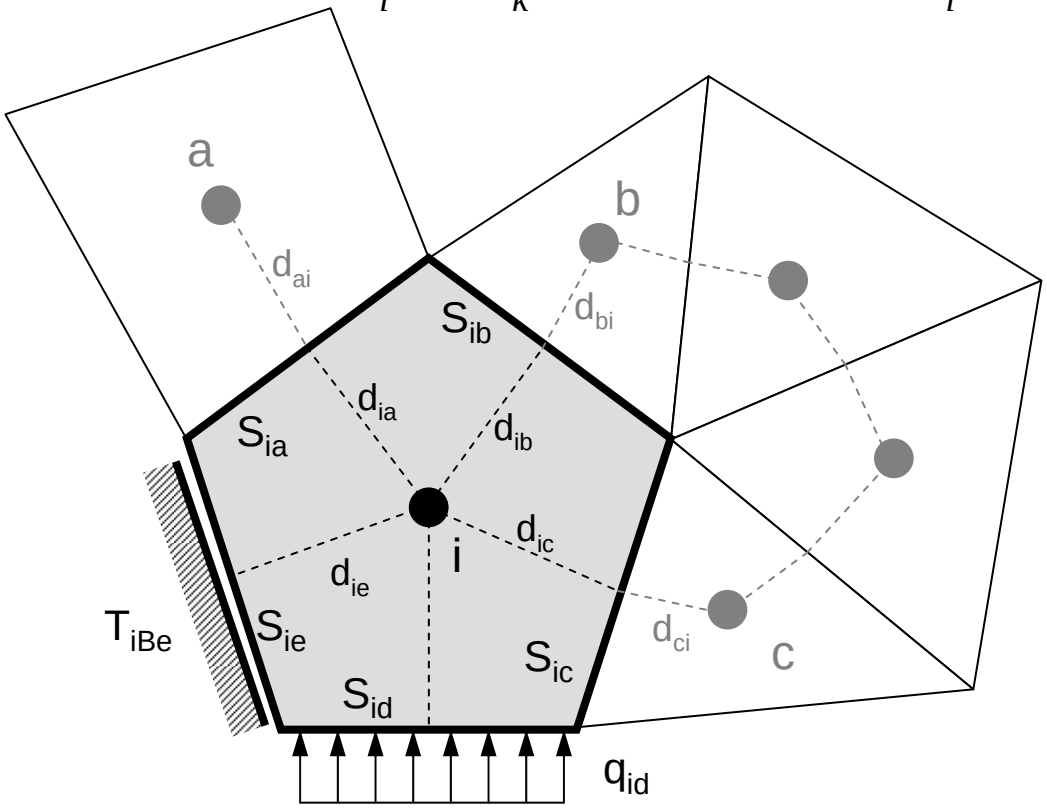
有限体積法による空間離散化

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

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

要素境界面通過熱流束　　体積発熱

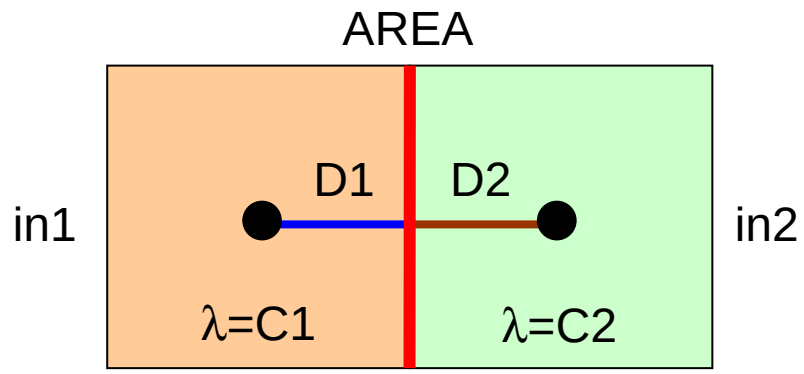


λ　：熱伝導率
V_i：要素体積
S　：表面面積
d_{ij}：要素中心から表面までの距離
q　：表面フラックス
Q　：体積発熱
T_{iB}：境界温度

「eps_fvm」処理：メッシュ読み込み（2/6）

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
```

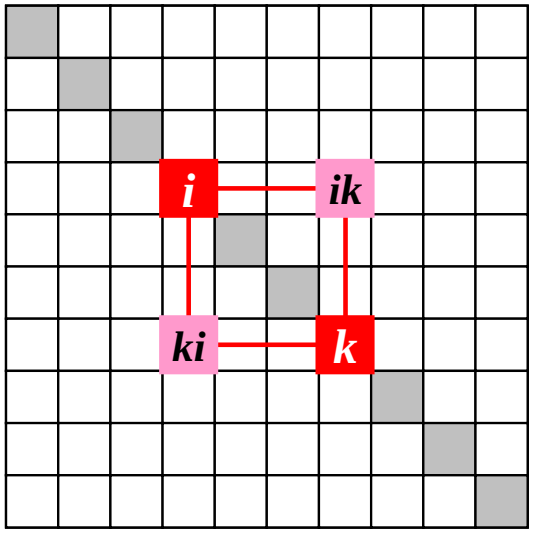


$$\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i}+\frac{d_{ki}}{\lambda_k}}\left(T_k - T_i\right) + \sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}}\left(T_{iBe} - T_i\right) + \sum_d S_{id}\dot{q}_{id} + V_i\dot{Q}_i = 0$$

CONN_COEF

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

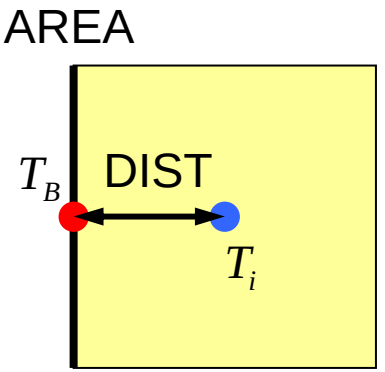


係数行列の非対角成分

「eps_fvm」処理：メッシュ読み込み（3/6）

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): Tk
        icel= FIX_NODE_ID(i)
        COND= NODE_COND(icel)
        FIX_NODE_COEF(i)= AREA / (DIST/COND)
      enddo
```




```
 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)
```

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

FIX_NODE_COEF

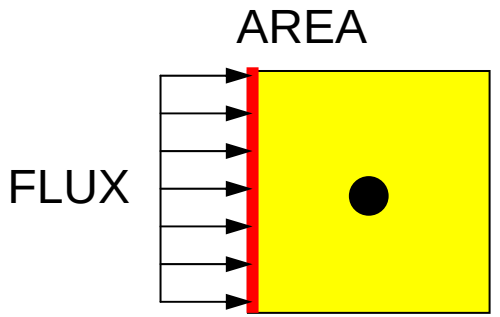
$$\sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}}\left(T_{iBe} - T_i\right)$$

FIX_NODE_VAL

「eps_fvm」処理：メッシュ読み込み（4/6）
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)
```



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

SURF_NODE_FLUX

「eps_fvm」処理：メッシュ読み込み（5/6）

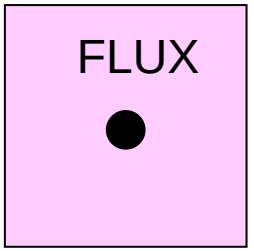
体積発熱

```
!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
```

```
  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
```

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

BODY_NODE_FLUX



「eps_fvm」処理：メッシュ読み込み（6/6）

```
!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

      close (IUNIT)
!C===

!C
!C +------------+
!C | COMM INPUT |
!C +------------+
!C===
      n_neighbor_pe= 0

      allocate (neighbor_pe(n_neighbor_pe))
      allocate (import_index(0:n_neighbor_pe))
      allocate (export_index(0:n_neighbor_pe))

      import_index= 0
      export_index= 0

      nn= 0
      allocate (import_item(nn))
      allocate (export_item(nn))

      intNODE_tot= NODE_tot

      close (IUNIT)
!C===

      end subroutine hpcmw_eps_fvm_input_grid
```

ここは，本来並列情報を読み込むところであるが，「eps_fvm」では必要ない。
念のため，初期化を実施している。

メッシュ関連変数（1/2）

変数名	型	配列サイズ	内容
NODE_tot	I	-	内点数＋外点数
intNODE_tot	I	-	内点数
NODE_GLOBAL(:)	I	NODE_tot	グローバル要素番号
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

シリアル計算では
`NODE_tot= intNODE_tot`

メッシュ関連変数（2/2）

変数名	型	配列サイズ	内容
PETOT	I	-	プロセッサ数
errno	I	-	エラーもどり値
my_rank	I	-	ランク番号
n_neighbor_pe	I	-	隣接領域数
neighbor_pe(:)	I	n_neighbor_pe	隣接領域ID
import_index(:)	I	0:n_neighbor_pe	受信テーブル用インデックス
import_item(:)	I	import_index(n_neighbor_pe)	受信テーブル
export_index(:)	I	0:n_neighbor_pe	送信テーブル用インデックス
export_item(:)	I	export_index(n_neighbor_pe)	送信テーブル

これは主として並列計算用

```
program eps_fvm
use hpcmw_eps_fvm_all

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

call hpcmw_eps_fvm_init
call hpcmw_eps_fvm_input_grid
call poi_gen
call hpcmw_eps_fvm_solver
call output_ucd

call hpcmw_eps_fvm_finalize

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

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）

行列ベクトル積への適用

非ゼロ成分のみを格納，疎行列向け方法

D (i)	対角成分(実数，i=1,N)
INDEX(i)	非対角成分数に関する一次元配列 (整数，i=0,N)
ITEM(k)	非対角成分の要素番号 (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
```

マトリクス格納形式（1/4）

	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

マトリクス格納形式(2/4)

	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

```
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
```

マトリクス格納形式(3/4)

	1	2	3	4	5	6	7	8	非零非対角成分数	
1	1.1	2.4	0	0	3.2	0	0	0	2	INDEX(1)= 2
2	4.3	3.6	0	2.5	0	3.7	0	9.1	4	INDEX(2)= 6
3	0	0	5.7	0	1.5	0	3.1	0	2	INDEX(3)= 8
4	0	4.1	0	9.8	2.5	2.7	0	0	3	INDEX(4)= 11
5	3.1	9.5	10.4	0	11.5	0	4.3	0	4	INDEX(5)= 15
6	0	0	6.5	0	0	12.4	9.5	0	2	INDEX(6)= 17
7	0	6.4	2.5	0	0	1.4	23.1	13.1	4	INDEX(7)= 21
8	0	9.5	1.3	9.6	0	3.1	0	51.3	4	INDEX(8)= 25

NPLU= 25

マトリクス格納形式(4/4)

	1	2	3	4	5	6	7	8
1	1.1	2.4 1	0	0	3.2 2	0	0	0
2	4.3 3	3.6	0	2.5 4	0	3.7 5	0	9.1 6
3	0	0	5.7	0	1.5 7	0	3.1 8	0
4	0	4.1 9	0	9.8	2.5 10	2.7 11	0	0
5	3.1	9.5 13	10.4 14	0	11.5	0	4.3 15	0
6	12	0	6.5 16	0	0	12.4	9.5 17	0
7	0	6.4 18	2.5 19	0	0	1.4 20	23.1	13.1 21
8	0	9.5 22	1.3	9.6 24	0	3.1 25	0	51.3
			23					

例

```
ITEM( 7)= 5, AMAT( 7)= 1.5
ITEM(19)= 3, AMAT(19)= 2.5
```

「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
      subroutine POI_GEN

      use hpcmw_eps_fvm_all
      implicit REAL*8 (A-H,O-Z)
      integer, pointer :: IWKX(:,:)
!C
!C +-------+
!C | INIT. |
!C +-------+
!C===

!C
!C-- MATRIX
      nn = intNODE_tot
      nnp=    NODE_tot

      allocate (BFORCE(nnp), D(nn), PHI(nnp))
      allocate (index(0:nn))

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

      index= 0
```

「eps_fvm」処理：マトリクス生成（2/5）

```
!C
!C-- ETC.
      allocate (IWKX(intNODE_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))
```

非対角成分数算出
非対角成分抽出

index(in)：
各要素の非対角成分数
（のインデックス）

IWKX(in,1-6)：
各要素の非対角成分の
コネクティビティID

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(intNODE_tot,7))
      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のインデックス

NPLU：
非対角成分の総数

「eps_fvm」処理：マトリクス生成（3/5）

```
      do i= 1, intNODE_tot
        do j= 1, index(i)-index(i)
          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, intNODE_tot
        BFORCE(icel)= BFORCE(icel) + BODY_NODE_FLUX(icel)
      enddo

      do i= 1, intNODE_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===
```

item(k)：
非対角成分

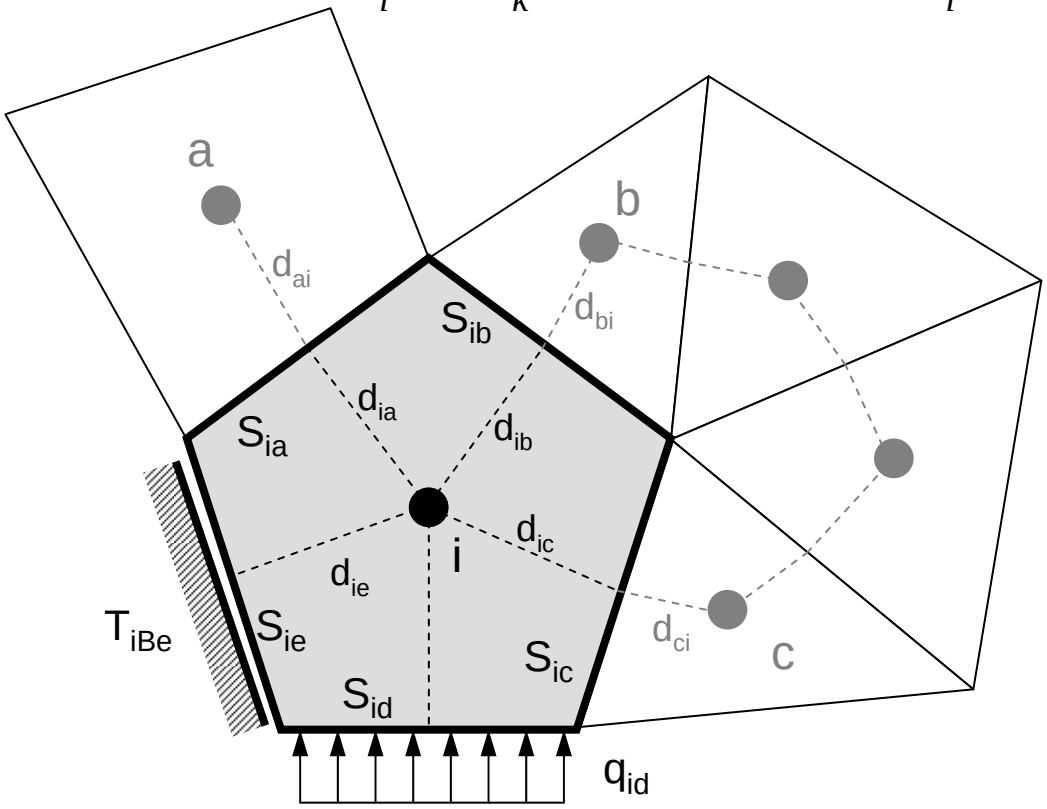
有限体積法による空間離散化

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

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

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

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

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

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

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

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

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

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

定数項：右辺へ移項

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

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, intNODE_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, intNODE_tot
        BFORCE(icel)= BFORCE(icel) + BODY_NODE_FLUX(icel)
      enddo

      do i= 1, intNODE_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===
```

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

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

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, intNODE_tot
        do j= 1, index(i)-index(i)
          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, intNODE_tot
        BFORCE(icel)= BFORCE(icel) + BODY_NODE_FLUX(icel)
      enddo

      do i= 1, intNODE_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===
```

D（対角成分） AMAT（非対角成分）

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

CONN_COEF CONN_COEF

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

「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===
```

D（対角成分）

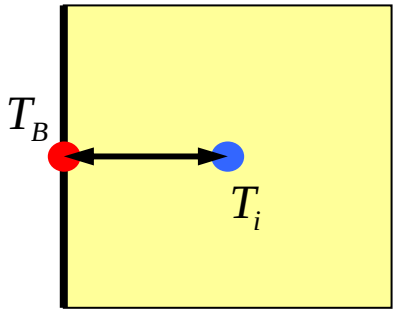
$$\left[\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i}+\frac{d_{ki}}{\lambda_k}}+\sum_e \frac{S_{ie}}{\frac{d_{ie}}{\lambda_i}}\right]T_i-\left[\sum_k \frac{S_{ik}}{\frac{d_{ik}}{\lambda_i}+\frac{d_{ki}}{\lambda_k}}T_k\right]$$

FIX_NODE_COEF

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

BFORCE（右辺）

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
```

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

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

BFORCE（右辺）

SURF_NODE_FLUX

「eps_fvm」処理：ソルバー

```
      subroutine hpcmw_eps_fvm_solver

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

      EPS = 1.d-8
      ITR = NODE_tot

      S_TIME= MPI_WTIME()
      call hpcmw_eps_fvm_solver_CG                                  &
     &    ( intNODE_tot, NODE_tot, NPLU, D, BFORCE, PHI, EPS,    &
     &      ITR,  IER, index, item, AMAT, COMMtime)
      E_TIME= MPI_WTIME()
      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(1pe16.6))') i, PHI(i)
      enddo

      if (my_rank.eq.0) then
        WALLtime= E_TIME-S_TIME
        ratio   = 100.d0 * (1.d0-COMMtime/WALLtime)
        write (*,'(a, 1pe16.6)') '### solver time', WALLtime
        write (*,'(a, 1pe16.6)') '### comm.  time', COMMtime
        write (*,'(a, 1pe16.6)') '### work ratio ', ratio
      endif

      end subroutine hpcmw_eps_fvm_solver
```

intNODE_tot（内点数）
NODE_tot（総要素数）

はシリアルコードでは同じ数であるが，並列計算では異なる。

代表的な反復法：共役勾配法

- Conjugate Gradient法，略して「CG」法⇒本講義で使用
  - 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
- アルゴリズム
  - 最急降下法（Steepest Descent Method）の変種
  - $\mathbf{x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}}$
    - **$x^{(i)}$：反復解，$p^{(i)}$：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$\{x\}$
  - 詳細は参考文献〔長谷川ら〕参照

共役勾配法のアルゴリズム

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

$x^{(i)}$ ：ベクトル
α_i ：スカラー

前処理付き共役勾配法
Preconditioned Conjugate Gradient Method（CG）

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

前処理：対角スケーリング

$x^{(i)}$ ：ベクトル
α_i ：スカラー

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

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

- **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。

「eps_fvm」処理:ソルバー(1/7)

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

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

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

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

      real(kind=kreal) :: EPS, Tcomm

      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(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===
```

「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
```

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

W(i,4)= W(i,DD) ⇒ 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
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  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 $r^{(0)}= b-[A]x^{(0)}$

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
  - $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
  - $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
- 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（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, …

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

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

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

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, …

  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

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

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

  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, …

- 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
  - $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
  - $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
- 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, 1pe16.6)

        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
```

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
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  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

いろいろなメッシュでやってみよ

- MicroAVSの使い方に慣れる。