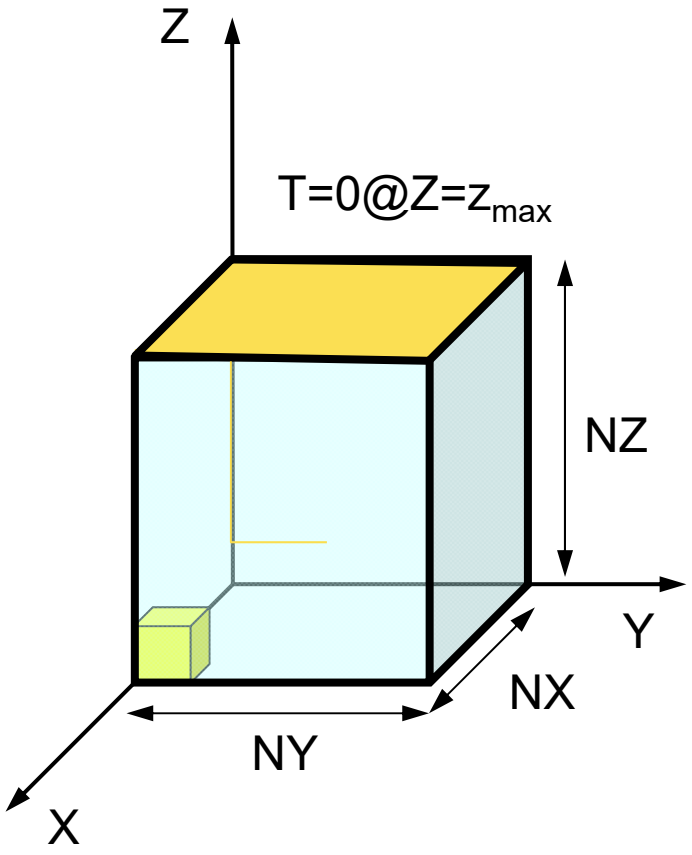
並列有限要素法による
三次元定常熱伝導解析プログラム
（2/2）Fortran編

中島 研吾
東京大学情報基盤センター

対象とする問題：三次元定常熱伝導

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



- 定常熱伝導＋発熱
- 一様な熱伝導率 λ
- 直方体
  - 一辺長さ1の立方体（六面体）要素
  - 各方向にNX・NY・NZ個
- 境界条件
  - T=0@Z=z_{max}
- 体積当たり発熱量は位置（メッシュの中心の座標 x_c,y_c）に依存
  - $\dot{Q}(x,y,z)=QVOL|x_C+y_C|$

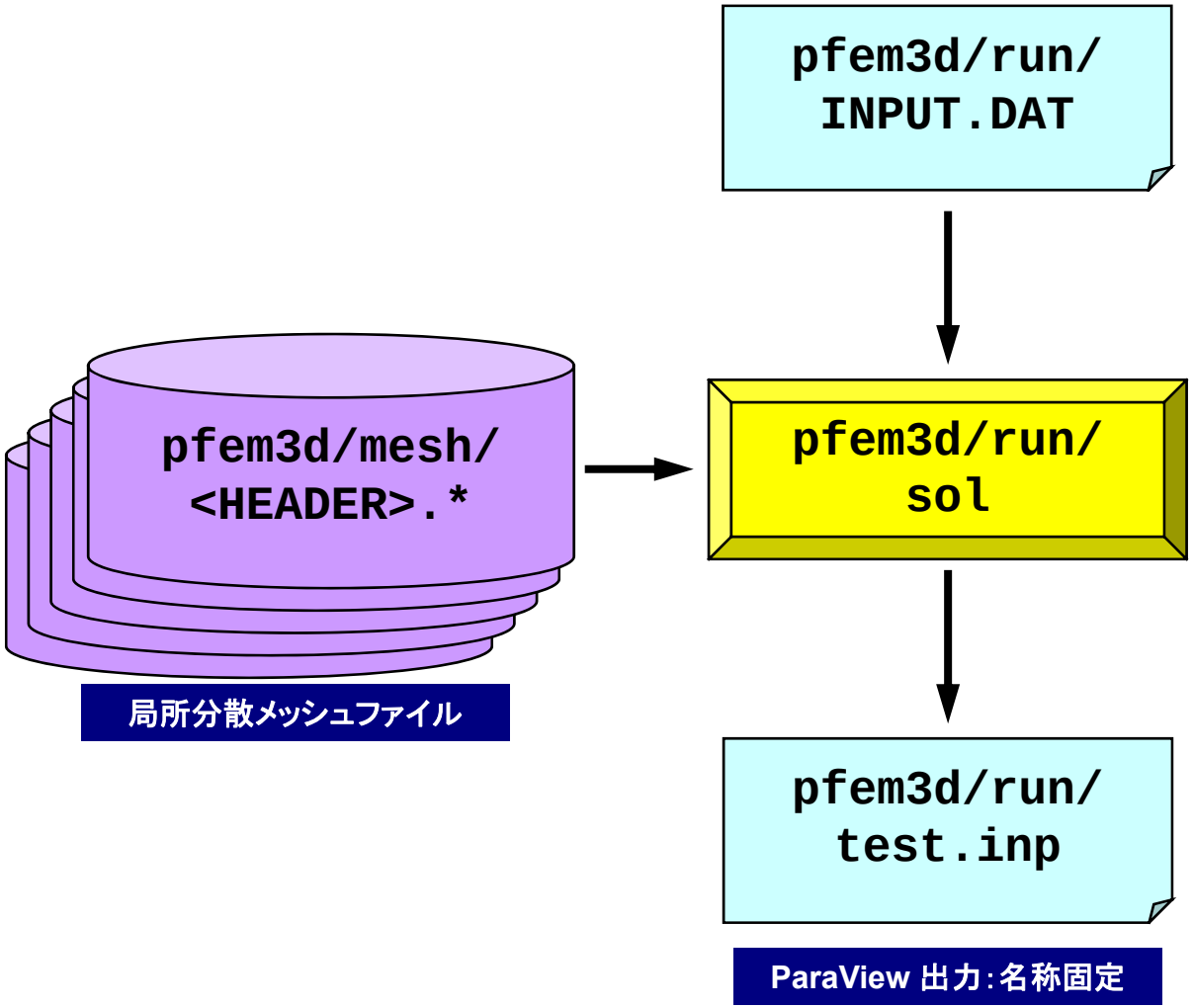
有限要素法の処理

- 支配方程式
- ガラーキン法：弱形式
- 要素単位の積分
  - 要素マトリクス生成
- 全体マトリクス生成
- 境界条件適用
- 連立一次方程式

並列有限要素法の処理：プログラム

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

並列有限要素法の手順（並列計算実行）



制御ファイル：INPUT.DAT

```
../mesh/aaa  HEADER
2000         ITER
1.0 1.0      COND, QVOL
1.0e-08      RESID
```

- HEADER：　局所分散ファイルヘッダ名
  <HEADER>.my_rank
- ITER：　反復回数上限
- COND：　熱伝導率
- QVOL：　体積当たり発熱量係数
- RESID：　反復法の収束判定値

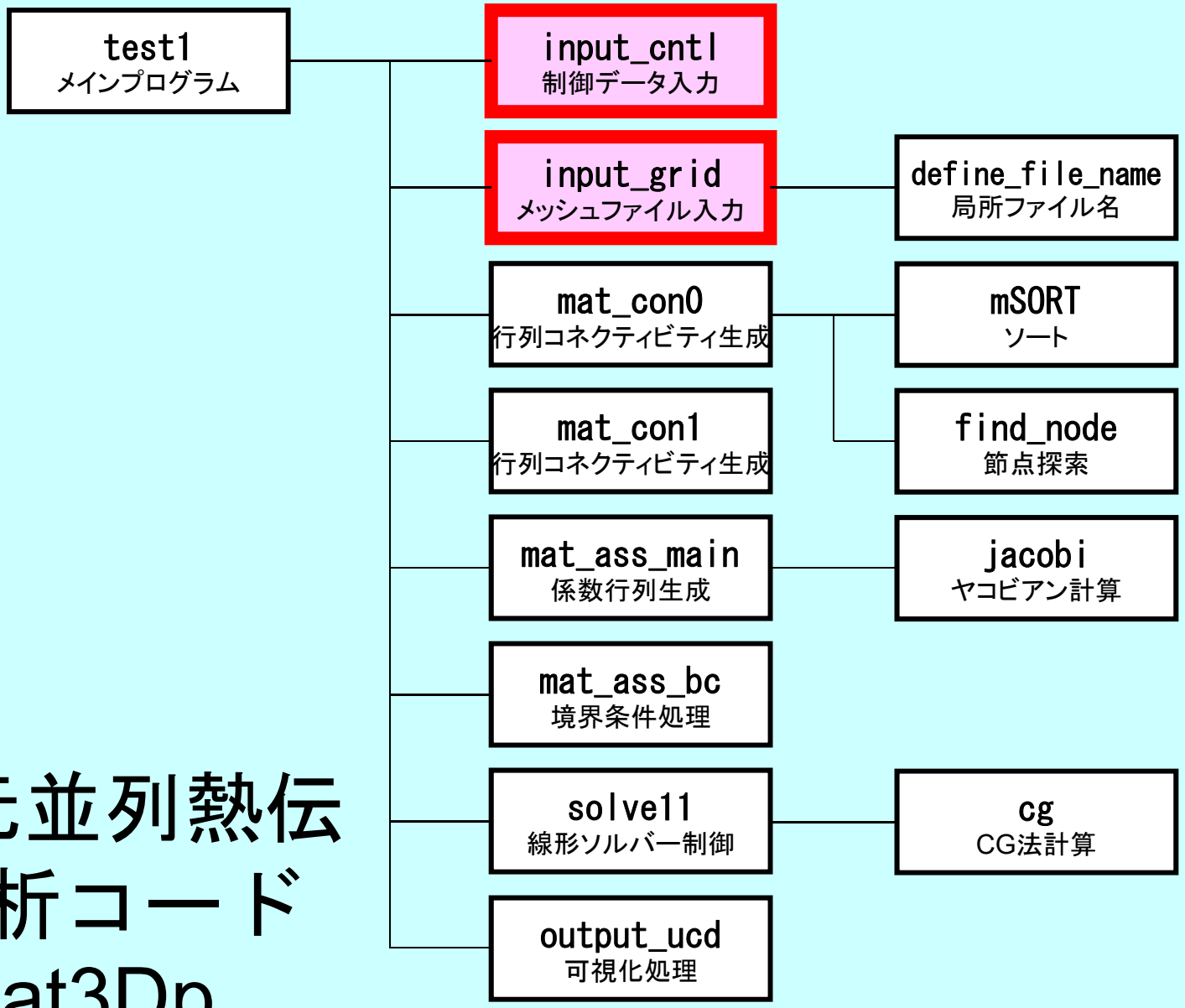
$$\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(\lambda\frac{\partial T}{\partial z}\right)+\dot{Q}(x,y,z)=0$$

$$\dot{Q}(x,y,z)=QVOL\left|x_C+y_C\right|$$

~/pFEM/pfem3d/run/go.sh

```
#!/bin/sh
#PJM -L “node=1“                ノード数（≦12）
#PJM -L “elapse=00:10:00“       実行時間（≦15分）
#PJM -L “rscgrp=school“         実行キュー名
#PJM -
#PJM -o “test.lst“              標準出力
#PJM --mpi “proc=8“             MPIプロセス数（≦192）

mpiexec ./sol
```

8分割	16分割	32分割	64分割	192分割
“node=1“ “proc=8”	“node=1“ “proc=16”	“node=2“ “proc=32”	“node=4“ “proc=64”	“node=12“ “proc=192”



三次元並列熱伝導解析コード heat3Dp

全体処理

```
      program heat3Dp

      use solver11
      use pfem_util

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

      call PFEM_INIT

      call INPUT_CNTL
      call INPUT_GRID

      call MAT_CON0
      call MAT_CON1

      call MAT_ASS_MAIN
      call MAT_ASS_BC

      call SOLVE11

      call OUTPUT_UCD

      call PFEM_FINALIZE

      end program heat3Dp
```

Global変数表：pfem_util.f（1/4）

変数名	種別	サイズ	I/O	内　容
fname	C	(80)	I	メッシュファイル名
N,NP	I		I	節点数（N：内点，NP：内点＋外点）
ICELTOT	I		I	要素数
NODGRPtot	I		I	節点グループ数
XYZ	R	(NP,3)	I	節点座標
ICELNOD	I	(ICELTOT,8)	I	要素コネクティビティ
NODGRP_INDEX	I	(0:NODGRPtot)	I	各節点グループに含まれる節点数（累積）
NODGRP_ITEM	I	(NODGRP_INDEX(NODGRPTOT)	I	節点グループに含まれる節点
NODGRP_NAME	C80	(NODGRPTOT)	I	節点グループ名
NLU	I		O	各節点非対角成分数
NPLU	I		O	非対角成分総数
D	R	(NP)	O	全体行列：対角ブロック
B,X	R	(NP)	O	右辺ベクトル，未知数ベクトル

Global変数表：pfem_util.f（2/4）

変数名	種別	サイズ	I/O	内　容
AMAT	R	(NPLU)	O	全体行列：非零非対角成分
index	I	(0:NP)	O	全体行列：非零非対角成分数
item	I	(NPLU)	O	全体行列：非零非対角成分（列番号）
INLU	I	(NP)	O	各節点の非零非対角成分数
IALU	I	(NP,NLU)	O	各節点の非零非対角成分数（列番号）
IWKX	I	(NP,2)	O	ワーク用配列
ITER, ITERactual	I		I	反復回数の上限，実際の反復回数
RESID	R		I	打ち切り誤差（1.e-8に設定）

Global変数表：pfem_util.f（3/4）

変数名	種別	サイズ	I/O	内　容
O8th	R		I	=0.125
PNQ, PNE, PNT	R	(2,2,8)	O	各ガウス積分点における $\frac{\partial N_i}{\partial \xi}, \frac{\partial N_i}{\partial \eta}, \frac{\partial N_i}{\partial \zeta} (i=1\sim 8)$
POS, WEI	R	(2,2)	O	各ガウス積分点の座標，重み係数
NCOL1, NCOL2	I	(100)	O	ソート用ワーク配列
SHAPE	R	(2,2,2,8)	O	各ガウス積分点における形状関数 N_i（i=1~8）
PNX, PNY, PNZ	R	(2,2,2,8)	O	各ガウス積分点における $\frac{\partial N_i}{\partial x}, \frac{\partial N_i}{\partial y}, \frac{\partial N_i}{\partial z} (i=1\sim 8)$
DETJ	R	(2,2,2)	O	各ガウス積分点におけるヤコビアン行列式
COND, QVOL	R		I	熱伝導率，体積当たり発熱量係数

Global変数表：pfem_util.f（4/4）

変数名	種別	サイズ	I/O	内　容
PETOT	I		O	領域数（MPIプロセス数）
my_rank	I		O	MPIプロセス番号
errno	I		O	エラーフラグ
NEIBPETOT	I		I	隣接領域数
NEIBPE	I	(NEIBPETOT)	I	隣接領域番号
IMPORT_INDEX EXPORT_INEDX	I	(0:NEIBPETOT)	I	送信，受信テーブルのサイズ（一次元圧縮配列）
IMPORT_ITEM	I	(Npimport)	I	受信テーブル（外点） (NPimport=IMPORT_INDEX(NEIBPETOT))
EXPORT_ITEM	I	(Npexport)	I	送信テーブル（境界点） (NPexport=EXPORT_INDEX(NEIBPETOT))
ICELTOT_INT	I		I	領域所属要素数
intELEM_list	I	(ICELTOT_INT)	I	領域所属要素のリスト:　可視化に使用

開始，終了：MPI_Init/Finalize

```
      subroutine PFEM_INIT
      use pfem_util
      implicit REAL*8 (A-H,O-Z)

      call MPI_INIT      (ierr)
      call MPI_COMM_SIZE (MPI_COMM_WORLD, PETOT, ierr )
      call MPI_COMM_RANK (MPI_COMM_WORLD, my_rank, ierr )

      pfemRarray= 0.d0
      pfemIarray= 0

      return
      end
```

```
      subroutine PFEM_FINALIZE
      use pfem_util
      implicit REAL*8 (A-H,O-Z)

      call MPI_FINALIZE (errno)
      if (my_rank.eq.0) stop ' * normal termination'

      return
      end
```

制御ファイル入力：INPUT_CNTL

```
      subroutine INPUT_CNTL
      use pfem_util

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

      if (my_rank.eq.0) then
        open (11,file='INPUT.DAT', status='unknown')
        read (11,'(a80)') HEADER
        read (11,*) ITER
        read (11,*) COND, QVOL
        read (11,*) RESID
        close (11)
      endif

      call MPI_BCAST (HEADER, 80, MPI_CHARACTER, 0, MPI_COMM_WORLD,ierr)
      call MPI_BCAST (ITER  , 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call MPI_BCAST (COND  , 1, MPI_DOUBLE_PRECISION, 0,
     &                                          MPI_COMM_WORLD, ierr)
      call MPI_BCAST (QVOL  , 1, MPI_DOUBLE_PRECISION, 0,
     &                                          MPI_COMM_WORLD, ierr)
      call MPI_BCAST (RESID , 1, MPI_DOUBLE_PRECISION, 0,
     &                                          MPI_COMM_WORLD, ierr)

      pfemRarray(1)= RESID
      pfemIarray(1)= ITER

      return
      end
```

メッシュ入力：INPUT_GRID（1/3）

```
      subroutine INPUT_GRID
      use pfem_util
      implicit REAL*8 (A-H,O-Z)

      call define_file_name (HEADER, fname, my_rank)
      open (11, file= fname, status= 'unknown', form= 'formatted')

!C
!C-- NEIB-PE
      read (11,'(10i10)') kkk
      read (11,'(10i10)') NEIBPETOT
      allocate (NEIBPE(NEIBPETOT))

      read (11,'(10i10)') (NEIBPE(i), i= 1, NEIBPETOT)

      do i= 1, NEIBPETOT
        if (NEIBPE(i).gt.PETOT-1) then
          call ERROR_EXIT (202, my_rank)
        endif
      enddo
```

分散メッシュファイル名：DEFINE_FILE_NAME

HEADER+ランク番号，実際は10^6個まで可能

```
subroutine DEFINE_FILE_NAME (HEADERo, filename, my_rank)

character (len=80) ::  HEADERo, filename
character (len=80) ::  HEADER
character (len= 1) ::  SUBindex1
character (len= 2) ::  SUBindex2
character (len= 3) ::  SUBindex3
integer:: LENGTH, ID

HEADER= adjustL (HEADERo)
LENGTH= len_trim(HEADER)

if (my_rank.le.9) then
  ID= 1
  write(SUBindex1 ,'(i1.1)') my_rank
 else if (my_rank.le.99) then
  ID= 2
  write(SUBindex2 ,'(i2.2)') my_rank
 else if (my_rank.le.999) then
  ID= 3
  write(SUBindex3 ,'(i3.3)') my_rank
endif

if (ID.eq.1) filename= HEADER(1:LENGTH)//'.'//SUBindex1
if (ID.eq.2) filename= HEADER(1:LENGTH)//'.'//SUBindex2
if (ID.eq.3) filename= HEADER(1:LENGTH)//'.'//SUBindex3

end subroutine define_file_name
```

allocate, deallocate関数：C言語

```
#include <stdio.h>
#include <stdlib.h>
void* allocate_vector(int size,int m)
{
  void *a;
  if ( ( a=(void * )malloc( m * size ) ) == NULL ) {
     fprintf(stdout,"Error:Memory does not enough! in vector ¥n");
     exit(1);
  }
  return a;
}

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

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

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

allocateをFORTRAN並みに簡単にやるための関数

メッシュ入力：INPUT_GRID（2/3）

```
!C
!C-- NODE
      read (11,'(10i10)')  NP, N
      allocate (XYZ(NP,3), NODE_ID(NP,2))
      XYZ= 0.d0
        do i= 1, NP
          read (11,*) NODE_ID(i,1), NODE_ID(i,2), (XYZ(i,kk),kk=1,3)
        enddo

!C
!C-- ELEMENT
      read (11,*)  ICELTOT, ICELTOT_INT

      allocate (ICELNOD(ICELTOT,8), intELEM_list(ICELTOT))
      allocate (ELEM_ID(ICELTOT,2))
      read (11,'(10i10)') (NTYPE, i= 1, ICELTOT)
      do icel= 1, ICELTOT
        read (11,'(i10,2i5,8i10)') (ELEM_ID(icel,jj),jj=1,2),           &
     &                         IMAT, (ICELNOD(icel,k), k= 1, 8)
      enddo

      read (11,'(10i10)') (intELEM_list(ic0), ic0= 1, ICELTOT_INT)
```

メッシュ入力：INPUT_GRID（3/3）

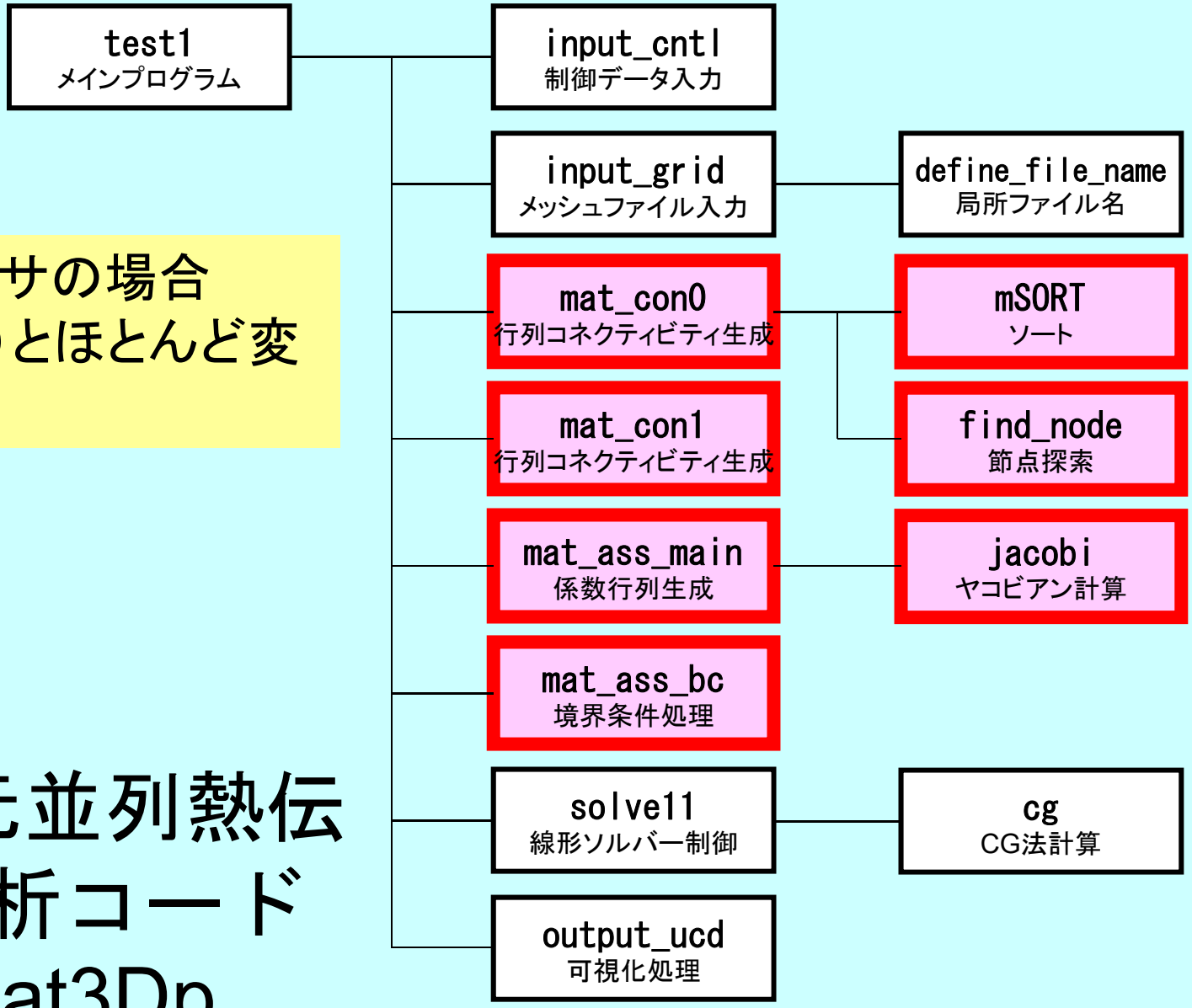
```
!C-- COMMUNICATION table
      allocate (IMPORT_INDEX(0:NEIBPETOT))
      allocate (EXPORT_INDEX(0:NEIBPETOT))

      IMPORT_INDEX= 0
      EXPORT_INDEX= 0

      if (PETOT.ne.1) then
      read (11,'(10i10)') (IMPORT_INDEX(i), i= 1, NEIBPETOT)
      nn= IMPORT_INDEX(NEIBPETOT)
      allocate (IMPORT_ITEM(nn))
      do i= 1, nn
        read (11,*) IMPORT_ITEM(i)
      enddo

      read (11,'(10i10)') (EXPORT_INDEX(i), i= 1, NEIBPETOT)
      nn= EXPORT_INDEX(NEIBPETOT)
      allocate (EXPORT_ITEM(nn))
      do i= 1, nn
        read (11,*) EXPORT_ITEM(i)
      enddo
      endif
!C-- NODE grp. info.
      read (11,'(10i10)') NODGRPtot
      allocate (NODGRP_INDEX(0:NODGRPtot),NODGRP_NAME(NODGRPtot))
      NODGRP_INDEX= 0

      read (11,'(10i10)') (NODGRP_INDEX(i), i= 1, NODGRPtot)
      nn= NODGRP_INDEX(NODGRPtot)
      allocate (NODGRP_ITEM(nn))

      do k= 1, NODGRPtot
        iS= NODGRP_INDEX(k-1) + 1
        iE= NODGRP_INDEX(k  )
        read (11,'(a80)') NODGRP_NAME(k)
        nn= iE - iS + 1
        if (nn.ne.0) then
          read (11,'(10i10)') (NODGRP_ITEM(kk),kk=iS, iE)
        endif
      enddo
```

並列有限要素法の処理：プログラム

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

1プロセッサの場合（heat3D）とほとんど変わらない

三次元並列熱伝導解析コード heat3Dp



マトリクス生成まで

- 一次元のときは，index，itemに関連した情報を簡単に作ることができた
  - 非ゼロ非対角成分の数は2
  - 番号が自分に対して：+1と-1

- 三次元の場合はもっと複雑
  - 非ゼロ非対角ブロックの数は7～26（現在の形状）
  - 実際はもっと複雑
  - 前以て，非ゼロ非対角ブロックの数はわからない

マトリクス生成まで

- 一次元のときは，index，itemに関連した情報を簡単に作ることができた
  - 非ゼロ非対角成分の数は2
  - 番号が自分に対して：+1と-1
- 三次元の場合はもっと複雑
  - 非ゼロ非対角ブロックの数は7～26（現在の形状）
  - 実際はもっと複雑
  - 前以て，非ゼロ非対角ブロックの数はわからない
- INLU(N)，IALU(N,NLU) を使って非ゼロ非対角成分数を予備的に勘定する

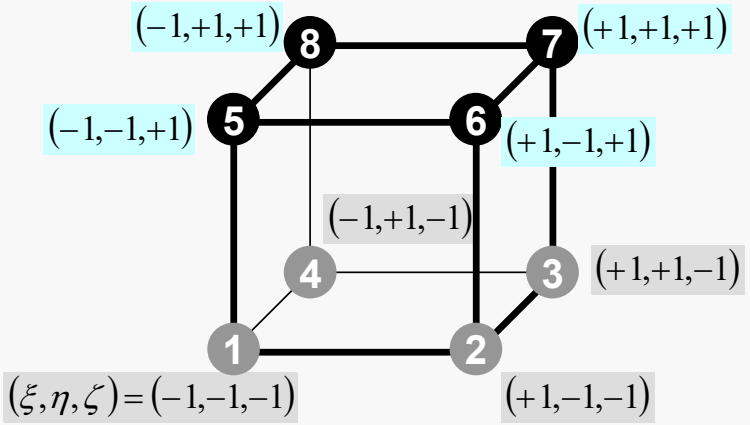
全体処理

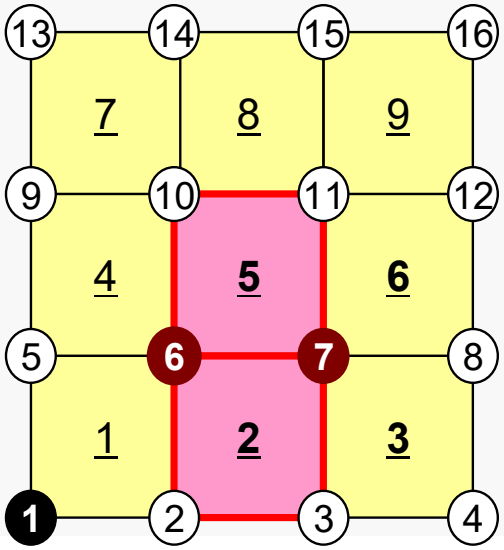
```
      program heat3Dp

      use solver11
      use pfem_util

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

      call PFEM_INIT
      call INPUT_CNTL
      call INPUT_GRID

      call MAT_CON0
      call MAT_CON1

      call MAT_ASS_MAIN
      call MAT_ASS_BC

      call SOLVE11

      call OUTPUT_UCD

      call PFEM_FINALIZE

      end program heat3Dp
```

MAT_CON0: INLU, IALU生成
MAT_CON1: index, item生成

MAT_CON0：全体構成

```
do icel= 1, ICELTOT
  8節点相互の関係から,
  INL, INU, IAL, IAUを生成
  (FIND_NODE)
enddo
```





行列コネクティビティ生成：MAT_CON0（1/4）

```
!C
!C***
!C*** MAT_CON0
!C***
!C
      subroutine MAT_CON0
      use pfem_util
      implicit REAL*8 (A-H,O-Z)

      NLU= 26

      allocate (INLU(NP), IALU(NP,NLU))

      INLU= 0
      IALU= 0
```

NLU:
各節点における
非ゼロ非対角成分
の最大数
（接続する節点数）

今の問題の場合は
わかっているので，
このようにできる

不明の場合の実装：
⇒レポート課題

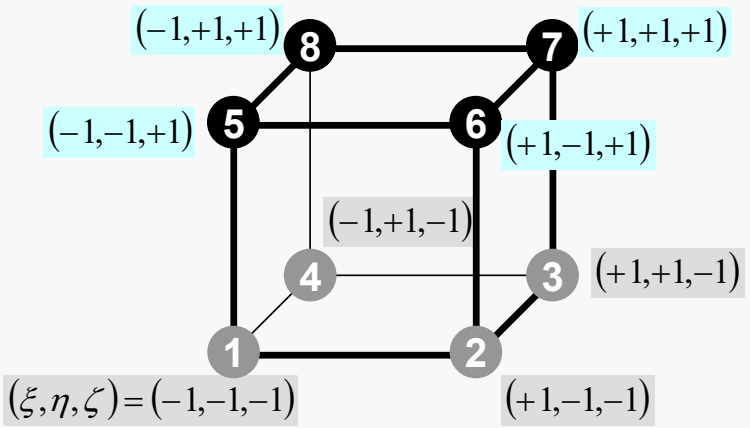
行列コネクティビティ生成：MAT_CON0（1/4）

```
!C
!C***
!C*** MAT_CON0
!C***
!C
      subroutine MAT_CON0
      use pfem_util
      implicit REAL*8 (A-H,O-Z)

      NLU= 26

      allocate (INLU(NP), IALU(NP,NLU))

      INLU= 0
      IALU= 0
```

変数名	サイズ	内　容
INLU	(NP)	各節点の非零非対角成分数
IALU	(NP,NLU)	各節点の非零非対角成分（列番号）

行列コネクティビティ生成：MAT_CON0（2/4）

```
do icel= 1, ICELTOT
  in1= ICELNOD(icel,1)
  in2= ICELNOD(icel,2)
  in3= ICELNOD(icel,3)
  in4= ICELNOD(icel,4)
  in5= ICELNOD(icel,5)
  in6= ICELNOD(icel,6)
  in7= ICELNOD(icel,7)
  in8= ICELNOD(icel,8)

  call FIND_TS_NODE (in1,in2)
  call FIND_TS_NODE (in1,in3)
  call FIND_TS_NODE (in1,in4)
  call FIND_TS_NODE (in1,in5)
  call FIND_TS_NODE (in1,in6)
  call FIND_TS_NODE (in1,in7)
  call FIND_TS_NODE (in1,in8)

  call FIND_TS_NODE (in2,in1)
  call FIND_TS_NODE (in2,in3)
  call FIND_TS_NODE (in2,in4)
  call FIND_TS_NODE (in2,in5)
  call FIND_TS_NODE (in2,in6)
  call FIND_TS_NODE (in2,in7)
  call FIND_TS_NODE (in2,in8)

  call FIND_TS_NODE (in3,in1)
  call FIND_TS_NODE (in3,in2)
  call FIND_TS_NODE (in3,in4)
  call FIND_TS_NODE (in3,in5)
  call FIND_TS_NODE (in3,in6)
  call FIND_TS_NODE (in3,in7)
  call FIND_TS_NODE (in3,in8)
```



節点探索：FIND_TS_NODE

INL,INU,IAL,IAU探索：一次元ではこの部分は手動

```
!C
!C***
!C*** FIND_TS_NODE
!C***
!C
        subroutine FIND_TS_NODE (ip1,ip2)

          do kk= 1, INLU(ip1)
            if (ip2.eq.IALU(ip1,kk)) return
          enddo

          icou= INLU(ip1) + 1
          IALU(ip1,icou)= ip2
          INLU(ip1     )= icou

          return

        end subroutine FIND_TS_NODE
```

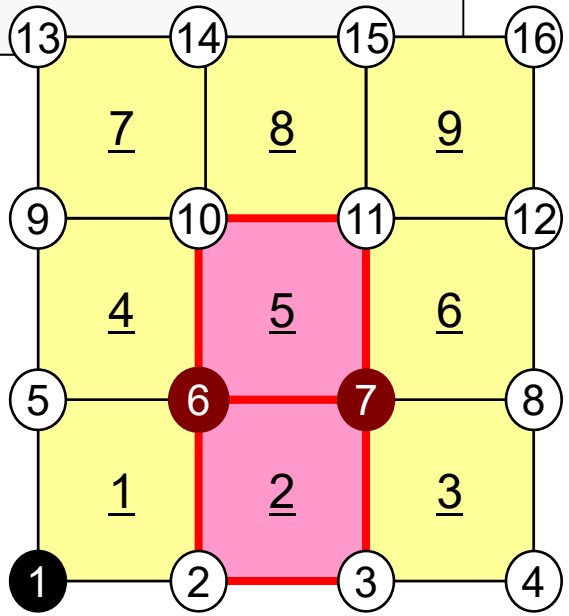
変数名	サイズ	内　容
INLU	(NP)	各節点の非零非対角成分数
IALU	(NP,NLU)	各節点の非零非対角成分（列番号）

節点探索：FIND_TS_NODE

一次元ではこの部分は手動

```
!C
!C***
!C*** FIND_TS_NODE
!C***
!C
        subroutine FIND_TS_NODE (ip1,ip2)

          do kk= 1, INLU(ip1)
            if (ip2.eq.IALU(ip1,kk)) return
          enddo

          icou= INLU(ip1) + 1
          IALU(ip1,icou)= ip2
          INLU(ip1     )= icou

          return

        end subroutine FIND_TS_NODE
```

既にIALUに含まれている場合は，次のペアへ

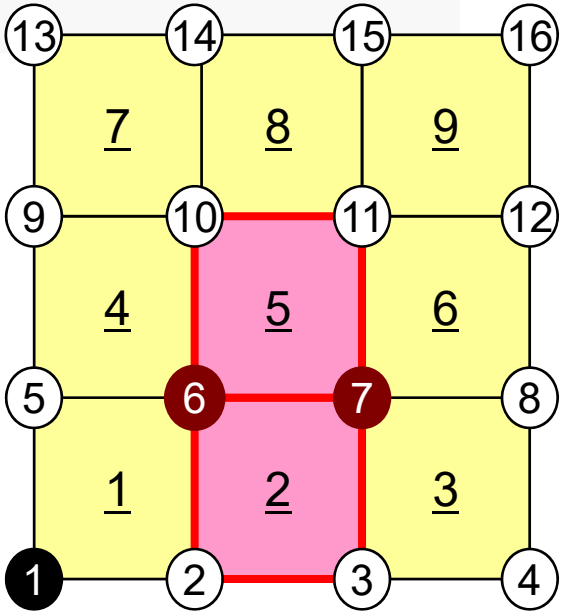


節点探索：FIND_TS_NODE

一次元ではこの部分は手動

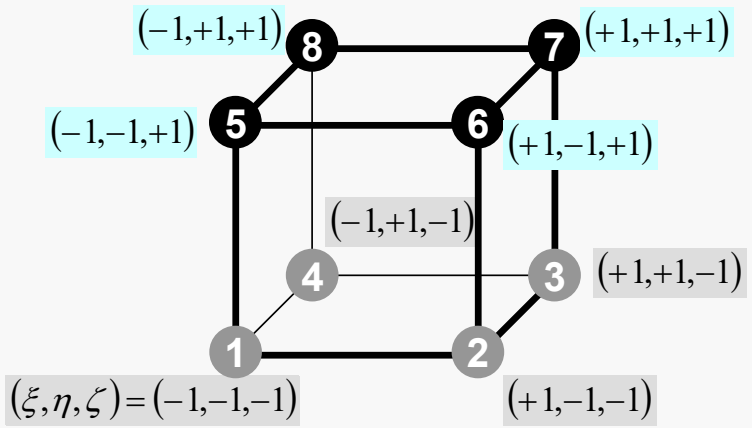
```
!C
!C***
!C*** FIND_TS_NODE
!C***
!C
        subroutine FIND_TS_NODE (ip1,ip2)

          do kk= 1, INLU(ip1)
            if (ip2.eq.IALU(ip1,kk)) return
          enddo

          icou= INLU(ip1) + 1
          IALU(ip1,icou)= ip2
          INLU(ip1     )= icou

          return

        end subroutine FIND_TS_NODE
```

IALUに含まれていない場合は，INLUに1を加えてIALUに格納



行列コネクティビティ生成：MAT_CON0（3/4）

```
      call FIND_TS_NODE (in4,in1)
      call FIND_TS_NODE (in4,in2)
      call FIND_TS_NODE (in4,in3)
      call FIND_TS_NODE (in4,in5)
      call FIND_TS_NODE (in4,in6)
      call FIND_TS_NODE (in4,in7)
      call FIND_TS_NODE (in4,in8)

      call FIND_TS_NODE (in5,in1)
      call FIND_TS_NODE (in5,in2)
      call FIND_TS_NODE (in5,in3)
      call FIND_TS_NODE (in5,in4)
      call FIND_TS_NODE (in5,in6)
      call FIND_TS_NODE (in5,in7)
      call FIND_TS_NODE (in5,in8)

      call FIND_TS_NODE (in6,in1)
      call FIND_TS_NODE (in6,in2)
      call FIND_TS_NODE (in6,in3)
      call FIND_TS_NODE (in6,in4)
      call FIND_TS_NODE (in6,in5)
      call FIND_TS_NODE (in6,in7)
      call FIND_TS_NODE (in6,in8)

      call FIND_TS_NODE (in7,in1)
      call FIND_TS_NODE (in7,in2)
      call FIND_TS_NODE (in7,in3)
      call FIND_TS_NODE (in7,in4)
      call FIND_TS_NODE (in7,in5)
      call FIND_TS_NODE (in7,in6)
      call FIND_TS_NODE (in7,in8)
```



行列コネクティビティ生成：MAT_CON0（4/4）

```
      call FIND_TS_NODE (in8,in1)
      call FIND_TS_NODE (in8,in2)
      call FIND_TS_NODE (in8,in3)
      call FIND_TS_NODE (in8,in4)
      call FIND_TS_NODE (in8,in5)
      call FIND_TS_NODE (in8,in6)
      call FIND_TS_NODE (in8,in7)
    enddo

    do in= 1, N
      NN= INLU(in)
      do k= 1, NN
        NCOL1(k)= IALU(in,k)
      enddo
      call mSORT (NCOL1, NCOL2, NN)
      do k= NN, 1, -1
        IALU(in,NN-k+1)= NCOL1(NCOL2(k))
      enddo
    enddo
```

各節点において，IALU[i][k]が
小さい番号から大きい番号に
並ぶようにソート（単純なバブルソート）
せいぜい100程度のものをソートする

CRS形式への変換：MAT_CON1

```
!C
!C***
!C*** MAT_CON1
!C***
!C
      subroutine MAT_CON1
      use pfem_util
      implicit REAL*8 (A-H,O-Z)

      allocate (index(0:NP))
      index= 0

      do i= 1, NP
        index(i)= index(i-1) + INLU(i)
      enddo

      NPLU= index(NP)

      allocate (item(NPLU))

      do i= 1, NP
        do k= 1, INLU(i)
              kk = k + index(i-1)
          item(kk)=      IALU(i,k)
        enddo
      enddo

      deallocate (INLU, IALU)

      end subroutine MAT_CON1
```

C

$$\mathrm{index}[i+1] = \sum_{k=0}^{i} \mathrm{INLU}[k]$$

$$\mathrm{index}[0] = 0$$

FORTRAN

$$\mathrm{index}(i) = \sum_{k=1}^{i} \mathrm{INLU}(k)$$

$$\mathrm{index}(0) = 0$$

CRS形式への変換：MAT_CON1

```
!C
!C***
!C*** MAT_CON1
!C***
!C
      subroutine MAT_CON1
      use pfem_util
      implicit REAL*8 (A-H,O-Z)

      allocate (index(0:NP))
      index= 0

      do i= 1, NP
        index(i)= index(i-1) + INLU(i)
      enddo

      NPLU= index(NP)

      allocate (item(NPLU))

      do i= 1, NP
        do k= 1, INLU(i)
              kk = k + index(i-1)
          item(kk)=      IALU(i,k)
        enddo
      enddo

      deallocate (INLU, IALU)

      end subroutine MAT_CON1
```

NPLU=index(NP)
itemのサイズ
非ゼロ非対角成分総数

CRS形式への変換：MAT_CON1

```
!C
!C***
!C*** MAT_CON1
!C***
!C
      subroutine MAT_CON1
      use pfem_util
      implicit REAL*8 (A-H,O-Z)

      allocate (index(0:NP))
      index= 0

      do i= 1, NP
        index(i)= index(i-1) + INLU(i)
      enddo

      NPLU= index(NP)

      allocate (item(NPLU))

      do i= 1, NP
        do k= 1, INLU(i)
              kk = k + index(i-1)
          item(kk)=      IALU(i,k)
        enddo
      enddo

      deallocate (INLU, IALU)

      end subroutine MAT_CON1
```

itemに1から始まる節点番号を記憶

CRS形式への変換：MAT_CON1

```
!C
!C***
!C*** MAT_CON1
!C***
!C
      subroutine MAT_CON1
      use pfem_util
      implicit REAL*8 (A-H,O-Z)

      allocate (index(0:NP))
      index= 0

      do i= 1, NP
        index(i)= index(i-1) + INLU(i)
      enddo

      NPLU= index(NP)

      allocate (item(NPLU))

      do i= 1, NP
        do k= 1, INLU(i)
              kk = k + index(i-1)
          item(kk)=      IALU(i,k)
        enddo
      enddo

      deallocate (INLU, IALU)

      end subroutine MAT_CON1
```

これらはもはや不要

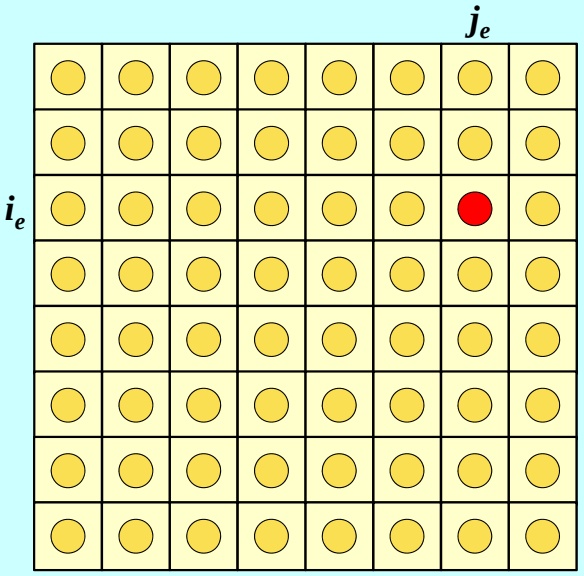
全体処理

```
      program heat3Dp

      use solver11
      use pfem_util

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

      call PFEM_INIT
      call INPUT_CNTL
      call INPUT_GRID

      call MAT_CON0
      call MAT_CON1

      call MAT_ASS_MAIN
      call MAT_ASS_BC

      call SOLVE11

      call OUTPUT_UCD

      call PFEM_FINALIZE

      end program heat3Dp
```

MAT_ASS_MAIN：全体構成

```
do kpn= 1, 2          ガウス積分点番号（ζ方向）
  do jpn= 1, 2        ガウス積分点番号（η方向）
    do ipn= 1, 2      ガウス積分点番号（ξ方向）
      ガウス積分点（8個）における形状関数，
      およびその「自然座標系」における微分の算出
    enddo
  enddo
enddo
```

```
do icel= 1, ICELTOT    要素ループ
  8節点の座標から，ガウス積分点における，形状関数の「全体座標系」における微分，
  およびヤコビアンを算出（JACOBI）

  do ie= 1, 8          局所節点番号
    do je= 1, 8        局所節点番号
      全体節点番号：ip, jp
      A_ip,jp のitemLUにおけるアドレス：kk

      do kpn= 1, 2       ガウス積分点番号（ζ方向）
        do jpn= 1, 2     ガウス積分点番号（η方向）
          do ipn= 1, 2   ガウス積分点番号（ξ方向）
            要素積分⇒要素行列成分計算，全体行列への足しこみ
          enddo
        enddo
      enddo
    enddo
  enddo
enddo
```



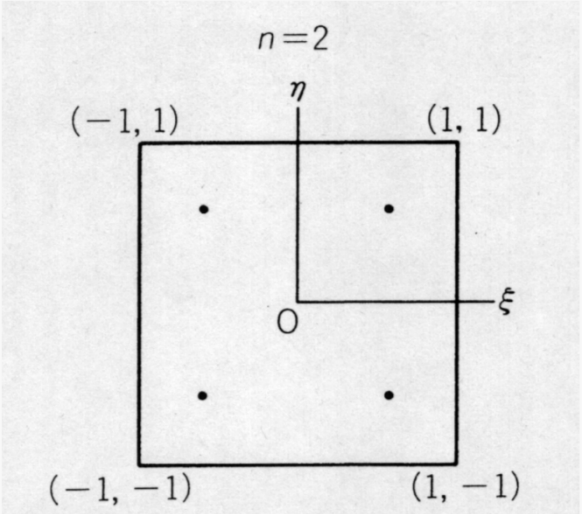
係数行列：MAT_ASS_MAIN（1/6）

```
!C
!C***
!C*** MAT_ASS_MAIN
!C***
!C
      subroutine MAT_ASS_MAIN
      use pfem_util
      implicit REAL*8 (A-H,O-Z)
      integer(kind=kint), dimension(  8) :: nodLOCAL

      allocate (AMAT(NPLU))
      allocate (B(NP), D(NP), X(NP))

      AMAT= 0.d0                        係数行列（非零非対角成分）
         B= 0.d0                        右辺ベクトル
         X= 0.d0                        未知数ベクトル
         D= 0.d0                        係数行列（対角成分）

      WEI(1)= +1.0000000000D+00
      WEI(2)= +1.0000000000D+00

      POS(1)= -0.5773502692D+00
      POS(2)= +0.5773502692D+00
```

係数行列：MAT_ASS_MAIN（1/6）

```
!C
!C***
!C*** MAT_ASS_MAIN
!C***
!C
      subroutine MAT_ASS_MAIN
      use pfem_util
      implicit REAL*8 (A-H,O-Z)
      integer(kind=kint), dimension(  8) :: nodLOCAL

      allocate (AMAT(NPLU))
      allocate (B(NP), D(NP), X(NP))

      AMAT= 0.d0                    係数行列（非零非対角成分）
         B= 0.d0                    右辺ベクトル
         X= 0.d0                    未知数ベクトル
         D= 0.d0                    係数行列（対角成分）

      WEI(1)= +1.0000000000D+00
      WEI(2)= +1.0000000000D+00

      POS(1)= -0.5773502692D+00
      POS(2)= +0.5773502692D+00
```

POS: 積分点座標
WEI: 重み係数



係数行列：MAT_ASS_MAIN（2/6）

```
!C
!C-- INIT.
!C     PNQ   - 1st-order derivative of shape function by QSI
!C     PNE   - 1st-order derivative of shape function by ETA
!C     PNT   - 1st-order derivative of shape function by ZET
!C

      do kp= 1, 2
      do jp= 1, 2
      do ip= 1, 2

        QP1= 1.d0 + POS(ip)
        QM1= 1.d0 - POS(ip)
        EP1= 1.d0 + POS(jp)
        EM1= 1.d0 - POS(jp)
        TP1= 1.d0 + POS(kp)
        TM1= 1.d0 - POS(kp)

        SHAPE(ip,jp,kp,1)= 0.8d0 * QM1 * EM1 * TM1
        SHAPE(ip,jp,kp,2)= 0.8d0 * QP1 * EM1 * TM1
        SHAPE(ip,jp,kp,3)= 0.8d0 * QP1 * EP1 * TM1
        SHAPE(ip,jp,kp,4)= 0.8d0 * QM1 * EP1 * TM1
        SHAPE(ip,jp,kp,5)= 0.8d0 * QM1 * EM1 * TP1
        SHAPE(ip,jp,kp,6)= 0.8d0 * QP1 * EM1 * TP1
        SHAPE(ip,jp,kp,7)= 0.8d0 * QP1 * EP1 * TP1
        SHAPE(ip,jp,kp,8)= 0.8d0 * QP1 * EP1 * TP1
```

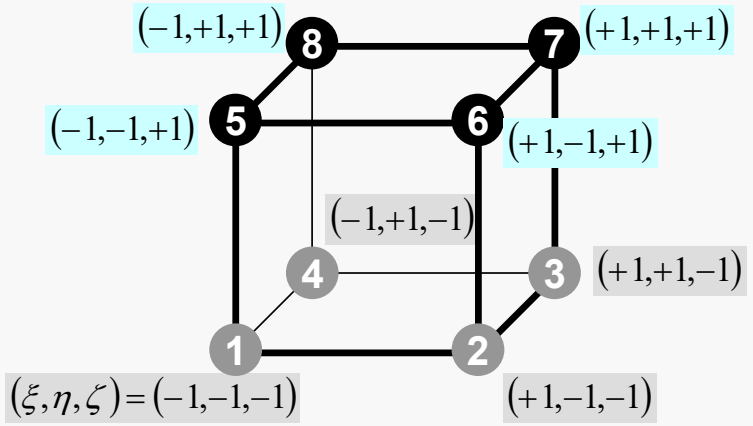
係数行列：MAT_ASS_MAIN（2/6）

```
!C
!C-- INIT.
!C     PNQ   - 1st-order derivative of shape function by QSI
!C     PNE   - 1st-order derivative of shape function by ETA
!C     PNT   - 1st-order derivative of shape function by ZET
!C

      do kp= 1, 2
      do jp= 1, 2
      do ip= 1, 2

        QP1= 1.d0 + POS(ip)
        QM1= 1.d0 - POS(ip)
        EP1= 1.d0 + POS(jp)
        EM1= 1.d0 - POS(jp)
        TP1= 1.d0 + POS(kp)
        TM1= 1.d0 - POS(kp)

        SHAPE(ip,jp,kp,1)= 08th * QM1 * EM1 * TM1
        SHAPE(ip,jp,kp,2)= 08th * QP1 * EM1 * TM1
        SHAPE(ip,jp,kp,3)= 08th * QP1 * EP1 * TM1
        SHAPE(ip,jp,kp,4)= 08th * QM1 * EP1 * TM1
        SHAPE(ip,jp,kp,5)= 08th * QM1 * EM1 * TP1
        SHAPE(ip,jp,kp,6)= 08th * QP1 * EM1 * TP1
        SHAPE(ip,jp,kp,7)= 08th * QP1 * EP1 * TP1
        SHAPE(ip,jp,kp,8)= 08th * QP1 * EP1 * TP1
```

$$\mathrm{QP1(i)} = (1+\xi_i), \quad \mathrm{QM1(i)} = (1-\xi_i)$$
$$\mathrm{EP1(j)} = (1+\eta_j), \quad \mathrm{EM1(j)} = (1-\eta_i)$$
$$\mathrm{TP1(k)} = (1+\zeta_k), \quad \mathrm{TM1(k)} = (1-\zeta_k)$$

係数行列：MAT_ASS_MAIN（2/6）

```
!C
!C-- INIT.
!C     PNQ   - 1st-order derivative of shape function by QSI
!C     PNE   - 1st-order derivative of shape function by ETA
!C     PNT   - 1st-order derivative of shape function by ZET
!C

      do kp= 1, 2
      do jp= 1, 2
      do ip= 1, 2

        QP1= 1.d0 + POS(ip)
        QM1= 1.d0 - POS(ip)
        EP1= 1.d0 + POS(jp)
        EM1= 1.d0 - POS(jp)
        TP1= 1.d0 + POS(kp)
        TM1= 1.d0 - POS(kp)

        SHAPE(ip,jp,kp,1)= 08th * QM1 * EM1 * TM1
        SHAPE(ip,jp,kp,2)= 08th * QP1 * EM1 * TM1
        SHAPE(ip,jp,kp,3)= 08th * QP1 * EP1 * TM1
        SHAPE(ip,jp,kp,4)= 08th * QM1 * EP1 * TM1
        SHAPE(ip,jp,kp,5)= 08th * QM1 * EM1 * TP1
        SHAPE(ip,jp,kp,6)= 08th * QP1 * EM1 * TP1
        SHAPE(ip,jp,kp,7)= 08th * QP1 * EP1 * TP1
        SHAPE(ip,jp,kp,8)= 08th * QP1 * EP1 * TP1
```



係数行列：MAT_ASS_MAIN（2/6）

```
!C
!C-- INIT.
!C     PNQ   - 1st-order derivative of shape function by QSI
!C     PNE   - 1st-order derivative of shape function by ETA
!C     PNT   - 1st-order derivative of shape function by ZET
!C

      do kp= 1, 2
      do jp= 1, 2
      do ip= 1, 2

        QP1= 1.d0 + POS(ip)
        QM1= 1.d0 - POS(ip)
        EP1= 1.d0 + POS(jp)
        EM1= 1.d0 - POS(jp)
        TP1= 1.d0 + POS(kp)
        TM1= 1.d0 - POS(kp)

        SHAPE(ip,jp,kp,1)= 08th * QM1 * EM1 * TM1
        SHAPE(ip,jp,kp,2)= 08th * QP1 * EM1 * TM1
        SHAPE(ip,jp,kp,3)= 08th * QP1 * EP1 * TM1
        SHAPE(ip,jp,kp,4)= 08th * QM1 * EP1 * TM1
        SHAPE(ip,jp,kp,5)= 08th * QM1 * EM1 * TP1
        SHAPE(ip,jp,kp,6)= 08th * QP1 * EM1 * TP1
        SHAPE(ip,jp,kp,7)= 08th * QP1 * EP1 * TP1
        SHAPE(ip,jp,kp,8)= 08th * QP1 * EP1 * TP1
```

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

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

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

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

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

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

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

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

係数行列：MAT_ASS_MAIN（3/6）

```
      PNQ(jp,kp,1)= - 08th * EM1 * TM1
      PNQ(jp,kp,2)= + 08th * EM1 * TM1
      PNQ(jp,kp,3)= + 08th * EP1 * TM1
      PNQ(jp,kp,4)= - 08th * EP1 * TM1
      PNQ(jp,kp,5)= - 08th * EM1 * TP1
      PNQ(jp,kp,6)= + 08th * EM1 * TP1
      PNQ(jp,kp,7)= + 08th * EP1 * TP1
      PNQ(jp,kp,8)= - 08th * EP1 * TP1
      PNE(ip,kp,1)= - 08th * QM1 * TM1
      PNE(ip,kp,2)= - 08th * QP1 * TM1
      PNE(ip,kp,3)= + 08th * QP1 * TM1
      PNE(ip,kp,4)= + 08th * QM1 * TM1
      PNE(ip,kp,5)= - 08th * QM1 * TP1
      PNE(ip,kp,6)= - 08th * QP1 * TP1
      PNE(ip,kp,7)= + 08th * QP1 * TP1
      PNE(ip,kp,8)= + 08th * QM1 * TP1
      PNT(ip,jp,1)= - 08th * QM1 * EM1
      PNT(ip,jp,2)= - 08th * QP1 * EM1
      PNT(ip,jp,3)= - 08th * QP1 * EP1
      PNT(ip,jp,4)= - 08th * QM1 * EP1
      PNT(ip,jp,5)= + 08th * QM1 * EM1
      PNT(ip,jp,6)= + 08th * QP1 * EM1
      PNT(ip,jp,7)= + 08th * QP1 * EP1
      PNT(ip,jp,8)= + 08th * QM1 * EP1
    enddo
    enddo
    enddo

    do icel= 1, ICELTOT
      CONDO= COND

      in1= ICELNOD(icel,1)
      in2= ICELNOD(icel,2)
      in3= ICELNOD(icel,3)
      in4= ICELNOD(icel,4)
      in5= ICELNOD(icel,5)
      in6= ICELNOD(icel,6)
      in7= ICELNOD(icel,7)
      in8= ICELNOD(icel,8)
```

$$PNQ(j,k)=\frac{\partial N_l}{\partial \xi}(\xi=\xi_i,\eta=\eta_j,\zeta=\zeta_k)$$

$$PNE(i,k)=\frac{\partial N_l}{\partial \eta}(\xi=\xi_i,\eta=\eta_j,\zeta=\zeta_k)$$

$$PNT(i,j)=\frac{\partial N_l}{\partial \zeta}(\xi=\xi_i,\eta=\eta_j,\zeta=\zeta_k)$$

$$\frac{\partial N_1}{\partial \xi}(\xi_i,\eta_j,\zeta_k)=-\frac{1}{8}(1-\eta_j)(1-\zeta_k)$$

$$\frac{\partial N_2}{\partial \xi}(\xi_i,\eta_j,\zeta_k)=+\frac{1}{8}(1-\eta_j)(1-\zeta_k)$$

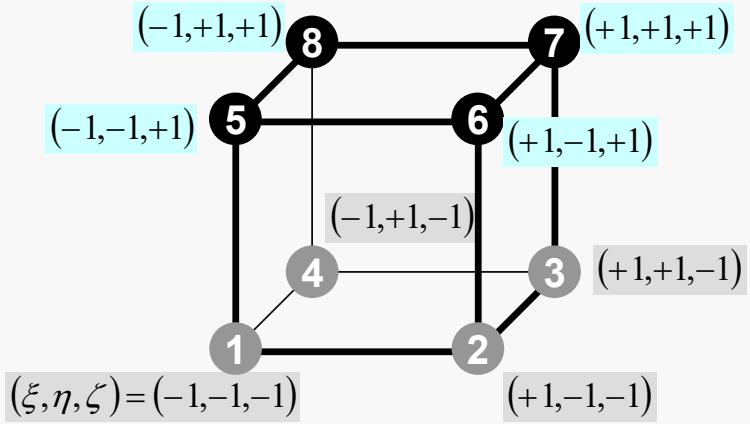
$$\frac{\partial N_3}{\partial \xi}(\xi_i,\eta_j,\zeta_k)=+\frac{1}{8}(1+\eta_j)(1-\zeta_k)$$

$$\frac{\partial N_3}{\partial \xi}(\xi_i,\eta_j,\zeta_k)=-\frac{1}{8}(1+\eta_j)(1-\zeta_k)$$

(ξ_i,η_j,ζ_k) における形状関数の一階微分

係数行列：MAT_ASS_MAIN（3/6）

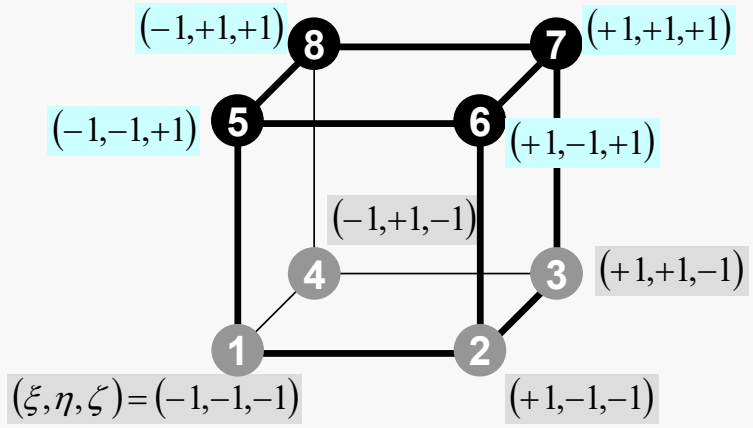
```
      PNQ(jp,kp,1)= - 08th * EM1 * TM1
      PNQ(jp,kp,2)= + 08th * EM1 * TM1
      PNQ(jp,kp,3)= + 08th * EP1 * TM1
      PNQ(jp,kp,4)= - 08th * EP1 * TM1
      PNQ(jp,kp,5)= - 08th * EM1 * TP1
      PNQ(jp,kp,6)= + 08th * EM1 * TP1
      PNQ(jp,kp,7)= + 08th * EP1 * TP1
      PNQ(jp,kp,8)= - 08th * EP1 * TP1
      PNE(ip,kp,1)= - 08th * QM1 * TM1
      PNE(ip,kp,2)= - 08th * QP1 * TM1
      PNE(ip,kp,3)= + 08th * QP1 * TM1
      PNE(ip,kp,4)= + 08th * QM1 * TM1
      PNE(ip,kp,5)= - 08th * QM1 * TP1
      PNE(ip,kp,6)= - 08th * QP1 * TP1
      PNE(ip,kp,7)= + 08th * QP1 * TP1
      PNE(ip,kp,8)= + 08th * QM1 * TP1
      PNT(ip,jp,1)= - 08th * QM1 * EM1
      PNT(ip,jp,2)= - 08th * QP1 * EM1
      PNT(ip,jp,3)= - 08th * QP1 * EP1
      PNT(ip,jp,4)= - 08th * QM1 * EP1
      PNT(ip,jp,5)= + 08th * QM1 * EM1
      PNT(ip,jp,6)= + 08th * QP1 * EM1
      PNT(ip,jp,7)= + 08th * QP1 * EP1
      PNT(ip,jp,8)= + 08th * QM1 * EP1
    enddo
    enddo
    enddo

    do icel= 1, ICELTOT
      CONDO= COND

      in1= ICELNOD(icel,1)
      in2= ICELNOD(icel,2)
      in3= ICELNOD(icel,3)
      in4= ICELNOD(icel,4)
      in5= ICELNOD(icel,5)
      in6= ICELNOD(icel,6)
      in7= ICELNOD(icel,7)
      in8= ICELNOD(icel,8)
```



係数行列：MAT_ASS_MAIN（4/6）

```
      nodLOCAL(1)= in1
      nodLOCAL(2)= in2
      nodLOCAL(3)= in3
      nodLOCAL(4)= in4
      nodLOCAL(5)= in5
      nodLOCAL(6)= in6
      nodLOCAL(7)= in7
      nodLOCAL(8)= in8

      X1= XYZ(in1,1)
      X2= XYZ(in2,1)
      X3= XYZ(in3,1)
      X4= XYZ(in4,1)
      X5= XYZ(in5,1)
      X6= XYZ(in6,1)
      X7= XYZ(in7,1)
      X8= XYZ(in8,1)
      Y1= XYZ(in1,2)
      Y2= XYZ(in2,2)
      Y3= XYZ(in3,2)
      Y4= XYZ(in4,2)
      Y5= XYZ(in5,2)
      Y6= XYZ(in6,2)
      Y7= XYZ(in7,2)
      Y8= XYZ(in8,2)
      QVC= O8th * (X1+X2+X3+X4+X5+X6+X7+X8+
     &             Y1+Y2+Y3+Y4+Y5+Y6+Y7+Y8)
      Z1= XYZ(in1,3)
      Z2= XYZ(in2,3)
      Z3= XYZ(in3,3)
      Z4= XYZ(in4,3)
      Z5= XYZ(in5,3)
      Z6= XYZ(in6,3)
      Z7= XYZ(in7,3)
      Z8= XYZ(in8,3)

      call JACOBI (DETJ, PNQ, PNE, PNT, PNX, PNY, PNZ,              &
     &             X1, X2, X3, X4, X5, X6, X7, X8,                  &
     &             Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8,                  &
     &             Z1, Z2, Z3, Z4, Z5, Z6, Z7, Z8 )
```

8節点の節点番号



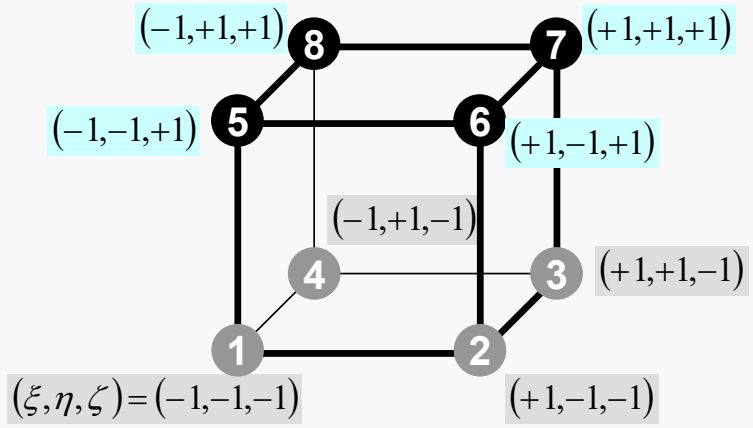
係数行列：MAT_ASS_MAIN（4/6）

```
      nodLOCAL(1)= in1
      nodLOCAL(2)= in2
      nodLOCAL(3)= in3
      nodLOCAL(4)= in4
      nodLOCAL(5)= in5
      nodLOCAL(6)= in6
      nodLOCAL(7)= in7
      nodLOCAL(8)= in8

      X1= XYZ(in1,1)
      X2= XYZ(in2,1)
      X3= XYZ(in3,1)
      X4= XYZ(in4,1)
      X5= XYZ(in5,1)
      X6= XYZ(in6,1)
      X7= XYZ(in7,1)
      X8= XYZ(in8,1)
      Y1= XYZ(in1,2)
      Y2= XYZ(in2,2)
      Y3= XYZ(in3,2)
      Y4= XYZ(in4,2)
      Y5= XYZ(in5,2)
      Y6= XYZ(in6,2)
      Y7= XYZ(in7,2)
      Y8= XYZ(in8,2)
      QVC= O8th * (X1+X2+X3+X4+X5+X6+X7+X8+
     &             Y1+Y2+Y3+Y4+Y5+Y6+Y7+Y8)
      Z1= XYZ(in1,3)
      Z2= XYZ(in2,3)
      Z3= XYZ(in3,3)
      Z4= XYZ(in4,3)
      Z5= XYZ(in5,3)
      Z6= XYZ(in6,3)
      Z7= XYZ(in7,3)
      Z8= XYZ(in8,3)

      call JACOBI (DETJ, PNQ, PNE, PNT, PNX, PNY, PNZ,              &
     &             X1, X2, X3, X4, X5, X6, X7, X8,                  &
     &             Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8,                  &
     &             Z1, Z2, Z3, Z4, Z5, Z6, Z7, Z8 )
```

8節点のX座標

8節点のY座標

8節点のZ座標

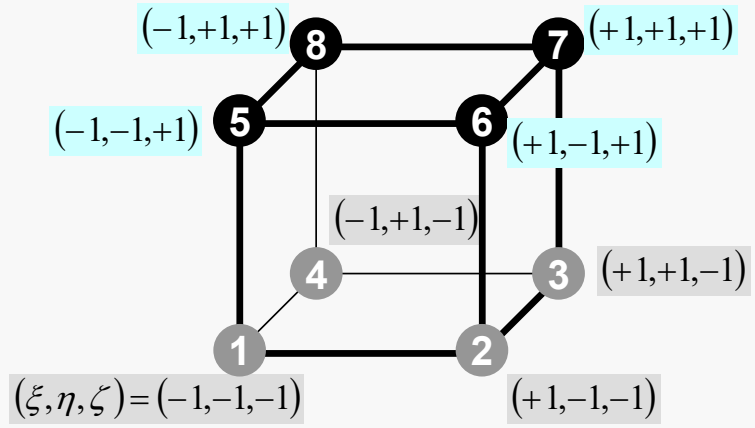


係数行列：MAT_ASS_MAIN（4/6）

```
      nodLOCAL(1)= in1
      nodLOCAL(2)= in2
      nodLOCAL(3)= in3
      nodLOCAL(4)= in4
      nodLOCAL(5)= in5
      nodLOCAL(6)= in6
      nodLOCAL(7)= in7
      nodLOCAL(8)= in8

      X1= XYZ(in1,1)
      X2= XYZ(in2,1)
      X3= XYZ(in3,1)
      X4= XYZ(in4,1)
      X5= XYZ(in5,1)
      X6= XYZ(in6,1)
      X7= XYZ(in7,1)
      X8= XYZ(in8,1)
      Y1= XYZ(in1,2)
      Y2= XYZ(in2,2)
      Y3= XYZ(in3,2)
      Y4= XYZ(in4,2)
      Y5= XYZ(in5,2)
      Y6= XYZ(in6,2)
      Y7= XYZ(in7,2)
      Y8= XYZ(in8,2)
      QVC= O8th * (X1+X2+X3+X4+X5+X6+X7+X8+
     &             Y1+Y2+Y3+Y4+Y5+Y6+Y7+Y8)
      Z1= XYZ(in1,3)
      Z2= XYZ(in2,3)
      Z3= XYZ(in3,3)
      Z4= XYZ(in4,3)
      Z5= XYZ(in5,3)
      Z6= XYZ(in6,3)
      Z7= XYZ(in7,3)
      Z8= XYZ(in8,3)

      call JACOBI (DETJ, PNQ, PNE, PNT, PNX, PNY, PN
     &             X1, X2, X3, X4, X5, X6, X7, X8,
     &             Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8,
     &             Z1, Z2, Z3, Z4, Z5, Z6, Z7, Z8 )
```

8節点のX座標

8節点のY座標



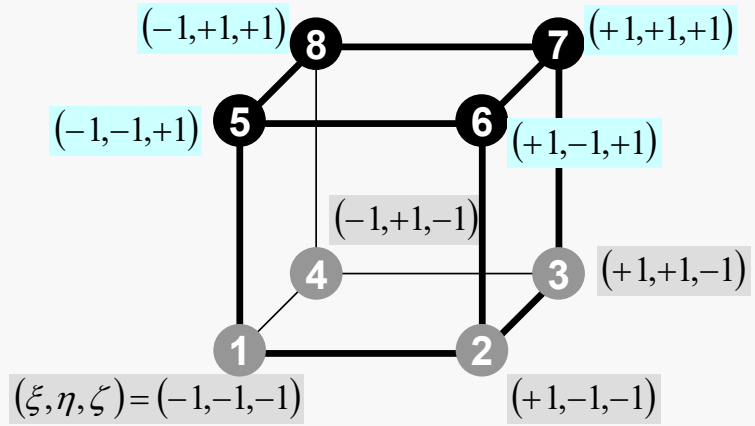
$$\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(\lambda\frac{\partial T}{\partial z}\right)+\dot{Q}(x,y,z)=0$$

$$\dot{Q}(x,y,z)=QVOL\left|x_C+y_C\right|$$

体積当たり発熱量は位置（メッシュの中心の座標 x_c,y_c）に依存

係数行列：MAT_ASS_MAIN（4/6）

```
      nodLOCAL(1)= in1
      nodLOCAL(2)= in2
      nodLOCAL(3)= in3
      nodLOCAL(4)= in4
      nodLOCAL(5)= in5
      nodLOCAL(6)= in6
      nodLOCAL(7)= in7
      nodLOCAL(8)= in8

      X1= XYZ(in1,1)
      X2= XYZ(in2,1)
      X3= XYZ(in3,1)
      X4= XYZ(in4,1)
      X5= XYZ(in5,1)
      X6= XYZ(in6,1)
      X7= XYZ(in7,1)
      X8= XYZ(in8,1)
      Y1= XYZ(in1,2)
      Y2= XYZ(in2,2)
      Y3= XYZ(in3,2)
      Y4= XYZ(in4,2)
      Y5= XYZ(in5,2)
      Y6= XYZ(in6,2)
      Y7= XYZ(in7,2)
      Y8= XYZ(in8,2)
      QVC= 08th * (X1+X2+X3+X4+X5+X6+X7+X8+
     &             Y1+Y2+Y3+Y4+Y5+Y6+Y7+Y8)
      Z1= XYZ(in1,3)
      Z2= XYZ(in2,3)
      Z3= XYZ(in3,3)
      Z4= XYZ(in4,3)
      Z5= XYZ(in5,3)
      Z6= XYZ(in6,3)
      Z7= XYZ(in7,3)
      Z8= XYZ(in8,3)

      call JACOBI (DETJ, PNQ, PNE, PNT, PNX, PNY, PNZ,
     &             X1, X2, X3, X4, X5, X6, X7, X8,
     &             Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8,
     &             Z1, Z2, Z3, Z4, Z5, Z6, Z7, Z8 )
```



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

$$\dot{Q}(x,y,z)=QVOL\left|x_C+y_C\right|$$

$$QVC=\left|x_C+y_C\right|$$

係数行列：MAT_ASS_MAIN（4/6）

```
      nodLOCAL(1)= in1
      nodLOCAL(2)= in2
      nodLOCAL(3)= in3
      nodLOCAL(4)= in4
      nodLOCAL(5)= in5
      nodLOCAL(6)= in6
      nodLOCAL(7)= in7
      nodLOCAL(8)= in8

      X1= XYZ(in1,1)
      X2= XYZ(in2,1)
      X3= XYZ(in3,1)
      X4= XYZ(in4,1)
      X5= XYZ(in5,1)
      X6= XYZ(in6,1)
      X7= XYZ(in7,1)
      X8= XYZ(in8,1)
      Y1= XYZ(in1,2)
      Y2= XYZ(in2,2)
      Y3= XYZ(in3,2)
      Y4= XYZ(in4,2)
      Y5= XYZ(in5,2)
      Y6= XYZ(in6,2)
      Y7= XYZ(in7,2)
      Y8= XYZ(in8,2)
      QVC= O8th * (X1+X2+X3+X4+X5+X6+X7+X8+
     &             Y1+Y2+Y3+Y4+Y5+Y6+Y7+Y8)
      Z1= XYZ(in1,3)
      Z2= XYZ(in2,3)
      Z3= XYZ(in3,3)
      Z4= XYZ(in4,3)
      Z5= XYZ(in5,3)
      Z6= XYZ(in6,3)
      Z7= XYZ(in7,3)
      Z8= XYZ(in8,3)

      call JACOBI (DETJ, PNQ, PNE, PNT, PNX, PNY, PNZ,              &
     &             X1, X2, X3, X4, X5, X6, X7, X8,                  &
     &             Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8,                  &
     &             Z1, Z2, Z3, Z4, Z5, Z6, Z7, Z8 )
```

係数行列：MAT_ASS_MAIN（5/6）

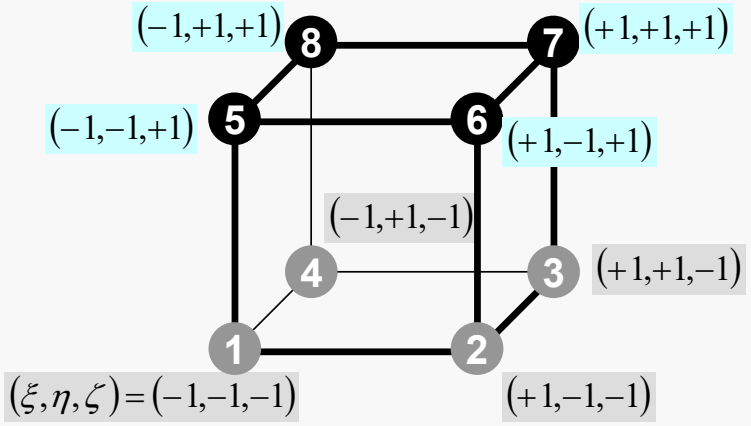
```
!C
!C== CONSTRUCT the GLOBAL MATRIX

        do ie= 1, 8
          ip = nodLOCAL(ie)
        do je= 1, 8
          jp = nodLOCAL(je)

          kk= 0
          if (jp.ne.ip) then
            iiS= index(ip-1) + 1
            iiE= index(ip  )
            do k= iiS, iiE
              if ( item(k).eq.jp ) then
                kk= k
                exit
              endif
            enddo
          endif
```

全体行列の非対角成分

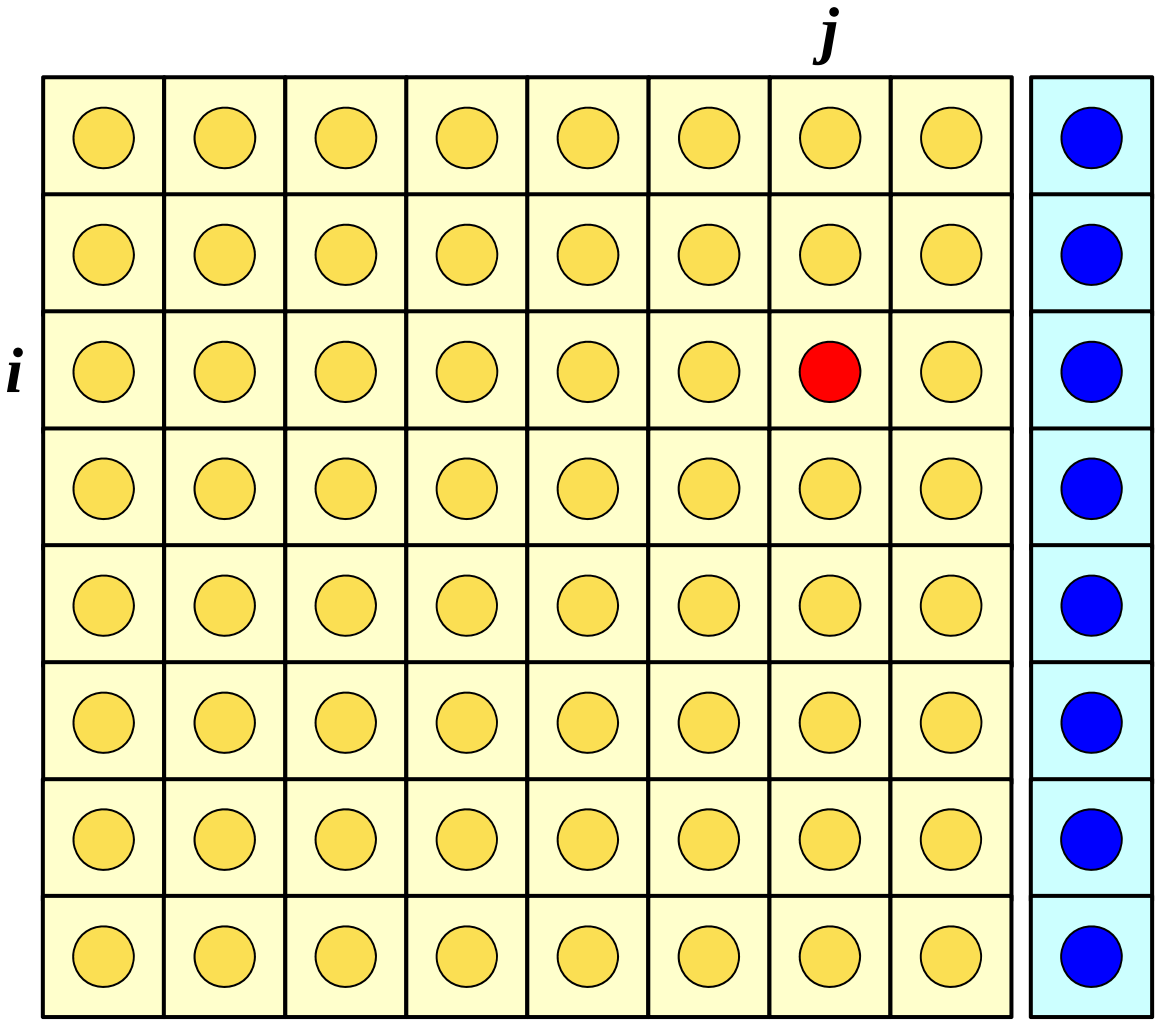
$A_{ip,jp}$

kk：itemにおけるアドレス

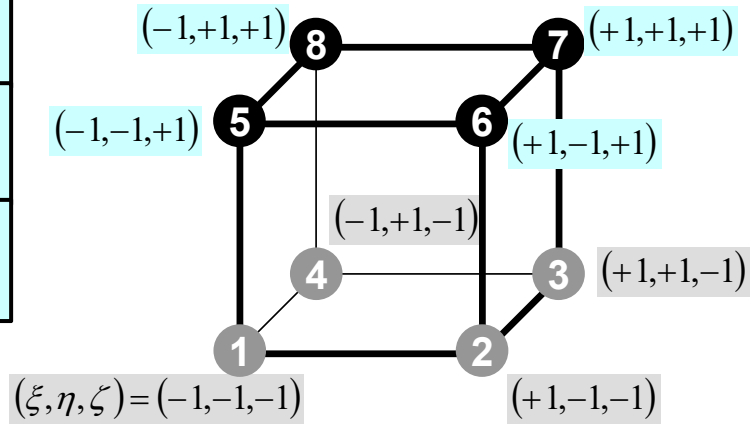
ip= nodLOCAL(ie)
jp= nodLOCAL(je)



要素マトリクス：8×8行列



$$\lfloor k_{ij} \rfloor \quad (i, j = 1 \ldots 8)$$

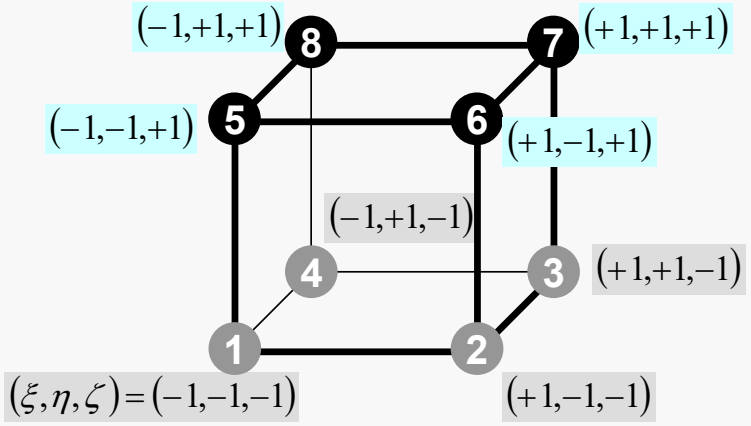


係数行列：MAT_ASS_MAIN（5/6）

```
!C
!C== CONSTRUCT the GLOBAL MATRIX

        do ie= 1, 8
          ip = nodLOCAL(ie)
        do je= 1, 8
          jp = nodLOCAL(je)

          kk= 0
          if (jp.ne.ip) then
            iiS= index(ip-1) + 1
            iiE= index(ip  )
            do k= iiS, iiE
              if ( item(k).eq.jp ) then
                kk= k
                exit
              endif
            enddo
          endif
```

要素マトリクス（i_e~j_e）
全体マトリクス（i_p~j_p）の関係

kk：itemにおけるアドレス



係数行列：MAT_ASS_MAIN（6/6）

```
        QVO   = 0.d0
        COEFij= 0.d0
        do kpn= 1, 2
        do jpn= 1, 2
        do ipn= 1, 2
          coef= dabs(DETJ(ipn,jpn,kpn))*WEI(ipn)*WEI(jpn)*WEI(kpn)

          PNXi= PNX(ipn,jpn,kpn,ie)
          PNYi= PNY(ipn,jpn,kpn,ie)
          PNZi= PNZ(ipn,jpn,kpn,ie)

          PNXj= PNX(ipn,jpn,kpn,je)
          PNYj= PNY(ipn,jpn,kpn,je)
          PNZj= PNZ(ipn,jpn,kpn,je)

          COEFij= COEFij + coef * CONDO *
     &                     (PNXi*PNXj+PNYi*PNYj+PNZi*PNZj)

          SHi= SHAPE(ipn,jpn,kpn,ie)
          QVO= QVO + SHi * QVOL * coef
        enddo
        enddo
        enddo

        if (jp.eq.ip) then
          D(ip)= D(ip) + COEFij
          B(ip)= B(ip) + QVO*QVC
         else
          AMAT(kk)= AMAT(kk) + COEFij
        endif
      enddo
      enddo
    enddo

    return
    end
```

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

係数行列：MAT_ASS_MAIN（6/6）

```
        QV0   = 0.d0
        COEFij= 0.d0
        do kpn= 1, 2
        do jpn= 1, 2
        do ipn= 1, 2
          coef= dabs(DETJ(ipn,jpn,kpn))*WEI(ipn)*WEI(jpn)*WEI(kpn)

          PNXi= PNX(ipn,jpn,kpn,ie)
          PNYi= PNY(ipn,jpn,kpn,ie)
          PNZi= PNZ(ipn,jpn,kpn,ie)

          PNXj= PNX(ipn,jpn,kpn,je)
          PNYj= PNY(ipn,jpn,kpn,je)
          PNZj= PNZ(ipn,jpn,kpn,je)

          COEFij= COEFij + coef * CONDO *
     &                     (PNXi*PNXj+PNYi*PNYj+PNZi*PNZj)

          SHi= SHAPE(ipn,jpn,kpn,ie)
          QV0= QV0 + SHi * QV0L * coef
        enddo
        enddo
        enddo

        if (jp.eq.ip) then
          D(ip)= D(ip) + COEFij
          B(ip)= B(ip) + QV0*QVC
         else
          AMAT(kk)= AMAT(kk) + COEFij
        endif
      enddo
      enddo
    enddo

    return
    end
```

$$I = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} f(\xi,\eta,\zeta)\, d\xi d\eta d\zeta = \sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{k=1}^{N} W_i \cdot W_j \cdot W_k \cdot f(\xi_i,\eta_j,\zeta_k)$$

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

係数行列：MAT_ASS_MAIN（6/6）

```
        QV0   = 0.d0
        COEFij= 0.d0
        do kpn= 1, 2
        do jpn= 1, 2
        do ipn= 1, 2
          coef= dabs(DETJ(ipn,jpn,kpn))*WEI(ipn)*WEI(jpn)*WEI(kpn)

          PNXi= PNX(ipn,jpn,kpn,ie)
          PNYi= PNY(ipn,jpn,kpn,ie)
          PNZi= PNZ(ipn,jpn,kpn,ie)

          PNXj= PNX(ipn,jpn,kpn,je)
          PNYj= PNY(ipn,jpn,kpn,je)
          PNZj= PNZ(ipn,jpn,kpn,je)

          COEFij= COEFij + coef * CONDO *
     &                     (PNXi*PNXj+PNYi*PNYj+PNZi*PNZj)

          SHi= SHAPE(ipn,jpn,kpn,ie)
          QV0= QV0 + SHi * QV0L * coef
        enddo
        enddo
        enddo

        if (jp.eq.ip) then
          D(ip)= D(ip) + COEFij
          B(ip)= B(ip) + QV0*QVC
         else
          AMAT(kk)= AMAT(kk) + COEFij
        endif
      enddo
      enddo
    enddo

    return
    end
```

$$\mathrm{coef} = W_i \cdot W_j \cdot W_k \cdot \det\left|J\left(\xi_i, \eta_j, \zeta_k\right)\right|$$

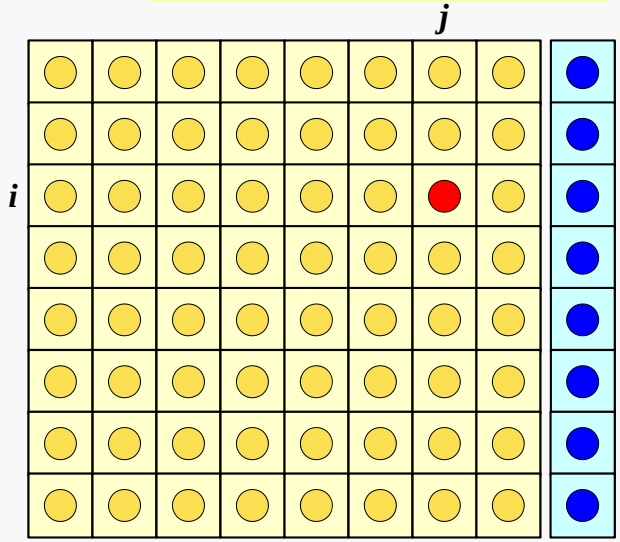
$$I = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} f(\xi,\eta,\zeta)\, d\xi d\eta d\zeta = \sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{k=1}^{N} W_i \cdot W_j \cdot W_k \cdot f(\xi_i,\eta_j,\zeta_k)$$

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

係数行列：MAT_ASS_MAIN（6/6）

```
QVO    = 0.d0
         COEFij= 0.d0
         do kpn= 1, 2
         do jpn= 1, 2
         do ipn= 1, 2
           coef= dabs(DETJ(ipn,jpn,kpn))*WEI(ipn)*WEI(jpn)*WEI(kpn)

           PNXi= PNX(ipn,jpn,kpn,ie)
           PNYi= PNY(ipn,jpn,kpn,ie)
           PNZi= PNZ(ipn,jpn,kpn,ie)

           PNXj= PNX(ipn,jpn,kpn,je)
           PNYj= PNY(ipn,jpn,kpn,je)
           PNZj= PNZ(ipn,jpn,kpn,je)

           COEFij= COEFij + coef * CONDO *
     &                      (PNXi*PNXj+PNYi*PNYj+PNZi*PNZj)

           SHi= SHAPE(ipn,jpn,kpn,ie)
           QVO= QVO + SHi * QVOL * coef
         enddo
         enddo
         enddo

         if (jp.eq.ip) then
           D(ip)= D(ip) + COEFij
           B(ip)= B(ip) + QVO*QVC
          else
           AMAT(kk)= AMAT(kk) + COEFij
         endif
       enddo
       enddo
     enddo

     return
     end
```

$$\left[k_{ij}\right] \quad (i,j=1\ldots8)$$



係数行列：MAT_ASS_MAIN（6/6）

```
        QV0   = 0.d0
        COEFij= 0.d0
        do kpn= 1, 2
        do jpn= 1, 2
        do ipn= 1, 2
          coef= dabs(DETJ(ipn,jpn,kpn))*WEI(ipn)*WEI(jpn)*WEI(kpn)

          PNXi= PNX(ipn,jpn,kpn,ie)
          PNYi= PNY(ipn,jpn,kpn,ie)
          PNZi= PNZ(ipn,jpn,kpn,ie)

          PNXj= PNX(ipn,jpn,kpn,je)
          PNYj= PNY(ipn,jpn,kpn,je)
          PNZj= PNZ(ipn,jpn,kpn,je)

          COEFij= COEFij + coef * CONDO *
     &                     (PNXi*PNXj+PNYi*PNYj+PNZi*PNZj)

          SHi= SHAPE(ipn,jpn,kpn,ie)
          QV0= QV0 + SHi * QVOL * coef
        enddo
        enddo
        enddo

        if (jp.eq.ip) then
          D(ip)= D(ip) + COEFij
          B(ip)= B(ip) + QV0*QVC
         else
          AMAT(kk)= AMAT(kk) + COEFij
        endif
      enddo
      enddo
    enddo

    return
    end
```

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

$$[f]^{(e)} = \int_V \dot{Q}[N]^T dV$$

$$\dot{Q}(x,y,z) = QVOL\left|x_C + y_C\right|$$

$$QVC = \left|x_C + y_C\right|$$

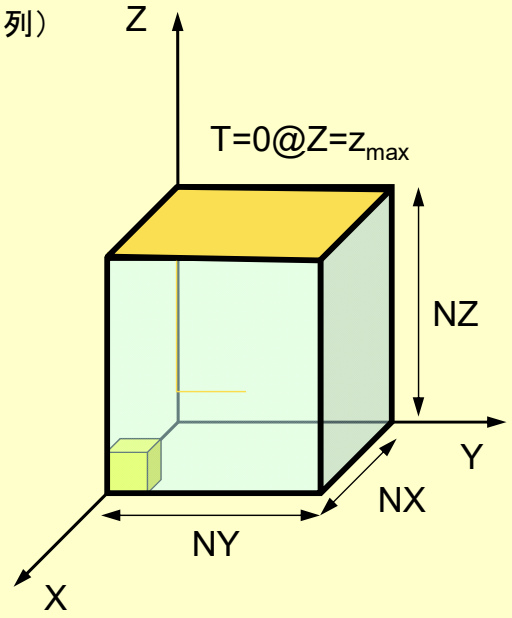
$$QV0 = \int_V QVOL[N]^T dV$$

$$[f]^{(e)} = QV0 \cdot QVC$$

MAT_ASS_BC：全体構成

```
do i= 1, NP    節点ループ
  （ディリクレ）境界条件を設定する節点をマーク（IWKX）
enddo
```

```
do i= 1, NP    節点ループ
  if (IWKX(i,1).eq.1) then   マークされた節点だったら
    対応する右辺ベクトル（B）の成分，対角成分（D）の成分の修正（行・列）
    do k= index(i-1)+1, index(i)
      対応する非零非対角成分（AMAT）の成分の修正（行）
    enddo
  endif
enddo

do i= 1, NP 節点ループ
  do k= index (i-1)+1, index (i)
    if (IWKX(item (k),1).eq.1) then   対応する非零非対角成分の
                                      節点がマークされていたら
      対応する右辺ベクトル，非零非対角成分（AMAT）の成分の修正（列）
    endif
  enddo
enddo
```



境界条件：MAT_ASS_BC（1/2）

```
      subroutine MAT_ASS_BC
      use pfem_util
      implicit REAL*8 (A-H,O-Z)

      allocate (IWKX(NP,2))
      IWKX= 0

!C
!C== Z=Zmax

      do in= 1, NP
        IWKX(in,1)= 0
      enddo

      ib0= -1
      do ib0= 1, NODGRPtot
        if (NODGRP_NAME(ib0).eq.'Zmax') exit
      enddo

      do ib= NODGRP_INDEX(ib0-1)+1, NODGRP_INDEX(ib0)
        in= NODGRP_ITEM(ib)
        IWKX(in,1)= 1
      enddo
```

節点グループ名が「Zmax」である節点inにおいて：

IWKX(in,1)= 1

とする

境界条件：MAT_ASS_BC（2/2）

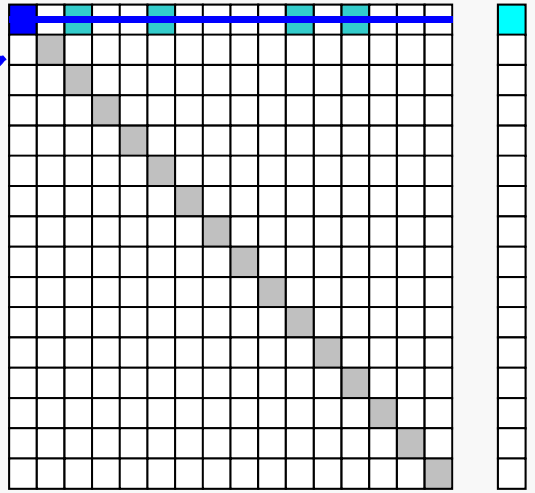
```
      do in= 1, NP
        if (IWKX(in,1).eq.1) then
          B(in)= 0.d0
          D(in)= 1.d0

          iS= index(in-1) + 1
          iE= index(in  )
          do k= iS, iE
            AMAT(k)= 0.d0
          enddo
        endif
      enddo

      do in= 1, NP
        iS= index(in-1) + 1
        iE= index(in  )
        do k= iS, iE
          if (IWKX(item(k),1).eq.1) then
            AMAT(k)= 0.d0
          endif
        enddo
      enddo
!C==
      return
      end
```

境界条件：MAT_ASS_BC（2/2）

```
      do in= 1, NP
        if (IWKX(in,1).eq.1) then
          B(in)= 0.d0
          D(in)= 1.d0

          iS= index(in-1) + 1
          iE= index(in  )
          do k= iS, iE
            AMAT(k)= 0.d0
          enddo
        endif
      enddo

      do in= 1, NP
        iS= index(in-1) + 1
        iE= index(in  )
        do k= iS, iE
          if (IWKX(item(k),1).eq.1) then
            AMAT(k)= 0.d0
          endif
        enddo
      enddo
!C==
      return
      end
```

IWKX(in,1)=1となる節点に対して
対角成分=1，右辺=0，非零対角成分=0

ゼロクリア

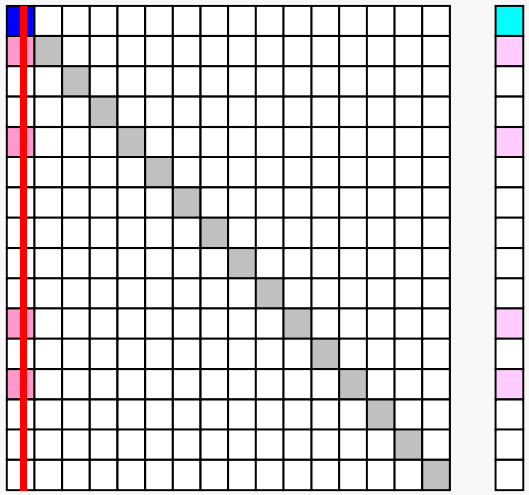


ここでやっていることも1CPUの時と全く変わらない

境界条件：MAT_ASS_BC（2/2）

```
      do in= 1, NP
        if (IWKX(in,1).eq.1) then
          B(in)= 0.d0
          D(in)= 1.d0

          iS= index(in-1) + 1
          iE= index(in  )
          do k= iS, iE
            AMAT(k)= 0.d0
          enddo
        endif
      enddo

      do in= 1, NP
        iS= index(in-1) + 1
        iE= index(in  )
        do k= iS, iE
          if (IWKX(item(k),1).eq.1) then
            AMAT(k)= 0.d0
          endif
        enddo
      enddo
!C==
      return
      end
```

IWKX(in,1)=1となる節点を非零非対角成分として有している節点に対して，右辺へ移項，当該非零非対角成分=0

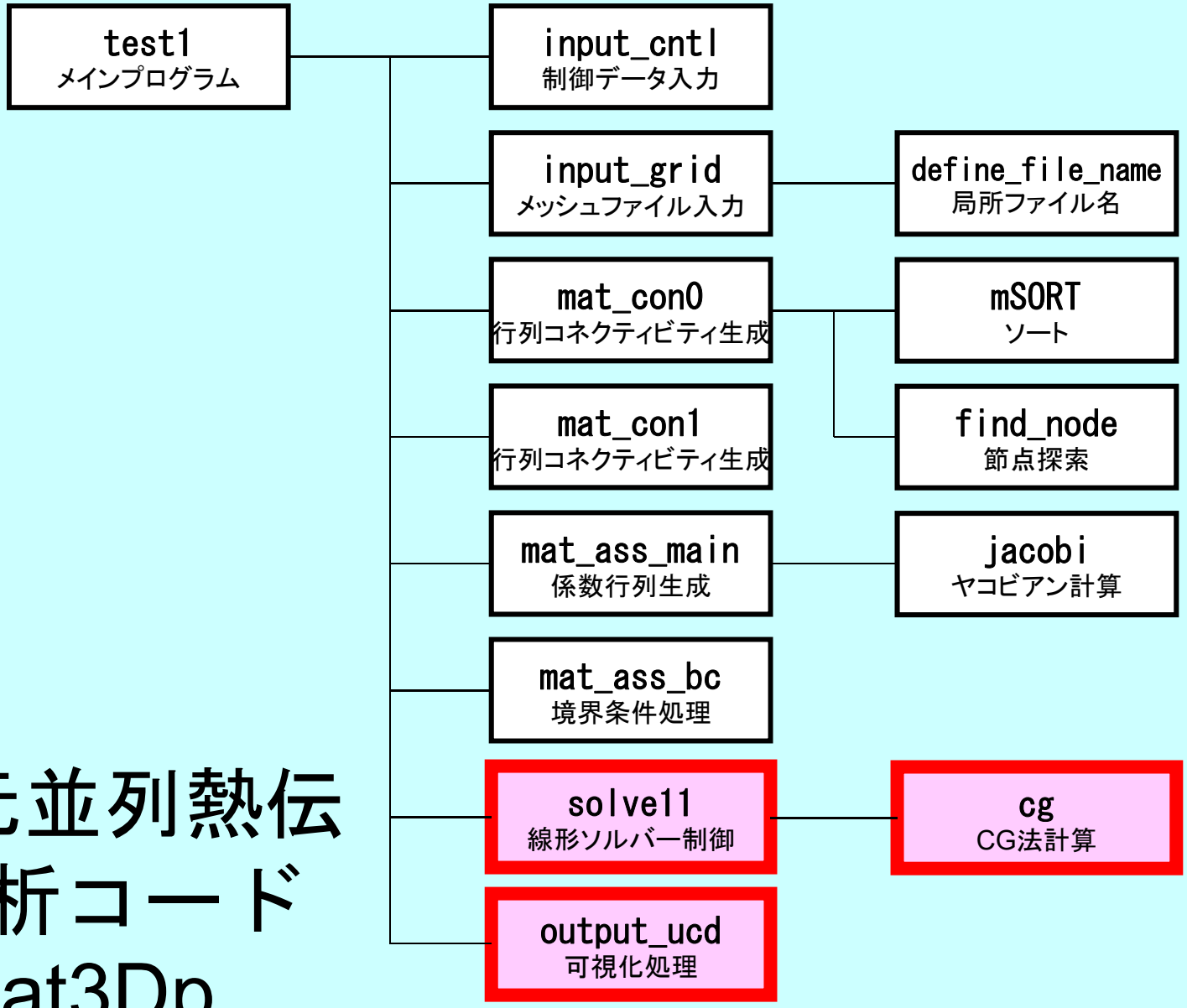


消去，ゼロクリア

ここでやっていることも1CPUの時と全く変わらない

並列有限要素法の処理：プログラム

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



三次元並列熱伝導解析コード heat3Dp

全体処理

```
      program heat3Dp

      use solver11
      use pfem_util

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

      call PFEM_INIT
      call INPUT_CNTL
      call INPUT_GRID

      call MAT_CON0
      call MAT_CON1

      call MAT_ASS_MAIN
      call MAT_ASS_BC

      call SOLVE11

      call OUTPUT_UCD

      call PFEM_FINALIZE

      end program heat3Dp
```

SOLVE11

```
      module SOLVER11

       contains
         subroutine SOLVE11

         use pfem_util
         use solver_CG

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

         integer :: ERROR, ICFLAG
         character(len=char_length) :: BUF

         data ICFLAG/0/

!C
!C +------------+
!C | PARAMETERs |
!C +------------+
!C===
         ITER      = pfemIarray(1)              CG法の最大反復回数
         RESID     = pfemRarray(1)              CG法の収束判定値
!C===
!C
!C +-----------------+
!C | ITERATIVE solver |
!C +-----------------+
!C===
         call CG                                                          &
     &      ( N, NP, NPLU, D, AMAT, index, item, B, X,  RESID,           &
     &        ITER, ERROR, my_rank,                                      &
     &        NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_ITEM,              &
     &                           EXPORT_INDEX, EXPORT_ITEM)
       ITERactual= ITER
!C===

       end subroutine SOLVE11
       end module SOLVER11
```

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

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

前処理：対角スケーリング

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。

CG法（1/6）

```
      subroutine CG                                                     &
     &   (N, NP, NPLU, D, AMAT, index, item, B, X, RESID,               &
     &    ITER, ERROR, my_rank,                                         &
     &    NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_ITEM,                 &
     &                       EXPORT_INDEX, EXPORT_ITEM)

      use  solver_SR

      implicit REAL*8(A-H,O-Z)
      include  'precision.inc'
      include  'mpif.h'

      integer(kind=kint ), intent(in):: N, NP, NPLU, my_rank
      integer(kind=kint ), intent(in):: NEIBPETOT
      integer(kind=kint ), intent(inout):: ITER, ERROR
      real   (kind=kreal), intent(inout):: RESID

      real(kind=kreal), dimension(NP)   , intent(inout):: B, X, D
      real(kind=kreal), dimension(NPLU), intent(inout):: AMAT

      integer(kind=kint ), dimension(0:NP),intent(in) :: index
      integer(kind=kint ), dimension(NPLU),intent(in) :: item

      integer(kind=kint ), pointer :: NEIBPE(:)
      integer(kind=kint ), pointer :: IMPORT_INDEX(:), IMPORT_ITEM(:)
      integer(kind=kint ), pointer :: EXPORT_INDEX(:), EXPORT_ITEM(:)

      real(kind=kreal), dimension(:),    allocatable:: WS, WR          送信バッファ，受信バッファ
      real(kind=kreal), dimension(:,:),  allocatable:: WW

      integer(kind=kint), parameter ::  R= 1
      integer(kind=kint), parameter ::  Z= 2
      integer(kind=kint), parameter ::  Q= 2
      integer(kind=kint), parameter ::  P= 3
      integer(kind=kint), parameter :: DD= 4

      integer(kind=kint ) :: MAXIT
      real   (kind=kreal) :: TOL, W, SS
```

CG法（2/6）

```
 COMMtime= 0.d0
      COMPtime= 0.d0

      ERROR= 0

      allocate (WW(NP,4),WR(NP),WS(NP))

      MAXIT  = ITER
       TOL   = RESID

      X = 0.d0
      WS= 0.d0
      WR= 0.d0
!C
!C +----------------------+
!C | {r0}= {b} - [A]{xini} |
!C +----------------------+
!C===

      call SOLVER_SEND_RECV
     &    ( NP, NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_ITEM,
     &      EXPORT_INDEX, EXPORT_ITEM, WS, WR, X , my_rank)

      do j= 1, N
        WW(j,DD)= 1.d0/D(j)
        WVAL= B(j) - D(j)*X(j)
        do k= index(j-1)+1, index(j)
          i= item(k)
          WVAL= WVAL - AMAT(k)*X(i)
        enddo
        WW(j,R)= WVAL
      enddo

      BNRM20= 0.d0
      do i= 1, N
        BNRM20= BNRM20 + B(i)**2
      enddo
      call MPI_Allreduce (BNRM20, BNRM2, 1, MPI_DOUBLE_PRECISION,       &
     &                    MPI_SUM, MPI_COMM_WORLD, ierr)
```

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

SOLVER SEND RECV (1/2)

```
 subroutine  SOLVER_SEND_RECV                                          &
&                ( N, NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_ITEM,&
&                                        EXPORT_INDEX, EXPORT_ITEM,&
&                  WS, WR, X, my_rank)

 implicit REAL*8 (A-H,O-Z)
 include  'mpif.h'
 include  'precision.inc'
 integer(kind=kint )                 , intent(in)    ::  N
 integer(kind=kint )                 , intent(in)    ::  NEIBPETOT
 integer(kind=kint ), pointer :: NEIBPE      (:)
 integer(kind=kint ), pointer :: IMPORT_INDEX(:)
 integer(kind=kint ), pointer :: IMPORT_ITEM  (:)
 integer(kind=kint ), pointer :: EXPORT_INDEX(:)
 integer(kind=kint ), pointer :: EXPORT_ITEM  (:)
 real   (kind=kreal), dimension(N), intent(inout):: WS
 real   (kind=kreal), dimension(N), intent(inout):: WR
 real   (kind=kreal), dimension(N), intent(inout):: X
 integer                           , intent(in)    :: my_rank
 integer(kind=kint ), dimension(:,:), save, allocatable :: sta1, sta2, req1, req2
 integer(kind=kint ), save :: NFLAG
 data NFLAG/0/

 if (NFLAG.eq.0) then
   allocate (sta1(MPI_STATUS_SIZE,NEIBPETOT), sta2(MPI_STATUS_SIZE,NEIBPETOT))
   allocate (req1(NEIBPETOT), req2(NEIBPETOT))
   NFLAG= 1
 endif

 do neib= 1, NEIBPETOT
   istart= EXPORT_INDEX(neib-1)
   inum  = EXPORT_INDEX(neib  ) - istart
   do k= istart+1, istart+inum
      ii   = EXPORT_ITEM(k)
      WS(k)= X(ii)
   enddo

   call MPI_Isend (WS(istart+1), inum, MPI_DOUBLE_PRECISION,        &
&                  NEIBPE(neib), 0, MPI_COMM_WORLD, req1(neib),    &
&                  ierr)
 enddo
```

SOLVER_SEND_RECV (2/2)

```
      do neib= 1, NEIBPETOT
        istart= IMPORT_INDEX(neib-1)
        inum  = IMPORT_INDEX(neib  ) - istart
        call MPI_Irecv (WR(istart+1), inum, MPI_DOUBLE_PRECISION,       &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD, req2(neib),    &
     &                  ierr)
      enddo

      call MPI_Waitall (NEIBPETOT, req2, sta2, ierr)

      do neib= 1, NEIBPETOT
        istart= IMPORT_INDEX(neib-1)
        inum  = IMPORT_INDEX(neib  ) - istart
        do k= istart+1, istart+inum
            ii = IMPORT_ITEM(k)
          X(ii)= WR(k)
        enddo
      enddo

      call MPI_Waitall (NEIBPETOT, req1, sta1, ierr)

      end subroutine solver_send_recv
      end module     solver_SR
```

CG法（3/6）

```
 do iter= 1, MAXIT

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

!C
!C +-----------------+
!C | {RHO}= {r}{z}   |
!C +-----------------+
      RHO0= 0.d0

      do i= 1, N
        RHO0= RHO0 + WW(i,R)*WW(i,Z)
      enddo

      call MPI_Allreduce (RHO0, RHO, 1, MPI_DOUBLE_PRECISION,
     &                    MPI_SUM, MPI_COMM_WORLD, ierr)

!C
!C +-----------------------------+
!C | {p} = {z} if      ITER=1    |
!C | BETA= RHO / RHO1  otherwise |
!C +-----------------------------+
      if ( ITER.eq.1 ) then
        do i= 1, N
          WW(i,P)= WW(i,Z)
        enddo
       else
        BETA= RHO / RHO1
        do i= 1, N
          WW(i,P)= WW(i,Z) + BETA*WW(i,P)
        enddo
      endif
```

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

CG法（4/6）

```
!C
!C +-------------+
!C | {q}= [A]{p} |
!C +-------------+
      call SOLVER_SEND_RECV
     &    ( NP, NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_ITEM,
     &      EXPORT_INDEX, EXPORT_ITEM, WS, WR, WW(1,P),
     &      my_rank)

      do j= 1, N
        WVAL= D(j)*WW(j,P)
        do k= index(j-1)+1, index(j)
           i= item(k)
          X1= WW(3*i-2,P)
          X2= WW(3*i-1,P)
          X3= WW(3*i  ,P)
          WVAL= WVAL + AMAT(k)*WW(i,P)
        enddo
        WW(j,Q)= WVAL
      enddo

!C
!C +---------------------+
!C | ALPHA= RHO / {p}{q} |
!C +---------------------+
      C10= 0.d0
      do i= 1, N
        C10= C10 + WW(i,P)*WW(i,Q)
      enddo
      call MPI_Allreduce (C10, C1, 1, MPI_DOUBLE_PRECISION,
     &                    MPI_SUM, MPI_COMM_WORLD, ierr)

      ALPHA= RHO / C1
```

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

CG法（5/6）

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

      DNRM20= 0.d0
      do i= 1, N
        DNRM20= DNRM20 + WW(i,R)**2
      enddo
      call MPI_Allreduce (DNRM20, DNRM2, 1,
     &                    MPI_DOUBLE_PRECISION,
     &                    MPI_SUM, MPI_COMM_WORLD, ierr)

      RESID= dsqrt(DNRM2/BNRM2)
        if ( RESID.le.TOL    ) exit
        if ( ITER .eq.MAXIT ) ERROR= -300

        RHO1 = RHO

      enddo
!C===

   30 continue

      call SOLVER_SEND_RECV
     &   ( NP, NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_ITEM,
     &     EXPORT_INDEX, EXPORT_ITEM, WS, WR, X , my_rank)

      deallocate (WW,WR,WS)

      end subroutine        CG
      end module     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
  - $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
  - $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
- 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|**

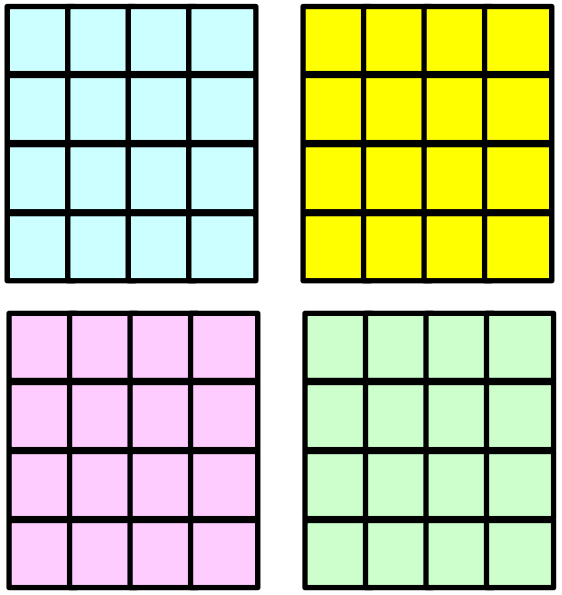
end

CG法（6/6）

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

      DNRM20= 0.d0
      do i= 1, N
        DNRM20= DNRM20 + WW(i,R)**2
      enddo
      call MPI_Allreduce (DNRM20, DNRM2, 1,
     &                    MPI_DOUBLE_PRECISION,
     &                    MPI_SUM, MPI_COMM_WORLD, ierr)

      RESID= dsqrt(DNRM2/BNRM2)
        if ( RESID.le.TOL    ) exit
        if ( ITER .eq.MAXIT ) ERROR= -300

        RHO1 = RHO

      enddo
!C===

   30 continue

      call SOLVER_SEND_RECV                                          &          外点に最新のX（温度）代入
     &   ( NP, NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_ITEM,         &
     &     EXPORT_INDEX, EXPORT_ITEM, WS, WR, X , my_rank)

      deallocate (WW,WR,WS)

      end subroutine        CG
      end module     solver_CG
```

OUTPUT_UCD

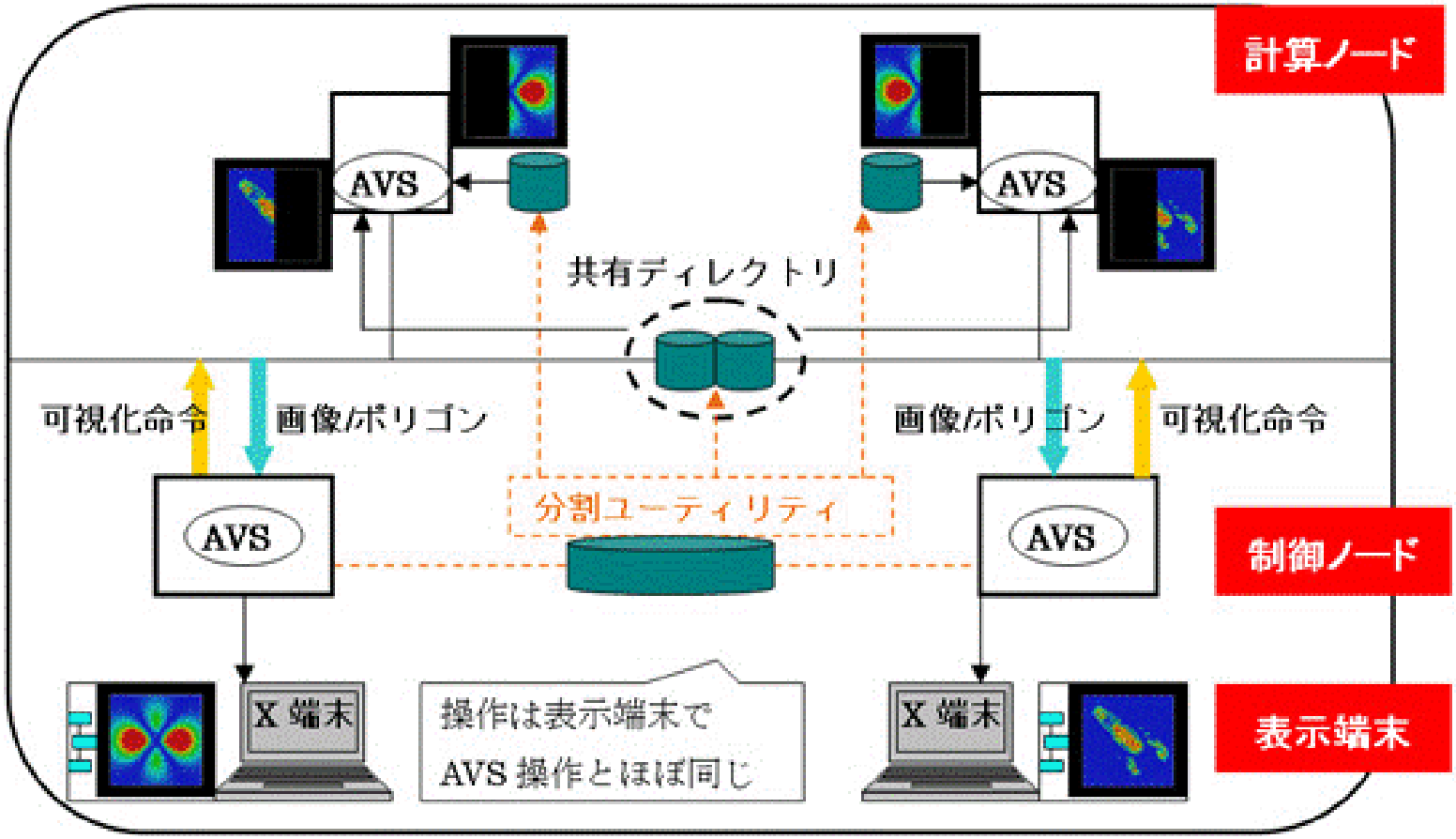
- 各領域からintELEM_listに所属する要素の情報を集める
  - intELEM_list: 各節点の属する領域番号のうち最も若い番号の領域に所属する・・・と見なす
- MPI_Allgathervを使って以下の情報を一箇所に集める
  - 節点:節点座標，温度
  - 要素:要素コネクティビティ(要素を構成する節点)
- 節点の情報は一部重複
- 問題規模が大きくなると困難
  - あまり賢いやり方ではない
  - 並列可視化



AVS/Express PCE Parallel Cluster Edition

- http://www.cybernet.co.jp/avs/products/pce/
- AVS/Express PCEでは，クラスタ化された複数のLinuxマシンで，各計算ノードが持つ部分領域のみを可視化し，最終的な可視化結果のみ制御ノード上で表示するという構成になっている。
- 並列計算の結果，出力される大規模データを可視化する場合でも，高い精度を保ったまま，可視化処理を実現することが可能。
- 並列計算機上で対話処理可能
  - 最新版ではWindowsより制御可能

AVS/Express PCE Parallel Cluster Edition

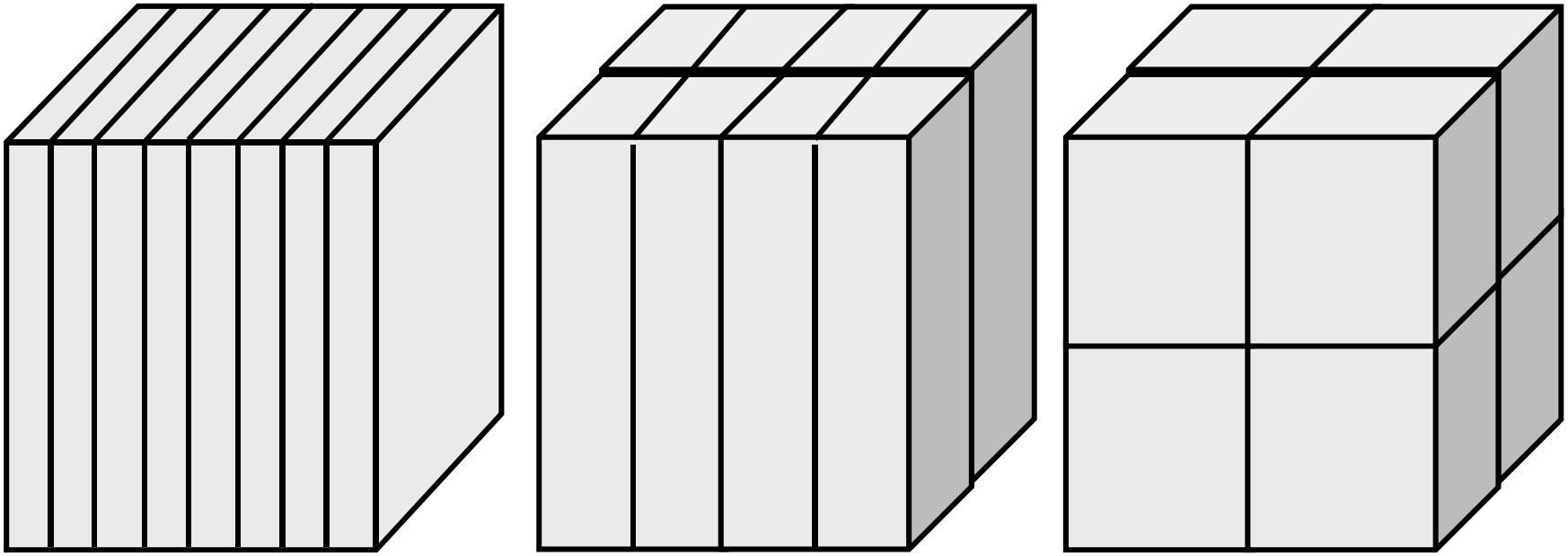


課題(1/2)

- 自分で問題設定を行い，「sol」の挙動を分析してみよ
- 例
  - Strong Scaling
    - 問題サイズを固定，PE数を変化させて時間(全体，各部分)を測定。
  - Weak Scaling
    - PEあたりの問題サイズを固定，1反復あたりの計算時間を求める。
  - 考慮すべき項目
    - 問題サイズ
    - 領域分割手法(RCB，K-METIS，P-METIS，1D～3D)の影響。
  - メッシュ生成，領域分割におけるFX10の性能低い
    - 128^3くらいが限界(領域分割に15分以上かかる)
- 「*.inp」の出力に時間がかかる場合がある。
  - OUTPUT_UCDの呼び出しのコメントアウト
  - src，part

1D～3D分割

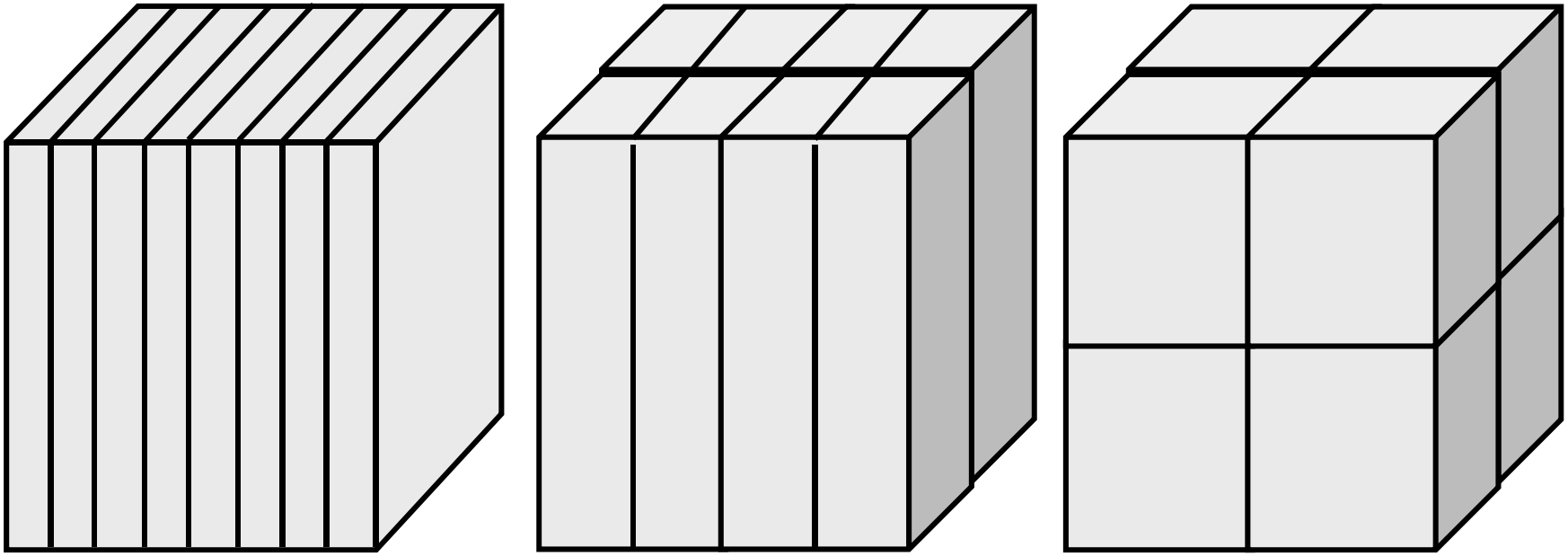
どの方法が良いか考えて見よ



1D型　2D型　3D型

1D～3D分割

通信量の総和（各辺4N，8領域とする）

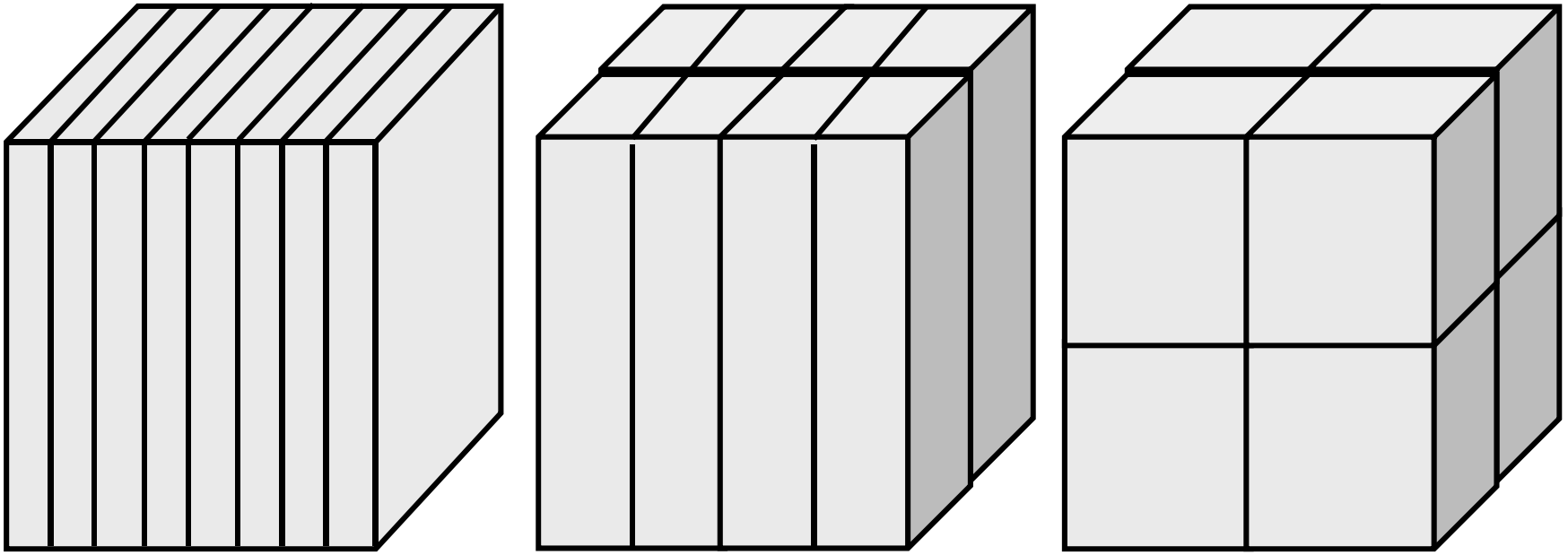


1D型　　2D型　　3D型

16 N^2 x 7 = 112 N^2　　16 N^2 x 4 = 64 N^2　　16 N^2 x 3 = 48 N^2

1D～3D分割

mesh.inp



1D型

```
64  64  64
 8   1   1
pcube
```

2D型

```
64  64  64
 4   2   1
pcube
```

3D型

```
64  64  64
 2   2   2
pcube
```

課題(2/2)

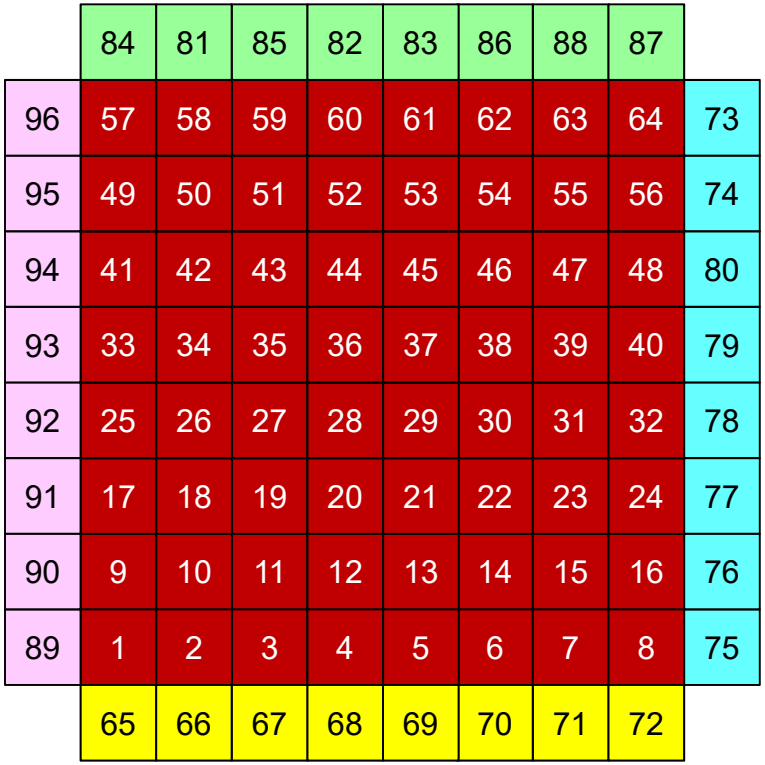
- 領域間通信(solver_SR)の性能改善ができないかどうか考えてみよ。
  - Recv. Bufferへのコピーを効率的に実施できないか?

```
      do neib= 1, NEIBPETOT
        istart= IMPORT_INDEX(neib-1)
        inum  = IMPORT_INDEX(neib  ) - istart
        call MPI_Irecv (WR(istart+1), inum, MPI_DOUBLE_PRECISION,       &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD, req2(neib),    &
     &                  ierr)
      enddo

      call MPI_Waitall (NEIBPETOT, req2, sta2, ierr)

      do neib= 1, NEIBPETOT
        istart= IMPORT_INDEX(neib-1)
        inum  = IMPORT_INDEX(neib  ) - istart
        do k= istart+1, istart+inum
            ii = IMPORT_ITEM(k)
          X(ii)= WR(k)
        enddo
      enddo
```

SEND/RECV (Original)

```
!C
!C-- INIT.
      allocate (sta1(MPI_STATUS_SIZE,NEIBPETOT), sta2(MPI_STATUS_SIZE,NEIBPETOT))
      allocate (req1(NEIBPETOT), req2(NEIBPETOT))

!C
!C-- SEND
      do neib= 1, NEIBPETOT
        istart= STACK_EXPORT(neib-1)
        inum  = STACK_EXPORT(neib  ) - istart
        do k= istart+1, istart+inum
          WS(k)= X(NOD_EXPORT(k))
        enddo
        call MPI_ISEND (WS(istart+1), inum, MPI_DOUBLE_PRECISION,           &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD, req1(neib), ierr)
      enddo

!C
!C-- RECEIVE
      do neib= 1, NEIBPETOT
        istart= STACK_IMPORT(neib-1)
        inum  = STACK_IMPORT(neib  ) - istart
        call MPI_IRECV (WR(istart+1), inum, MPI_DOUBLE_PRECISION,           &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD, req2(neib), ierr)
      enddo
      call MPI_WAITALL (NEIBPETOT, req2, sta2, ierr)

      do neib= 1, NEIBPETOT
        istart= STACK_IMPORT(neib-1)
        inum  = STACK_IMPORT(neib  ) - istart
        do k= istart+1, istart+inum
          X(NOD_IMPORT(k))= WR(k)
        enddo
      enddo
      call MPI_WAITALL (NEIBPETOT, req1, sta1, ierr)
```

If numbering of external nodes is continuous in each neighboring process ...



SEND/RECV (NEW:1)

```
!C
!C-- INIT.
      allocate (sta1(MPI_STATUS_SIZE,2*NEIBPETOT))
      allocate (req1(2*NEIBPETOT))

!C
!C-- SEND
      do neib= 1, NEIBPETOT
        istart= STACK_EXPORT(neib-1)
        inum  = STACK_EXPORT(neib  ) - istart
        do k= istart+1, istart+inum
          WS(k)= X(NOD_EXPORT(k))
        enddo
      enddo

      do neib= 1, NEIBPETOT
        istart= STACK_EXPORT(neib-1)
        inum  = STACK_EXPORT(neib  ) - istart
        call MPI_ISEND (WS(istart+1), inum, MPI_DOUBLE_PRECISION,                    &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD, req1(neib), ierr)
      enddo

!C
!C-- RECEIVE
      do neib= 1, NEIBPETOT
        inum  = STACK_IMPORT(neib) - STACK_IMPORT(neib-1)
        istart= NOD_IMPORT(STACK_IMPORT(neib-1)+1)

        call MPI_IRECV (X(istart), inum, MPI_DOUBLE_PRECISION,                       &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD, req1(NEIBPETOT+neib), ierr)
      enddo

      call MPI_WAITALL (2*NEIBPETOT, req1, sta1, ierr)
```

SEND/RECV (NEW:2), N0: int. node

N0: 内点総数

```
!C
!C-- INIT.
      allocate (sta1(MPI_STATUS_SIZE,2*NEIBPETOT))
      allocate (req1(2*NEIBPETOT))

!C
!C-- SEND
      do neib= 1, NEIBPETOT
        istart= STACK_EXPORT(neib-1)
        inum  = STACK_EXPORT(neib  ) - istart
        do k= istart+1, istart+inum
          WS(k)= X(NOD_EXPORT(k))
        enddo
      enddo

      do neib= 1, NEIBPETOT
        istart= STACK_EXPORT(neib-1)
        inum  = STACK_EXPORT(neib  ) - istart
        call MPI_ISEND (WS(istart+1), inum, MPI_DOUBLE_PRECISION,                  &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD, req1(neib), ierr)
      enddo

!C
!C-- RECEIVE
      do neib= 1, NEIBPETOT
        inum  = STACK_IMPORT(neib) - STACK_IMPORT(neib-1)
        istart= STACK_IMPORT(neib-1) + N0 + 1

        call MPI_IRECV (X(istart), inum, MPI_DOUBLE_PRECISION,                     &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD, req1(NEIBPETOT+neib), ierr)
      enddo

      call MPI_WAITALL (2*NEIBPETOT, req1, sta1, ierr)
```

計算例(通信最適化済)@東大 Strong Scaling

- 192×192×128節点(4,718,592節点，4,633,087要素)
- 16～192コア
- Solver計算時間

core #	sec. (speed-up)
16（4x2x2）	16.6（16.0）
32（4x4x2）	8.69（30.5）
64（4x4x4）	4.55（59.6）
96（6x4x4）	3.54（74.7）
128（8x4x4）	2.69（98.4）
192（8x6x4）	1.84（144.）

ファイルコピー on FX10

FORTRANユーザー

```
>$ cd ~/pFEM
>$ cp/home/S11502/nakajima/2015Summer/F/fem3d0.tar .
>$ tar xvf fem3d.tar
```

Cユーザー

```
>$ cd ~/pFEM
>$ cp/home/S11502/nakajima/2015Summer/C/fem3d0.tar .
>$ tar xvf fem3d0.tar
```

ディレクトリ確認

```
>$ cd pfem3d
>$ cd src0
>$ make
>$ ls ../run/sol0
   sol0
>$ cd ../run
>$ <modify “go0.sh”>
>$ pjsub go0.sh
```