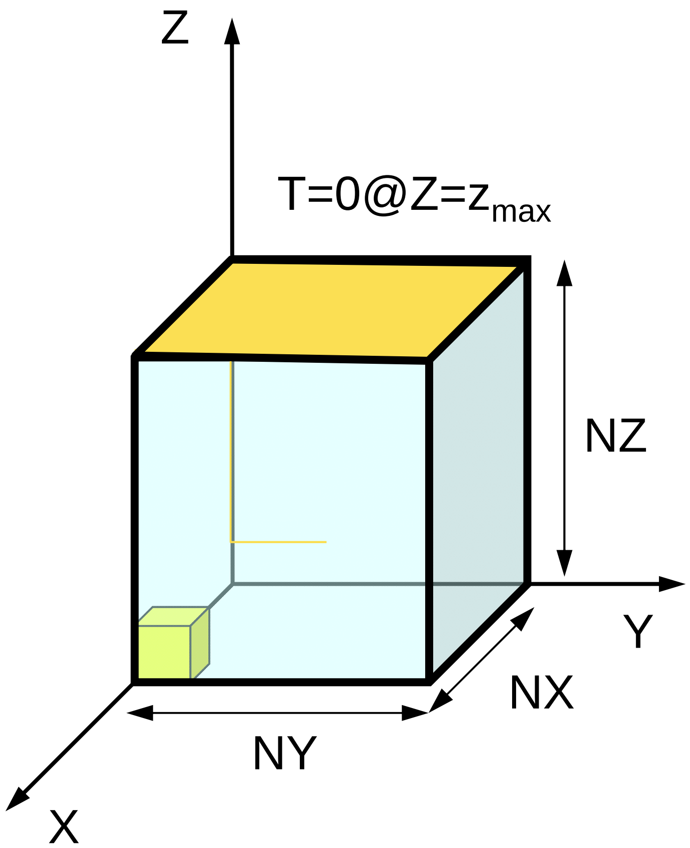
並列有限要素法による
三次元定常熱伝導解析プログラム
by ppOpen-APPL/FVM
（2/2）

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

ソースコード

- **<$ppohFVM>/examples/heat3D/src**
  - OpenMP/MPI Hybrid＋簡易可視化機能
  - pmesh，pmesh_binで作成したメッシュしか可視化できない（計算はできる）
- **Flat MPIにしたいときは<$ppohFVM>/Makefileを以下のように修正して再コンパイル（,openmpトル）**

```
# Install directory
PREFIX          = /home/z30088/ppohFILES
INCDIR          = $(PREFIX)/include
LIBDIR          = $(PREFIX)/lib
BINDIR          = $(PREFIX)/bin

# Fortran compiler settings
F90             = frtpx
F77             = frtpx
MPIF90          = mpifrtpx
MPIF77          = mpifrtpx
sFFLAGS         = -Kfast
sMGFLAGS        = -Kfast
pFFLAGS         = -Kfast,openmp
```

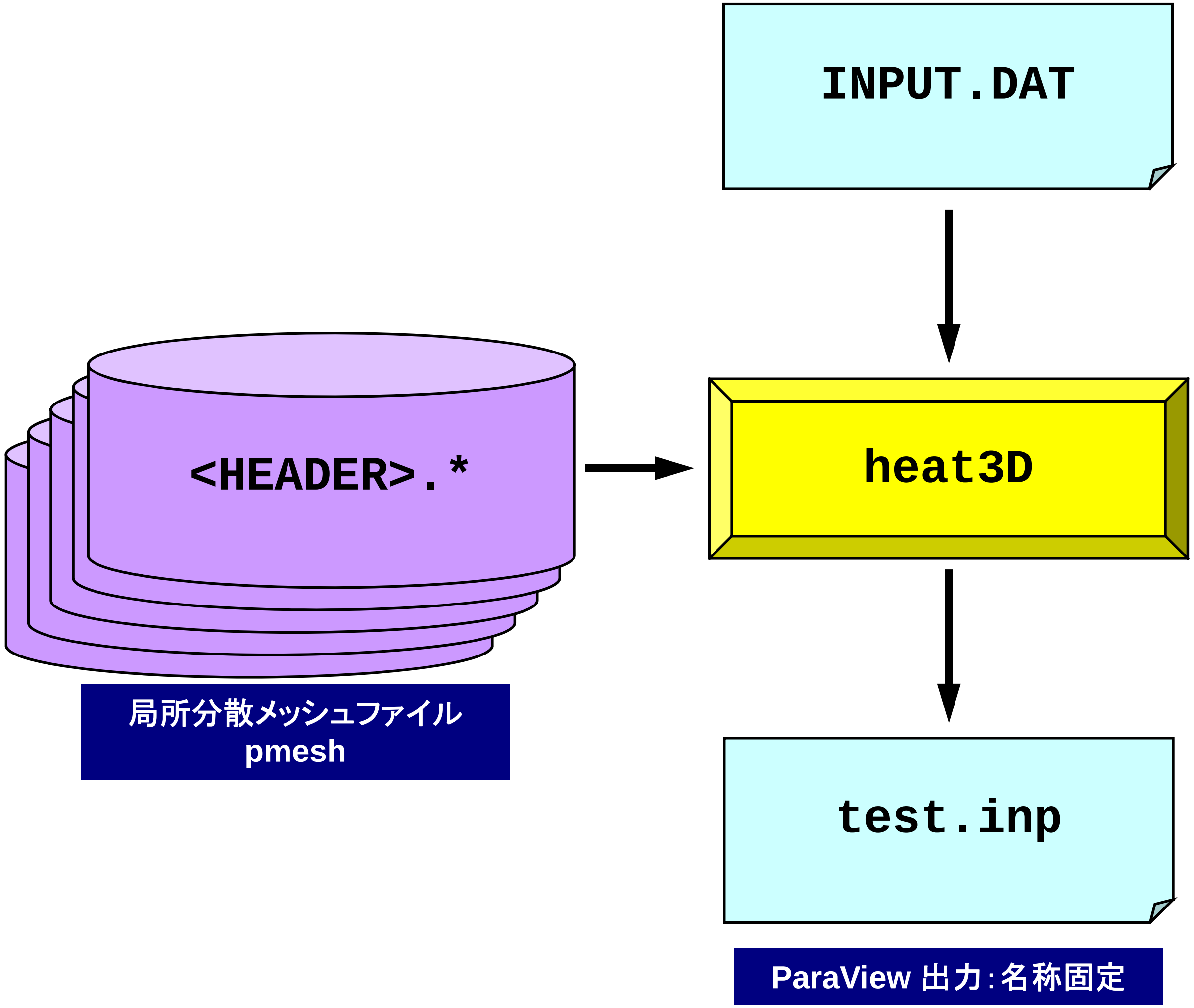
有限要素法の処理

- 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
  - メッシュファイル読み込み（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

```
../../pmesh/pcube_asci   HEADER
2000                     ITER
1.0 1.0                  COND, QVOL
1.0e-08                  RESID
T                        MESH_ASCI
1000                     N_MESH_VIS
```

- 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
- ASCI_MESH： メッシュファイル形式，ASCI:T, BIN:F
- N_MESH_VIS: 簡易可視化機能における表示メッシュ数の目安

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

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

<$ppohFVM>/examples/heat3D/run/go.sh

```
#!/bin/sh

#PJM -L "rscgrp=tutorial"
#PJM -L "node=8"
#PJM --mpi "proc=8"
#PJM -g "gt00"
#PJM -L "elapse=00:30:00"
#PJM -j
#PJM -o "test01.lst“
export OMP_NUM_THREADS=16

mpiexec ./../../../bin/heat3D
```

8分割
“node=1“
“proc=8”

16分割
“node=1“
“proc=16”

32分割
“node=2“
“proc=32”

64分割
“node=4“
“proc=64”

192分割
“node=12“
“proc=192”

<$ppohFVM>/examples/heat3D/run/gof.sh

```
#!/bin/sh

#PJM -L "rscgrp=tutorial"
#PJM -L "node=8"
#PJM --mpi "proc=128"
#PJM -g "gt00"
#PJM -L "elapse=00:30:00"
#PJM -j
#PJM -o "test01.lst"

mpiexec ./../../../bin/heat3D
```

8分割
"node=1"
"proc=8"

16分割
"node=1"
"proc=16"

32分割
"node=2"
"proc=32"

64分割
"node=4"
"proc=64"

192分割
"node=12"
"proc=192"

全体処理: test1.f（1/2）

赤字:ppOpen-HPC（現在）
青字:ppOpen-HPC（将来）
黒字:ユーザー定義

```
      program heat3Dp

      use ppohFVM_util
      use ppohFVM_util_matrix

      use pfem_util

      implicit REAL*8(A-H,O-Z)
      type (ppohFVM_file_info)   ::   file_info
      type (ppohFVM_local_mesh)  ::  local_mesh
      type (ppohFVM_grp_data)    ::    grp_data
      type (ppohFVM_comm_info)   ::   comm_info
      type (ppohFVM_edge_info)   ::   edge_info
      type (ppohFVM_matrix_info) :: matrix_info
      type (ppohFVM_solver_info) :: solver_info
      type (ppohFVM_vis_info)    ::    vis_info
!C
!C +--------+
!C | INIT.  |
!C +--------+
!C===
      call ppohFVM_Init (file_info, comm_info, edge_info)

      PETOT    = comm_info%PETOT
      PEsmpTOT= comm_info%PEsmpTOT
      my_rank = comm_info%my_rank

      call INPUT_CNTL (file_info, comm_info, edge_info, matrix_info, solver_info, vis_info)
      call ppohFVM_dist_file (file_info, HEADER, 80, 0, my_rank, 1)

      call ppohFVM_pre (file_info, local_mesh, grp_data, comm_info, edge_info)
      call LOAD_MESH   (           local_mesh, grp_data, comm_info)
!C===
```

全体処理: test1.f（2/2）

```
!C
!C +---------------------------------+
!C | Matrix Connectivity/Assembling |
!C +---------------------------------+
!C===
      call ppohFVM_mat_con (local_mesh, comm_info, matrix_info)
      call MAT_ASS_MAIN (local_mesh, matrix_info)
      call MAT_ASS_BC   (local_mesh, matrix_info)
!C===

!C
!C +--------+
!C | SOLVER |
!C +--------+
!C===
      call ppohFVM_solver11 (local_mesh, comm_info, matrix_info, solver_info)
!C===

!C
!C +--------+
!C | OUTPUT |
!C +--------+
!C===
      call ppohFVM_ucd_regular_hexa_1 (local_mesh, comm_info, matrix_info, vis_info)
!C===

      call ppohFVM_Finalize (comm_info)

      end program heat3Dp
```

開始，終了：MPI_Init/Finalize

```
subroutine ppohFVM_Init (file_info, comm_info, edge_info)
use ppohFVM_util
implicit REAL*8 (A-H,O-Z)
type (ppohFVM_file_info)  ::  file_info
type (ppohFVM_comm_info)  ::  comm_info
type (ppohFVM_edge_info)  ::  edge_info

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

file_info%mesh_asci= .true.
edge_info%use_edges= .true.

return
end
```

```
subroutine ppohFVM_Finalize (comm_info)
use ppohFVM_util
implicit REAL*8 (A-H,O-Z)
integer:: errno
type (ppohFVM_comm_info)  ::  comm_info

call MPI_Finalize (errno)
if (comm_info%my_rank.eq.0) stop ' * normal termination'

return
end
```

ppohFVM_util_matrix（1/2）

ppohFVM_matrix_info%TYPE

```
= 1: [A] without [D]
= 2: [A] with    [D]
= 3: [L], [U], and [D]
```

ppohFVM_matrix_info%BLOCKsize

```
= 1: 1x1 block
= 2: 2x2 block
= 3: 3x3 block
= 4: 4x4 block
```

ppohFVM_matrix_info%DomainDecomposition

```
= 0: LBJ (Localized Block Jacobi)
= 1: LBJ with 1-layer overlapping extention
= 2: LBJ with 2-layer overlapping extention
= 3: LBJ with 3-layer overlapping extention
=10: LBJ/RCM global
=11: LBJ/RCM global with 1-layer overlapping extention
=12: LBJ/RCM global with 2-layer overlapping extention
=13: LBJ/RCM global with 3-layer overlapping extention
=20: HID with 1-layer overlapping extention
=21: HID with 1-layer overlapping extention
=22: HID with 1-layer overlapping extention
=23: HID with 1-layer overlapping extention
```

ppohFVM_solver_info%PRECOND

```
= 0: NO preconditioning
= 1: Point/Block Jacobi
=10: ILU(0)/IC(0)
=11: ILU(1)/IC(1)
=12: ILU(2)/IC(2)
=13: ILU(3)/IC(3)
```

ppohFVM_solver_info%METHOD

CG, GMRES, etc.

ppohFVM_util_matrix（2/2）

```
module ppohFVM_util_matrix

use ppohFVM_util

implicit none
public

type ppohFVM_matrix_info
  integer TYPE, BLOCKsize, DomainDecomposition
  integer N, NP
  integer NL , NU , NLU
  integer NPL, NPU, NPLU
  integer,pointer:: INL(:)  , INU(:)  , INLU(:)
  integer,pointer:: IAL(:,:), IAU(:,:), IALU(:,:)

  integer, dimension(:), allocatable :: indexL, indexU, index
  integer, dimension(:), allocatable ::  itemL,  itemU, item

  real(kind=ppohFVM_kreal), dimension(:), allocatable :: AL, AU, D, RHS, X, AMAT

  integer hexa_color_tot
  integer, dimension(:), allocatable :: hexa_color_index, hexa_color_item
end type ppohFVM_matrix_info

type ppohFVM_solver_info
  integer METHOD, PRECOND, ITER, ITERactual, ERROR, ICFLAG
  real(kind=ppohFVM_kreal) :: RESID
  real(kind=ppohFVM_kreal) :: COMMtime, COMPtime
end type ppohFVM_solver_info

type ppohFVM_vis_info
  integer n_cell_ucd_reg_hexa_1
end type ppohFVM_vis_info

end module ppohFVM_util_matrix
```

疎行列ベクトル積

CRS形式，対角・非対角成分

```
do j= 1, N
  Q(j)= matrix_info%D(j)*P(j)
  do k= matrix_info%index(j-1)+1, matrix_info%index(j)
      i = matrix_info%item(k)
    Q(j)= Q(j) + matrix_info%AMAT(k)*P(i)
  enddo
enddo
```

制御情報：INPUT_CNTL（1/2）

```
      subroutine INPUT_CNTL (file_info, comm_info, edge_info,
     &                       matrix_info, solver_info, vis_info)

      use ppohFVM_util
      use ppohFVM_util_matrix
      use pfem_util

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

      type (ppohFVM_file_info)   ::   file_info
      type (ppohFVM_comm_info)   ::   comm_info
      type (ppohFVM_edge_info)   ::   edge_info
      type (ppohFVM_matrix_info) :: matrix_info
      type (ppohFVM_solver_info) :: solver_info
      type (ppohFVM_vis_info)    ::    vis_info

      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
        read (11,*) file_info%mesh_asci
        read (11,*) vis_info%n_cell_ucd_reg_hexa_1
        close (11)

        write (*,'(a80)') HEADER
        write (*,*) file_info%mesh_asci
        write (11,'(i10)') vis_info%n_cell_ucd_reg_hexa_1
        write (*,*) ' '
      endif
```

制御情報：INPUT_CNTL（2/2）

```
call ppohFVM_Bcast_C (HEADER, 80, 0)
call ppohFVM_Bcast_I (ITER , 0)
call ppohFVM_Bcast_R (COND , 0)
call ppohFVM_Bcast_R (QVOL , 0)
call ppohFVM_Bcast_R (RESID, 0)
call ppohFVM_Bcast_L (file_info%mesh_asci, 0)
call ppohFVM_Bcast_I (vis_info%n_cell_ucd_reg_hexa_1, 0)

edge_info%use_edges= .false.

solver_info%RESID= RESID
solver_info%ITER = ITER

matrix_info%TYPE               = 2
matrix_info%BLOCKsize          = 1
matrix_info%DomainDecomposition= 0
solver_info%PRECOND            = 1

return
end
```

ppohFVM_Bcast_X

現在はスカラー版のみ

```
subroutine ppohFVM_Bcast_R ( VAL, nbase )
use ppohFVM_util
implicit REAL*8 (A-H,O-Z)
integer :: nbase, ierr
real(kind=ppohFVM_kreal) :: VAL

call MPI_Bcast (VAL, 1, MPI_DOUBLE_PRECISION, nbase, MPI_COMM_WORLD, ierr)
end subroutine ppohFVM_Bcast_R

subroutine ppohFVM_Bcast_I ( VAL, nbase )
use ppohFVM_util
implicit REAL*8 (A-H,O-Z)
integer :: nbase, ierr
integer :: VAL

call MPI_Bcast (VAL, 1, MPI_INTEGER, nbase, MPI_COMM_WORLD, ierr)
end subroutine ppohFVM_Bcast_I

subroutine ppohFVM_Bcast_C ( VAL, nn, nbase )
use ppohFVM_util
implicit REAL*8 (A-H,O-Z)
integer :: nn, nbase, ierr
character(len=nn) :: VAL

call MPI_Bcast (VAL, nn, MPI_CHARACTER, nbase, MPI_COMM_WORLD, ierr)
end subroutine ppohFVM_Bcast_C

subroutine ppohFVM_Bcast_L ( VAL, nbase )
use ppohFVM_util
implicit REAL*8 (A-H,O-Z)
integer :: nbase, ierr
logical :: VAL

call MPI_Bcast (VAL, 1, MPI_LOGICAL, nbase, MPI_COMM_WORLD, ierr)
end subroutine ppohFVM_Bcast_L
```

LOAD_MESH（1/2） ポインタのコピー

```
      subroutine LOAD_MESH (local_mesh, grp_data, comm_info)

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

      type (ppohFVM_local_mesh) :: local_mesh
      type (ppohFVM_grp_data)   ::   grp_data
      type (ppohFVM_comm_info)  ::  comm_info
!C
!C-- Parallel Info.
      NEIBPETOT =  comm_info%n_neighbor_pe
      NEIBPE    => comm_info%neighbor_pe

      IMPORT_INDEX => comm_info%import_index
      IMPORT_ITEM  => comm_info%import_item
      EXPORT_INDEX => comm_info%export_index
      EXPORT_ITEM  => comm_info%export_item
!C
!C-- MESH
      NP= local_mesh%n_node
      N = local_mesh%n_internal
      ICELTOT     = local_mesh%n_elem
      ICELTOT_INT= local_mesh%ne_internal

      NODE_ID   => local_mesh%node_id
      ELEM_ID   => local_mesh%elem_id

      intELEM_list => local_mesh%ne_internal_list

      allocate (XYZ(NP,3))
      do i= 1, NP
        XYZ(i,1)= local_mesh%node(1,i)
        XYZ(i,2)= local_mesh%node(2,i)
        XYZ(i,3)= local_mesh%node(3,i)
      enddo
```

これは今後の技術的課題

LOAD_MESH（2/2）
ポインタのコピー

```
allocate (ICELNOD(ICELTOT,8))
do icel= 1, ICELTOT
  ICELNOD(icel,1)= local_mesh%ptr_elem(8*icel-7)
  ICELNOD(icel,2)= local_mesh%ptr_elem(8*icel-6)
  ICELNOD(icel,3)= local_mesh%ptr_elem(8*icel-5)
  ICELNOD(icel,4)= local_mesh%ptr_elem(8*icel-4)
  ICELNOD(icel,5)= local_mesh%ptr_elem(8*icel-3)
  ICELNOD(icel,6)= local_mesh%ptr_elem(8*icel-2)
  ICELNOD(icel,7)= local_mesh%ptr_elem(8*icel-1)
  ICELNOD(icel,8)= local_mesh%ptr_elem(8*icel-0)
enddo

if (N.le.0)       call ppohFVM_error_exit(1001)
if (ICELTOT.le.0) call ppohFVM_error_exit(1001)

NODGRP_NAME => grp_data%node_grp%enum_grp_name
NODGRP_ITEM => grp_data%node_grp%enum_grp_node
NODGRP_INDEX=> grp_data%node_grp%enum_grp_index

NODGRPtot = grp_data%node_grp%n_enum_grp

return
end subroutine LOAD_MESH
```

LOAD_MESH（2/2）
ポインタのコピー

```
allocate (ICELNOD(ICELTOT,8))
do icel= 1, ICELTOT
  ICELNOD(icel,1)= local_mesh%ptr_elem(8*icel-7)
  ICELNOD(icel,2)= local_mesh%ptr_elem(8*icel-6)
  ICELNOD(icel,3)= local_mesh%ptr_elem(8*icel-5)
  ICELNOD(icel,4)= local_mesh%ptr_elem(8*icel-4)
  ICELNOD(icel,5)= local_mesh%ptr_elem(8*icel-3)
  ICELNOD(icel,6)= local_mesh%ptr_elem(8*icel-2)
  ICELNOD(icel,7)= local_mesh%ptr_elem(8*icel-1)
  ICELNOD(icel,8)= local_mesh%ptr_elem(8*icel-0)
enddo

if (N.le.0)       call ppohFVM_error_exit(1001)
if (ICELTOT.le.0) call ppohFVM_error_exit(1001)

NODGRP_NAME => grp_data%node_grp%enum_grp_name
NODGRP_ITEM => grp_data%node_grp%enum_grp_node
NODGRP_INDEX=> grp_data%node_grp%enum_grp_index

NODGRPtot = grp_data%node_grp%n_enum_grp

return
end subroutine LOAD_MESH
```

Global変数表：pfem_util.f（1/4）

変数名	種別	サイズ	I/O	内　容
N, NP	I		I	節点数（N：内点，NP：内点＋外点） **local_mesh%n_internal,local_mesh%n_node**
ICELTOT	I		I	要素数 local_mesh%n_elem
NODGRPtot	I		I	節点グループ数 grp_data%node_grp%n_enum_grp
XYZ	R	(NP, 3)	I	節点座標 local_mesh%node
ICELNOD	I	(ICELTOT, 8)	I	要素コネクティビティ local_mesh%ptr_elem
NODGRP_INDEX	I	(0:NODGRPtot)	I	各節点グループに含まれる節点数（累積） grp_data%node_grp%enum_grp_index
NODGRP_ITEM	I	(NODGRP_INDEX(NODGRPTOT)	I	節点グループに含まれる節点 grp_data%node_grp%enum_grp_node
NODGRP_NAME	C80	(NODGRPTOT)	I	節点グループ名 grp_data%node_grp%n_enum_grp
NLU	I		O	各節点非対角成分数　matrix_info%NLU
NPLU	I		O	非対角成分総数　matrix_info%NPLU
D	R	(NP)	O	全体行列：対角ブロック　matrix_info%D
B, X	R	(NP)	O	右辺ベクトル，未知数ベクトル matrix_info%RHS, matrix_info%X

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 matrix_info%AMAT
index	I	(0:NP)	O	全体行列：非零非対角成分数 matrix_info%index
item	I	(NPLU)	O	全体行列：非零非対角成分（列番号） matrix_info%item
INLU	I	(NP)	O	各節点の非零非対角成分数 matrix_info%INLU
IALU	I	(NP, NLU)	O	各節点の非零非対角成分数（列番号） matrix_info%IALU
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 solver_info%ITER, solver_info%ITERactual
RESID	R		I	打ち切り誤差（1.e-8に設定） solver_info%RESID

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）

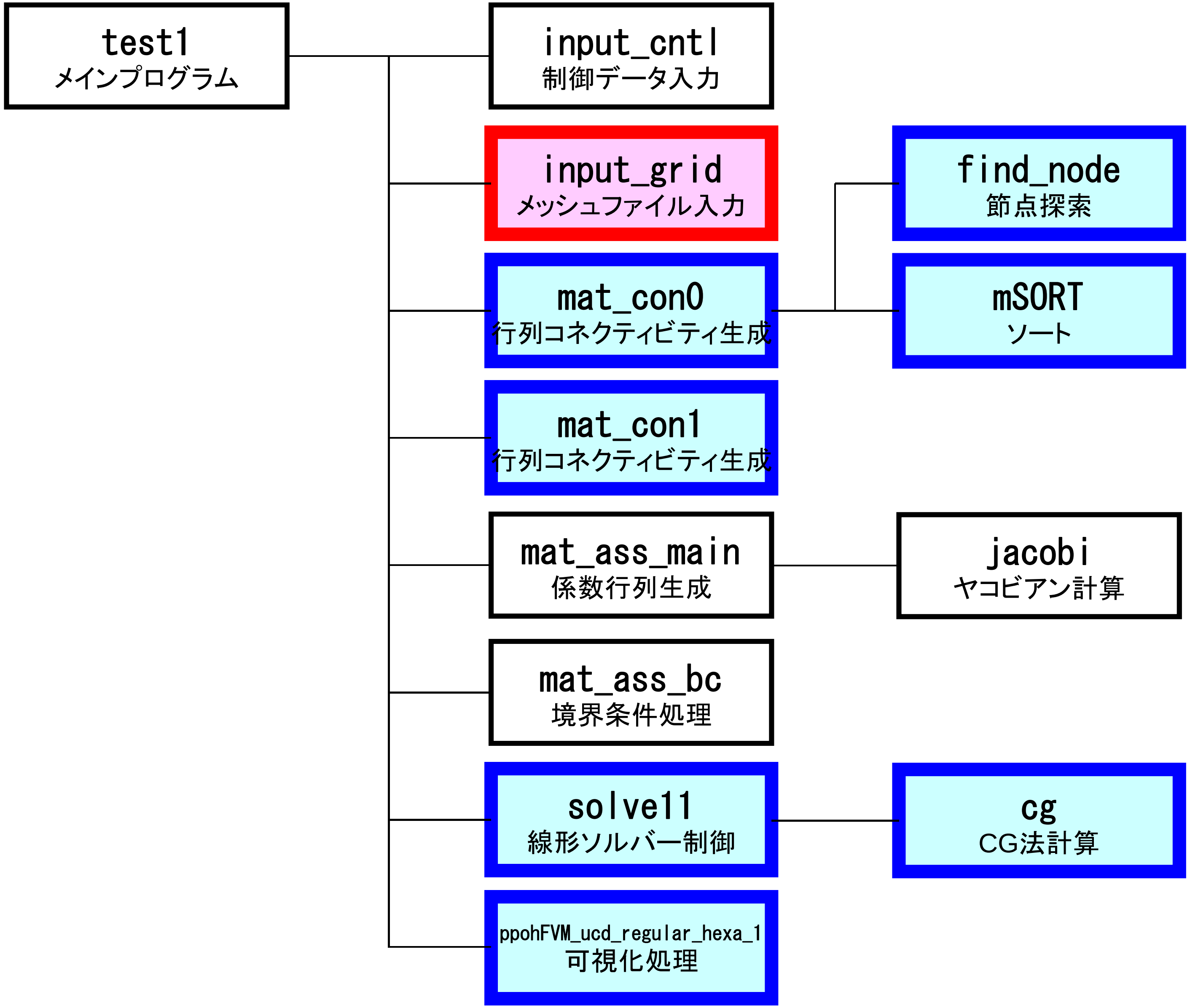
変数名	種別	サイズ	I/O	内　容
PETOT	I		O	領域数（MPIプロセス数） comm_info%PETOT
my_rank	I		O	MPIプロセス番号 comm_info%my_rank
errno	I		O	エラーフラグ
NEIBPETOT	I		I	隣接領域数 comm_info%n_neighbor_pe
NEIBPE	I	(NEIBPETOT)	I	隣接領域番号 comm_info%neighbor_pe
IMPORT_INDEX EXPORT_INEDX	I	(0:NEIBPETOT)	I	送信，受信テーブルのサイズ（一次元圧縮配列） comm_info%import_index,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

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
  - メッシュファイル読み込み（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）

全体処理: test1.f（2/2）

```
!C
!C +--------------------------------+
!C | Matrix Connectivity/Assembling |
!C +--------------------------------+
!C===
      call ppohFVM_mat_con (local_mesh, comm_info, matrix_info)
      call MAT_ASS_MAIN (local_mesh, matrix_info)
      call MAT_ASS_BC   (local_mesh, matrix_info)
!C===

!C
!C +--------+
!C | SOLVER |
!C +--------+
!C===
      call ppohFVM_solver11 (local_mesh, comm_info, matrix_info, solver_info)
!C===

!C
!C +--------+
!C | OUTPUT |
!C +--------+
!C===
      call ppohFVM_ucd_regular_hexa_1 (local_mesh, comm_info, matrix_info, vis_info)
!C===

      call ppohFVM_Finalize (comm_info)

      end program heat3Dp
```


test1
メインプログラム
input_cntl
制御データ入力
input_grid
メッシュファイル入力
mat_con0
行列コネクティビティ生成
mat_con1
行列コネクティビティ生成
mat_ass_main
係数行列生成
mat_ass_bc
境界条件処理
solve11
線形ソルバー制御
ppohFVM_ucd_regular_hexa_1
可視化処理
find_node
節点探索
mSORT
ソート
jacobi
ヤコビアン計算
cg
CG法計算

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

ppohFVM_mat_con（1/4）

```
      subroutine ppohFVM_mat_con (local_mesh, comm_info, matrix_info)

      use ppohFVM_util
      use ppohFVM_util_matrix

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

      type (ppohFVM_local_mesh)  ::  local_mesh
      type (ppohFVM_comm_info)   ::   comm_info
      type (ppohFVM_matrix_info) :: matrix_info

      integer, dimension(1000) :: NCOL1, NCOL2

!C
!C +--------+
!C | INIT.  |
!C +--------+
!C===

      matrix_info%N = local_mesh%n_internal
      matrix_info%NP= local_mesh%n_node

      N = local_mesh%n_internal
      NP= local_mesh%n_node
!C===

      if (matrix_info%TYPE.eq.2) then
!C
!C +-----------------+
!C | 2: [A] with [D] |
!C +-----------------+
!C===
      matrix_info%NLU= 26
                  NLU= matrix_info%NLU
      allocate (matrix_info%INLU(NP), matrix_info%IALU(NP,NLU))

      matrix_info%INLU= 0
      matrix_info%IALU= 0
```

ppohFVM_mat_con（2/4）

```
!C
!C-- HEXA.
      do icel0= 1, local_mesh%n_ACThexa
        icel=local_mesh%ACThexa_id(icel0)
         iS0=local_mesh%index_elem(icel-1)

        in1= local_mesh%ptr_elem(iS0+1)
        in2= local_mesh%ptr_elem(iS0+2)
        in3= local_mesh%ptr_elem(iS0+3)
        in4= local_mesh%ptr_elem(iS0+4)
        in5= local_mesh%ptr_elem(iS0+5)
        in6= local_mesh%ptr_elem(iS0+6)
        in7= local_mesh%ptr_elem(iS0+7)
        in8= local_mesh%ptr_elem(iS0+8)

        call ppohFVM_FIND_TS_NODE_2 (in1,in2)
        call ppohFVM_FIND_TS_NODE_2 (in1,in3)
        call ppohFVM_FIND_TS_NODE_2 (in1,in4)
        call ppohFVM_FIND_TS_NODE_2 (in1,in5)
        call ppohFVM_FIND_TS_NODE_2 (in1,in6)
        call ppohFVM_FIND_TS_NODE_2 (in1,in7)
        call ppohFVM_FIND_TS_NODE_2 (in1,in8)

        call ppohFVM_FIND_TS_NODE_2 (in2,in1)
        call ppohFVM_FIND_TS_NODE_2 (in2,in3)
        call ppohFVM_FIND_TS_NODE_2 (in2,in4)
        call ppohFVM_FIND_TS_NODE_2 (in2,in5)
        call ppohFVM_FIND_TS_NODE_2 (in2,in6)
        call ppohFVM_FIND_TS_NODE_2 (in2,in7)
        call ppohFVM_FIND_TS_NODE_2 (in2,in8)

        call ppohFVM_FIND_TS_NODE_2 (in3,in1)
...
        call ppohFVM_FIND_TS_NODE_2 (in8,in7)
      enddo
!C===
      endif
```

ppohFVM_mat_con（3/4）

```
!C
!C +----------+
!C | SORTING |
!C +----------+
!C===
      if (matrix_info%TYPE.eq.2) then
      do in= 1, N
        NN= matrix_info%INLU(in)
        do k= 1, NN
          NCOL1(k)= matrix_info%IALU(in,k)
        enddo
          call ppohFVM_mSORT (NCOL1, NCOL2, NN)
        do k= NN, 1, -1
          matrix_info%IALU(in,NN-k+1)= NCOL1(NCOL2(k))
        enddo
      enddo
      endif
!C===
```

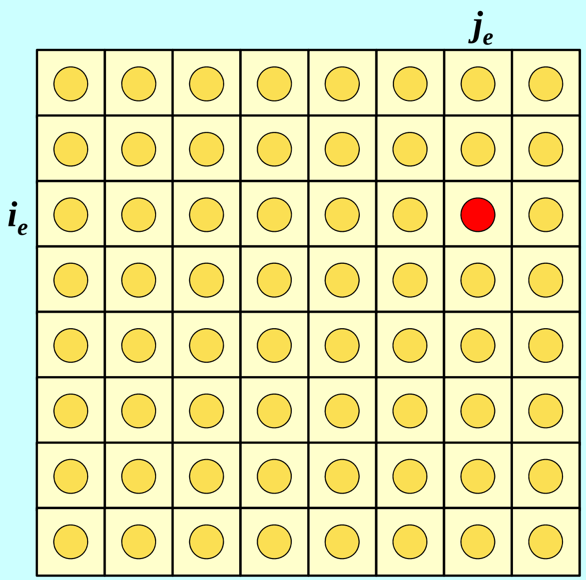
ppohFVM_mat_con（4/4）

```
!C
!C +------+
!C | CRS |
!C +------+
!C===
      allocate (matrix_info%index(0:NP))
      matrix_info%index= 0

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

      matrix_info%NPLU= matrix_info%index(NP)

      allocate (matrix_info%item(matrix_info%NPLU))

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

      deallocate (matrix_info%INLU, matrix_info%IALU)

      contains

      subroutine ppohFVM_FIND_TS_NODE_2 (ip1,ip2)

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

        end subroutine ppohFVM_FIND_TS_NODE_2
      end subroutine ppohFVM_mat_con
```

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
```



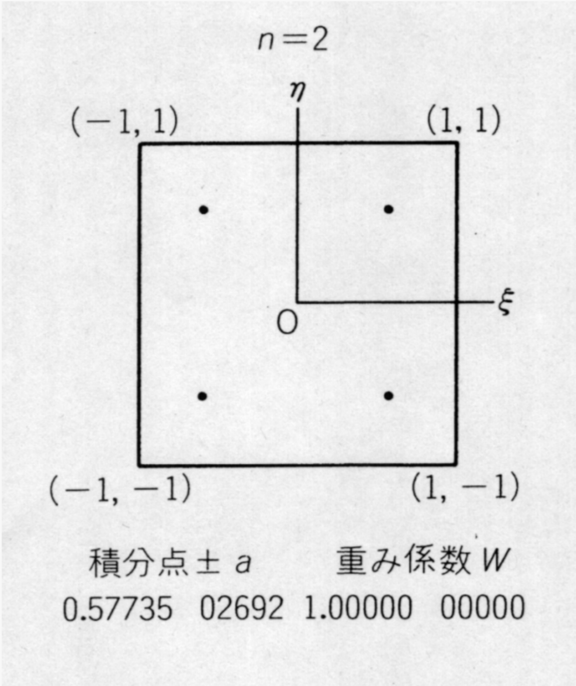
係数行列：MAT_ASS_MAIN（1/6）

```
subroutine MAT_ASS_MAIN (local_mesh, matrix_info)

use ppohFVM_util
use ppohFVM_util_matrix
use pfem_util
implicit REAL*8 (A-H,O-Z)

type (ppohFVM_local_mesh)  ::  local_mesh
type (ppohFVM_matrix_info) :: matrix_info

integer(kind=kint), dimension(  8) :: nodLOCAL

NPLU= matrix_info%NPLU

allocate (matrix_info%AMAT(NPLU), matrix_info%X(NP))
allocate (matrix_info%RHS(NP), matrix_info%D(NP))

matrix_info%AMAT= 0.d0
matrix_info%RHS = 0.d0
matrix_info%X   = 0.d0
matrix_info%D   = 0.d0

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）

```
subroutine MAT_ASS_MAIN (local_mesh, matrix_info)

use ppohFVM_util
use ppohFVM_util_matrix
use pfem_util
implicit REAL*8 (A-H,O-Z)

type (ppohFVM_local_mesh)  ::  local_mesh
type (ppohFVM_matrix_info) :: matrix_info

integer(kind=kint), dimension(  8) :: nodLOCAL

NPLU= matrix_info%NPLU

allocate (matrix_info%AMAT(NPLU), matrix_info%X(NP))
allocate (matrix_info%RHS(NP), matrix_info%D(NP))

matrix_info%AMAT= 0.d0
matrix_info%RHS = 0.d0
matrix_info%X   = 0.d0
matrix_info%D   = 0.d0

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)= O8th * QM1 * EM1 * TM1
        SHAPE(ip,jp,kp,2)= O8th * QP1 * EM1 * TM1
        SHAPE(ip,jp,kp,3)= O8th * QP1 * EP1 * TM1
        SHAPE(ip,jp,kp,4)= O8th * QM1 * EP1 * TM1
        SHAPE(ip,jp,kp,5)= O8th * QM1 * EM1 * TP1
        SHAPE(ip,jp,kp,6)= O8th * QP1 * EM1 * TP1
        SHAPE(ip,jp,kp,7)= O8th * QP1 * EP1 * TP1
```

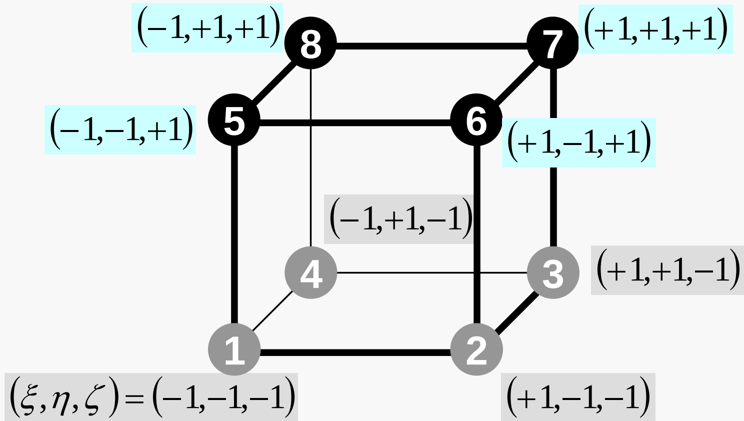
係数行列：MAT_ASS_MAIN（2/6）

```
!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)= O8th * QM1 * EM1 * TM1
        SHAPE(ip,jp,kp,2)= O8th * QP1 * EM1 * TM1
        SHAPE(ip,jp,kp,3)= O8th * QP1 * EP1 * TM1
        SHAPE(ip,jp,kp,4)= O8th * QM1 * EP1 * TM1
        SHAPE(ip,jp,kp,5)= O8th * QM1 * EM1 * TP1
        SHAPE(ip,jp,kp,6)= O8th * QP1 * EM1 * TP1
        SHAPE(ip,jp,kp,7)= O8th * QP1 * EP1 * TP1
```

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

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)= O8th * QM1 * EM1 * TM1
        SHAPE(ip,jp,kp,2)= O8th * QP1 * EM1 * TM1
        SHAPE(ip,jp,kp,3)= O8th * QP1 * EP1 * TM1
        SHAPE(ip,jp,kp,4)= O8th * QM1 * EP1 * TM1
        SHAPE(ip,jp,kp,5)= O8th * QM1 * EM1 * TP1
        SHAPE(ip,jp,kp,6)= O8th * QP1 * EM1 * TP1
        SHAPE(ip,jp,kp,7)= O8th * 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)= O8th * QM1 * EM1 * TM1
        SHAPE(ip,jp,kp,2)= O8th * QP1 * EM1 * TM1
        SHAPE(ip,jp,kp,3)= O8th * QP1 * EP1 * TM1
        SHAPE(ip,jp,kp,4)= O8th * QM1 * EP1 * TM1
        SHAPE(ip,jp,kp,5)= O8th * QM1 * EM1 * TP1
        SHAPE(ip,jp,kp,6)= O8th * QP1 * EM1 * TP1
        SHAPE(ip,jp,kp,7)= O8th * 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)= - O8th * EM1 * TM1
      PNQ(jp,kp,2)= + O8th * EM1 * TM1
      PNQ(jp,kp,3)= + O8th * EP1 * TM1
      PNQ(jp,kp,4)= - O8th * EP1 * TM1
      PNQ(jp,kp,5)= - O8th * EM1 * TP1
      PNQ(jp,kp,6)= + O8th * EM1 * TP1
      PNQ(jp,kp,7)= + O8th * EP1 * TP1
      PNQ(jp,kp,8)= - O8th * EP1 * TP1
      PNE(ip,kp,1)= - O8th * QM1 * TM1
      PNE(ip,kp,2)= - O8th * QP1 * TM1
      PNE(ip,kp,3)= + O8th * QP1 * TM1
      PNE(ip,kp,4)= + O8th * QM1 * TM1
      PNE(ip,kp,5)= - O8th * QM1 * TP1
      PNE(ip,kp,6)= - O8th * QP1 * TP1
      PNE(ip,kp,7)= + O8th * QP1 * TP1
      PNE(ip,kp,8)= + O8th * QM1 * TP1
      PNT(ip,jp,1)= - O8th * QM1 * EM1
      PNT(ip,jp,2)= - O8th * QP1 * EM1
      PNT(ip,jp,3)= - O8th * QP1 * EP1
      PNT(ip,jp,4)= - O8th * QM1 * EP1
      PNT(ip,jp,5)= + O8th * QM1 * EM1
      PNT(ip,jp,6)= + O8th * QP1 * EM1
      PNT(ip,jp,7)= + O8th * QP1 * EP1
      PNT(ip,jp,8)= + O8th * QM1 * EP1
    enddo
    enddo
    enddo

    do icel= 1, ICELTOT
      COND0= 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)
```

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

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

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

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

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

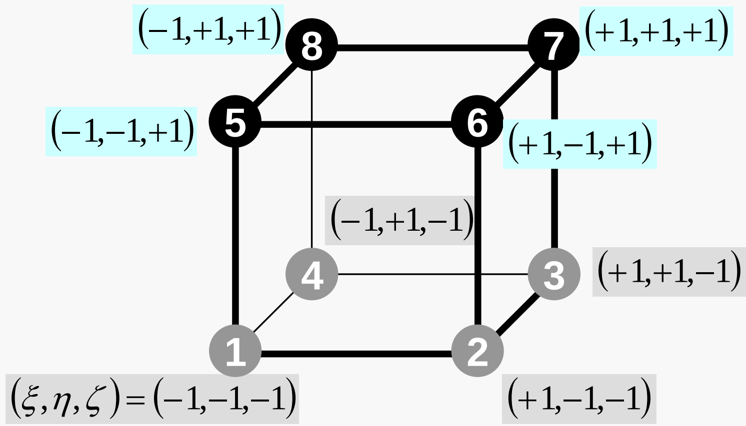
$$\frac{\partial N_3}{\partial \xi}(\xi_i, \eta_j, \zeta_k) = +\frac{1}{8}(1+\eta_j)(1-\zeta_k)$$

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

(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）

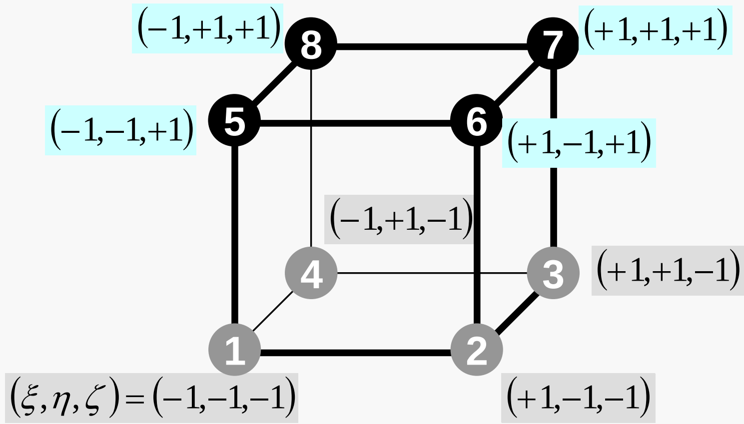
```
          PNQ(jp,kp,1)= - O8th * EM1 * TM1
          PNQ(jp,kp,2)= + O8th * EM1 * TM1
          PNQ(jp,kp,3)= + O8th * EP1 * TM1
          PNQ(jp,kp,4)= - O8th * EP1 * TM1
          PNQ(jp,kp,5)= - O8th * EM1 * TP1
          PNQ(jp,kp,6)= + O8th * EM1 * TP1
          PNQ(jp,kp,7)= + O8th * EP1 * TP1
          PNQ(jp,kp,8)= - O8th * EP1 * TP1
          PNE(ip,kp,1)= - O8th * QM1 * TM1
          PNE(ip,kp,2)= - O8th * QP1 * TM1
          PNE(ip,kp,3)= + O8th * QP1 * TM1
          PNE(ip,kp,4)= + O8th * QM1 * TM1
          PNE(ip,kp,5)= - O8th * QM1 * TP1
          PNE(ip,kp,6)= - O8th * QP1 * TP1
          PNE(ip,kp,7)= + O8th * QP1 * TP1
          PNE(ip,kp,8)= + O8th * QM1 * TP1
          PNT(ip,jp,1)= - O8th * QM1 * EM1
          PNT(ip,jp,2)= - O8th * QP1 * EM1
          PNT(ip,jp,3)= - O8th * QP1 * EP1
          PNT(ip,jp,4)= - O8th * QM1 * EP1
          PNT(ip,jp,5)= + O8th * QM1 * EM1
          PNT(ip,jp,6)= + O8th * QP1 * EM1
          PNT(ip,jp,7)= + O8th * QP1 * EP1
          PNT(ip,jp,8)= + O8th * 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



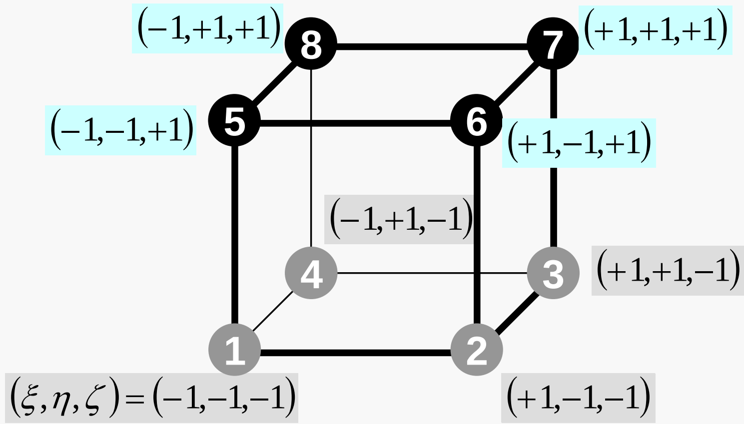
係数行列：MAT_ASS_MAIN（4/6）

```
      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

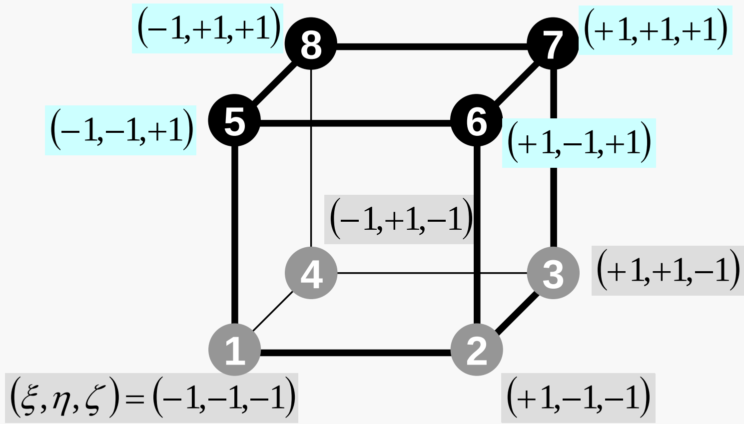


係数行列：MAT_ASS_MAIN（4/6）

```
          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,
     &                 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



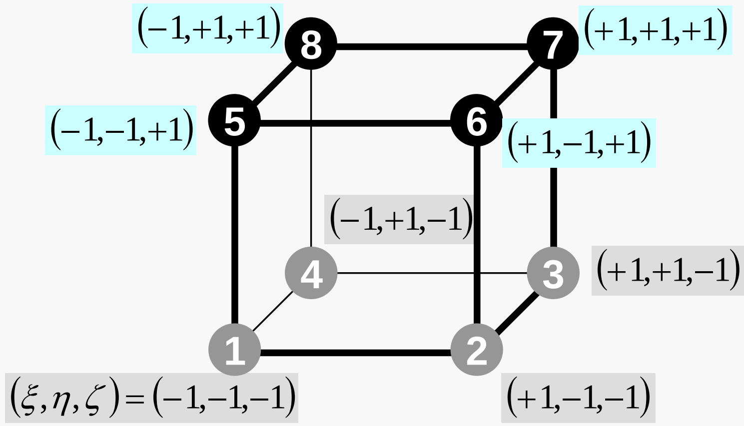
$$\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(\lambda\frac{\partial T}{\partial z}\right)+\dot{Q}(x,y,z)=0$$

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

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= 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 )
```



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

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

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

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）

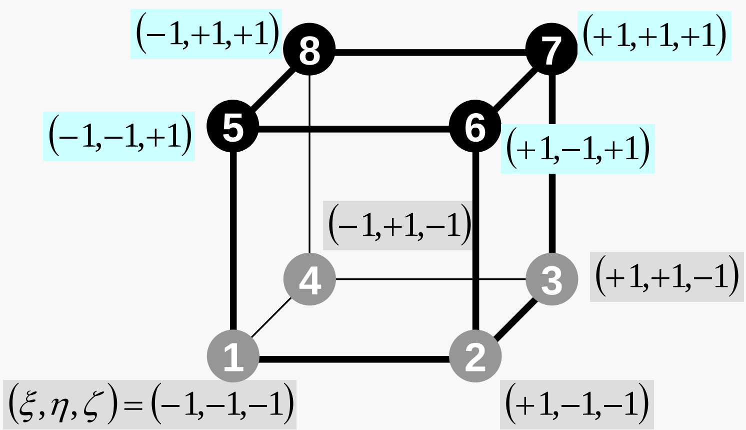
```
!C
!C== CONSTRUCT the GLOBAL MATRIX

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

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

全体行列の非対角成分

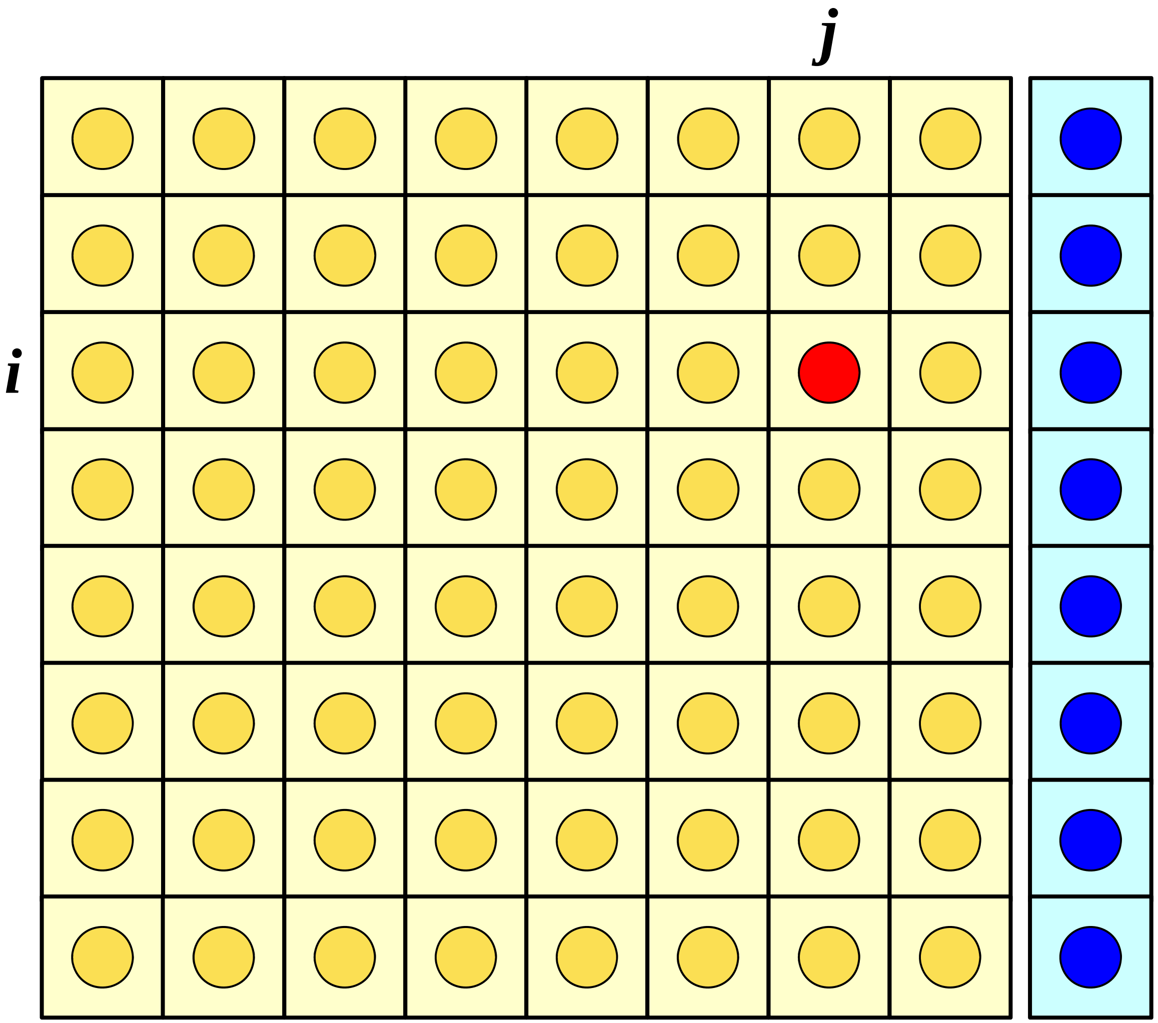
$A_{ip,jp}$

kk: matrix_info%itemにおけるアドレス

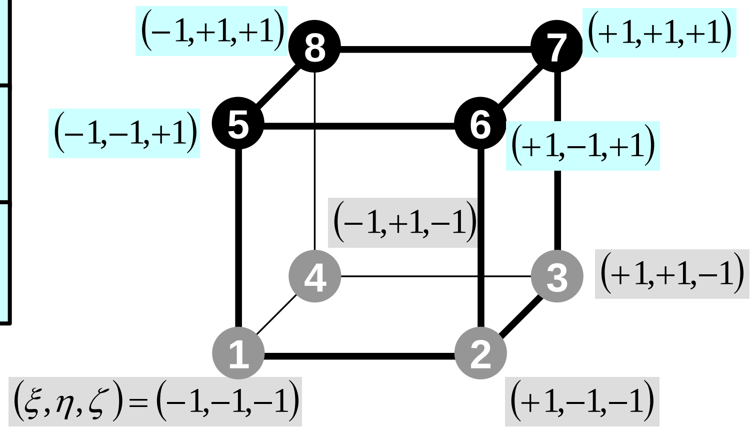
ip= nodLOCAL(ie)
jp= nodLOCAL(je)



要素マトリクス：8×8行列



$$\left[k_{ij}\right] \quad (i, j = 1 \ldots 8)$$

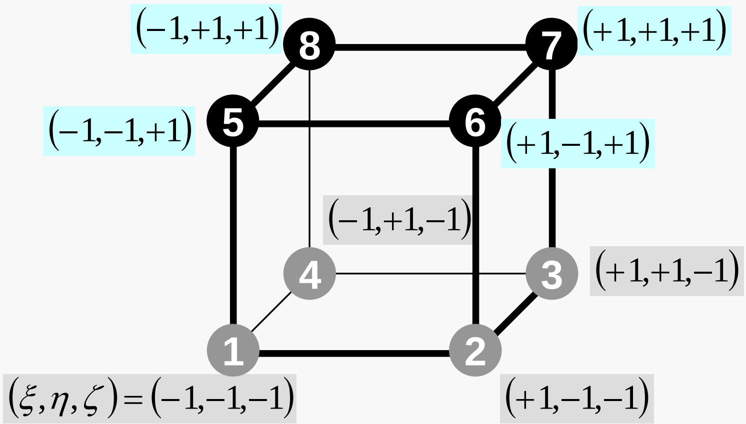


係数行列：MAT_ASS_MAIN（5/6）

```
!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= matrix_info%index(ip-1) + 1
            iiE= matrix_info%index(ip  )
            do k= iiS, iiE
              if (matrix_info%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:matrix_info%itemにおけるアドレス



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 * COND0 *
     &                 (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
        matrix_info%D  (ip)= matrix_info%D  (ip) + COEFij
        matrix_info%RHS(ip)= matrix_info%RHS(ip) + QV0*QVC
       else
        matrix_info%AMAT(kk)= matrix_info%AMAT(kk) + COEFij
      endif
    enddo
    enddo
  enddo

  return
  end
```

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

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
          matrix_info%D  (ip)= matrix_info%D  (ip) +
          matrix_info%RHS(ip)= matrix_info%RHS(ip) +
         else
          matrix_info%AMAT(kk)= matrix_info%AMAT(kk) + COEFij
        endif
      enddo
      enddo
    enddo

    return
    end
```

$$I = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} f(\xi,\eta,\zeta)\, d\xi d\eta d\zeta$$

$$= \sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{k=1}^{N}\left[W_i \cdot W_j \cdot W_k \cdot f(\xi_i,\eta_j,\zeta_k)\right]$$

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

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 * COND0 *
     &                   (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
          matrix_info%D  (ip)= matrix_info%D  (ip) +
          matrix_info%RHS(ip)= matrix_info%RHS(ip) +
         else
          matrix_info%AMAT(kk)= matrix_info%AMAT(kk) + COEFij
        endif
      enddo
      enddo
    enddo

    return
    end
```

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

$$I = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} f(\xi,\eta,\zeta)\, d\xi d\eta d\zeta$$

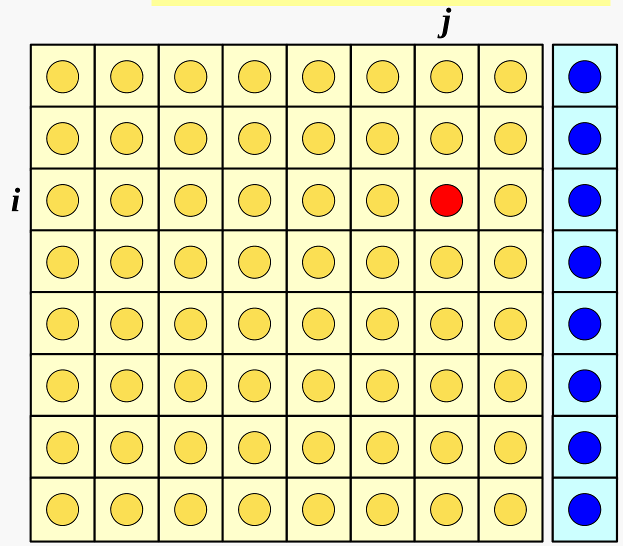
$$= \sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{k=1}^{N}\left[W_i \cdot W_j \cdot W_k \cdot f(\xi_i,\eta_j,\zeta_k)\right]$$

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

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
            matrix_info%D   (ip)= matrix_info%D   (ip) + COEFij
            matrix_info%RHS(ip)= matrix_info%RHS(ip) + QVO*QVC
           else
            matrix_info%AMAT(kk)= matrix_info%AMAT(kk) + COEFij
          endif
        enddo
        enddo
      enddo

      return
      end
```

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



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 * COND0 *
     &                (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
          matrix_info%D  (ip)= matrix_info%D  (ip) + COEFij
          matrix_info%RHS(ip)= matrix_info%RHS(ip) + QV0*QVC
         else
          matrix_info%AMAT(kk)= matrix_info%AMAT(kk) + COEFij
        endif
      enddo
      enddo
    enddo

    return
    end
```

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

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

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

$$QVC=|x_C+y_C|$$

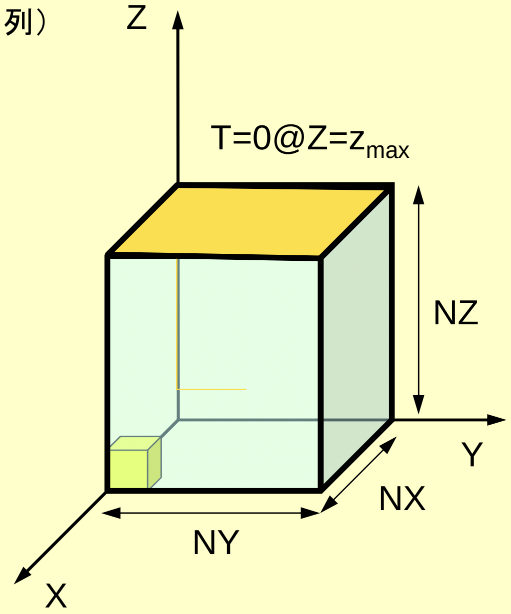
$$QV0=\int_V QVOL[N]^T dV$$

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

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

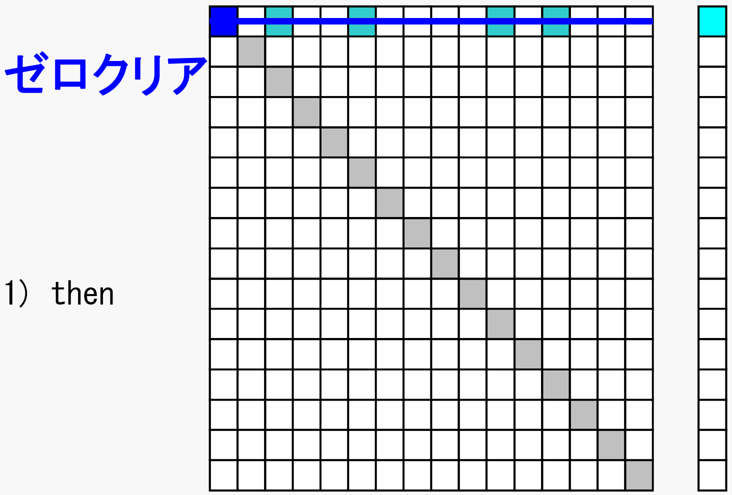
境界条件：MAT_ASS_BC（2/2）

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

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

      do in= 1, NP
        iS= matrix_info%index(in-1) + 1
        iE= matrix_info%index(in  )
        do k= iS, iE
          if (IWKX(matrix_info% item(k),1).eq.1) then
            matrix_info%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
          matrix_info%RHS(in)= 0.d0
          matrix_info%D  (in)= 1.d0

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

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

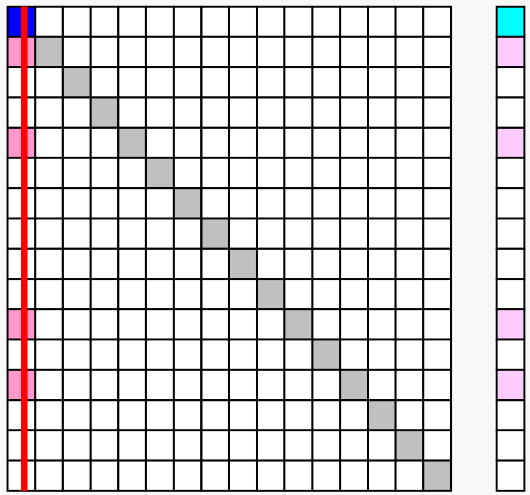
IWKX(in,1)=1となる節点に対して
対角成分=1，右辺=0，非零対角成分=0



境界条件：MAT_ASS_BC（2/2）

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

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

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

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
  - メッシュファイル読み込み（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）**

全体処理: test1.f（2/2）

```
!C
!C +-------------------------------+
!C | Matrix Connectivity/Assembling |
!C +-------------------------------+
!C===
      call ppohFVM_mat_con (local_mesh, comm_info, matrix_info)
      call MAT_ASS_MAIN (local_mesh, matrix_info)
      call MAT_ASS_BC   (local_mesh, matrix_info)
!C===

!C
!C +--------+
!C | SOLVER |
!C +--------+
!C===
      call ppohFVM_solver11 (local_mesh, comm_info, matrix_info, solver_info)
!C===

!C
!C +--------+
!C | OUTPUT |
!C +--------+
!C===
      call ppohFVM_ucd_regular_hexa_1 (local_mesh, comm_info, matrix_info, vis_info)
!C===

      call ppohFVM_Finalize (comm_info)

      end program heat3Dp
```

ppohFVM_solver11

```
      subroutine ppohFVM_solver11 (local_mesh, comm_info, matrix_info, solver_info)

      use ppohFVM_util
      use ppohFVM_util_matrix

        implicit REAL*8 (A-H,O-Z)
        type (ppohFVM_local_mesh) :: local_mesh
        type (ppohFVM_comm_info)  ::  comm_info
        type (ppohFVM_matrix_info) :: matrix_info
        type (ppohFVM_solver_info) :: solver_info

        character(len=ppohFVM_name_len) :: BUF

        call ppohFVM_CG_2_1_00                                          &
     &      ( comm_info, matrix_info, solver_info)


      end subroutine ppohFVM_solver11
```

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

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

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

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

- `solve [M]z(i-1)= r(i-1)` ($[M]z^{(i-1)}=r^{(i-1)}$) という場合に逆行列を簡単に求めることができる。

ppohFVM_CG_2_1_00（1/6）

```
      subroutine ppohFVM_CG_2_1_00 ( comm_info, matrix_info, solver_info)

      use ppohFVM_util
      use ppohFVM_util_matrix
      implicit REAL*8(A-H,O-Z)
      type (ppohFVM_comm_info)  ::  comm_info
      type (ppohFVM_matrix_info) :: matrix_info
      type (ppohFVM_solver_info) :: solver_info

      real(kind=ppohFVM_kreal), dimension(:,:),  allocatable:: WW

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

      integer(kind=ppohFVM_kint ) :: MAXIT, N, NP
      real   (kind=ppohFVM_kreal) :: TOL

!C
!C +--------+
!C | INIT. |
!C +--------+
!C===
      N = matrix_info%N
      NP= matrix_info%NP

      solver_info%COMMtime= 0.d0
      solver_info%COMPtime= 0.d0

      solver_info%ERROR= 0

      allocate (WW(NP,4))

      MAXIT  = solver_info%ITER
       TOL   = solver_info%RESID

      matrix_info%X = 0.d0
!C===
```

ppohFVM_CG_2_1_00 (2/6)

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

!C
!C-- INTERFACE data EXCHANGE
      call ppohFVM_update_1_R (comm_info, matrix_info%X, NP, N)

      do j= 1, N
        WW(j,DD)= 1.d0/matrix_info%D(j)
        WVAL= matrix_info%RHS(j) - matrix_info%D(j)*matrix_info%X(j)
        do k= matrix_info%index(j-1)+1, matrix_info%index(j)
          i= matrix_info%item(k)
          if (i.gt.NP) write (*,*) comm_info%my_rank, j, i
         WVAL= WVAL - matrix_info%AMAT(k)*matrix_info%X(i)
        enddo
        WW(j,R)= WVAL
      enddo


      BNRM20= 0.d0
      do i= 1, N
        BNRM20= BNRM20 + matrix_info%RHS(i)**2
      enddo

      call ppohFVM_Allreduce_R (BNRM20, ppohFVM_sum)
      BNRM2= BNRM20

!      call MPI_Allreduce (BNRM20, BNRM2, 1, MPI_DOUBLE_PRECISION,
!     &                    MPI_SUM, MPI_COMM_WORLD, ierr)

      if (BNRM2.eq.0.d0) BNRM2= 1.d0
      ITER = 0
!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
  - $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
  - $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
- 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

ppohFVM_update_1_R

```
      subroutine ppohFVM_update_1_R (comm_info, VAL, n, n0)
      use ppohFVM_SR_r1

      implicit REAL*8 (A-H,O-Z)
      integer :: n, ierr
      real(kind=ppohFVM_kreal), dimension(n):: VAL
      type (ppohFVM_comm_info)      :: comm_info

      call ppohFVM_SEND_RECV_r1                                     &
     &   ( n, n0, comm_info%n_neighbor_pe, comm_info%neighbor_pe,   &
     &     comm_info%import_index, comm_info%import_item,           &
     &     comm_info%export_index, comm_info%export_item,           &
     &     comm_info%WS, comm_info%WR, VAL, comm_info%my_rank)

      end subroutine ppohFVM_update_1_R


      subroutine ppohFVM_update_2_R (comm_info, VAL, n, n0)
      use ppohFVM_SR_r2

      implicit REAL*8 (A-H,O-Z)
      integer :: n, ierr
      real(kind=ppohFVM_kreal), dimension(2,n):: VAL
      type (ppohFVM_comm_info)        :: comm_info

      call ppohFVM_SEND_RECV_r2                                     &
     &   ( n, n0, comm_info%n_neighbor_pe, comm_info%neighbor_pe,   &
     &     comm_info%import_index, comm_info%import_item,           &
     &     comm_info%export_index, comm_info%export_item,           &
     &     comm_info%WS2, comm_info%WR2, VAL, comm_info%my_rank)

      end subroutine ppohFVM_update_2_R
```

ppohFVM_SEND_RECV_r1（1/2）

```
      subroutine  ppohFVM_SEND_RECV_r1                                    &
     &              ( N, NO, NEIBPETOT,NEIBPE, IMPORTindex, IMPORTitem, &
     &                                         EXPORTindex, EXPORTitem, &
     &                WS, WR, X, my_rank)

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

      integer(kind=ppohFVM_kint )                , intent(in) :: N, NO
      integer(kind=ppohFVM_kint )                , intent(in) :: NEIBPETOT
      integer(kind=ppohFVM_kint ), pointer       , intent(in) :: NEIBPE      (:)
      integer(kind=ppohFVM_kint ), pointer       , intent(in) :: IMPORTindex(:), IMPORTitem(:)
      integer(kind=ppohFVM_kint ), pointer       , intent(in) :: EXPORTindex(:), EXPORTitem(:)
      real   (kind=ppohFVM_kreal), dimension(N)  , intent(inout):: WS
      real   (kind=ppohFVM_kreal), dimension(N)  , intent(inout):: WR
      real   (kind=ppohFVM_kreal), dimension(N)  , intent(inout):: X
      integer                            , intent(in)   :: my_rank

      integer(kind=ppohFVM_kint ), dimension(:,:), save, allocatable :: sta1
      integer(kind=ppohFVM_kint ), dimension(:  ), save, allocatable :: req1

      integer(kind=ppohFVM_kint ), save :: NFLAG
      data NFLAG/0/

!C
!C-- INIT.
      if (NFLAG.eq.0) then
        allocate (sta1(MPI_STATUS_SIZE,2*NEIBPETOT))
        allocate (req1(2*NEIBPETOT))
        NFLAG= 1
      endif
```

ppohFVM_SEND_RECV_r1（2/2）

```
!C
!C-- SEND
      do neib= 1, NEIBPETOT
        istart= EXPORTindex(neib-1)
        inum  = EXPORTindex(neib  ) - istart
!$omp parallel do private (k)
        do k= istart+1, istart+inum
           WS(k)= X(EXPORTitem(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= N0 + IMPORTindex(neib-1)
        inum  = IMPORTindex(neib) - IMPORTindex(neib-1)
        call MPI_Irecv (X(istart+1), inum, MPI_DOUBLE_PRECISION,        &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD,                &
     &                  req1(neib+NEIBPETOT), ierr)
      enddo

      call MPI_Waitall (2*NEIBPETOT, req1, sta1, ierr)


      end subroutine ppohFVM_SEND_RECV_r1
      end module     ppohFVM_SR_r1
```

ppohFVM_Allreduce_R

```
      subroutine ppohFVM_Allreduce_R ( VAL, ntag )

      use ppohFVM_util
      implicit REAL*8 (A-H,O-Z)
      integer :: ntag, ierr
      real(kind=ppohFVM_kreal) :: VAL, VALM

      if (ntag .eq. ppohFVM_sum) then
        call MPI_Allreduce                                          &
     &       (VAL, VALM, 1, MPI_DOUBLE_PRECISION, MPI_SUM,          &
     &        MPI_COMM_WORLD, ierr)
      endif

      if (ntag .eq. ppohFVM_max) then
        call MPI_Allreduce                                          &
     &       (VAL, VALM, 1, MPI_DOUBLE_PRECISION, MPI_MAX,          &
     &        MPI_COMM_WORLD, ierr)
      endif

      if (ntag .eq. ppohFVM_min) then
        call MPI_Allreduce                                          &
     &       (VAL, VALM, 1, MPI_DOUBLE_PRECISION, MPI_MIN,          &
     &        MPI_COMM_WORLD, ierr)
      endif

      VAL= VALM

      end subroutine ppohFVM_Allreduce_R
```

ppohFVM_Allreduce_RV

```
      subroutine ppohFVM_Allreduce_RV ( VAL, n, ntag )

      use ppohFVM_util
      implicit REAL*8 (A-H,O-Z)
      integer :: n, ntag, ierr
      real(kind=ppohFVM_kreal), dimension(n)              :: VAL
      real(kind=ppohFVM_kreal), dimension(:), allocatable :: VALM

      allocate (VALM(n))
      if (ntag .eq. ppohFVM_sum) then
        call MPI_Allreduce                                              &
     &       (VAL, VALM, n, MPI_DOUBLE_PRECISION, MPI_SUM,              &
     &        MPI_COMM_WORLD, ierr)
      endif

      if (ntag .eq. ppohFVM_max) then
        call MPI_Allreduce                                              &
     &       (VAL, VALM, n, MPI_DOUBLE_PRECISION, MPI_MAX,              &
     &        MPI_COMM_WORLD, ierr)
      endif

      if (ntag .eq. ppohFVM_min) then
        call MPI_Allreduce                                              &
     &       (VAL, VALM, n, MPI_DOUBLE_PRECISION, MPI_MIN,              &
     &        MPI_COMM_WORLD, ierr)
      endif

      VAL= VALM
      deallocate (VALM)

      end subroutine ppohFVM_Allreduce_RV
```

ppohFVM_CG_2_1_00（3/6）

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

!C
!C +---------------+
!C | {RHO}= {r}{z} |
!C +---------------+
!C===
        RHO0= 0.d0
        do i= 1, N
          RHO0= RHO0 + WW(i,R)*WW(i,Z)
        enddo

        call ppohFVM_Allreduce_R (RHO0, ppohFVM_sum)
        RHO= RHO0
!C===

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

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

ppohFVM_CG_2_1_00（4/6）

```
!C
!C +-------------+
!C | {q}= [A]{p} |
!C +-------------+
!C===
!C
!C-- INTERFACE data EXCHANGE
      call ppohFVM_update_1_R (comm_info, WW(1,P), NP, N)

      do j= 1, N
        WVAL= matrix_info%D(j)*WW(j,P)
        do k= matrix_info%index(j-1)+1, matrix_info%index(j)
           i= matrix_info%item(k)
          WVAL= WVAL + matrix_info%AMAT(k)*WW(i,P)
        enddo
        WW(j,Q)= WVAL
      enddo
!C===

!C
!C +---------------------+
!C | ALPHA= RHO / {p}{q} |
!C +---------------------+
!C===
      C10= 0.d0
      do i= 1, N
        C10= C10 + WW(i,P)*WW(i,Q)
      enddo

      call ppohFVM_Allreduce_R (C10, ppohFVM_sum)
      C1= C10
!       call MPI_Allreduce (C10, C1, 1, MPI_DOUBLE_PRECISION,
!     &                     MPI_SUM, MPI_COMM_WORLD, ierr)

      ALPHA= RHO / C1
!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
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
  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

ppohFVM_CG_2_1_00（5/6）

```
!C
!C +----------------------+
!C | {x}= {x} + ALPHA*{p} |
!C | {r}= {r} - ALPHA*{q} |
!C +----------------------+
!C===
      do i= 1, N
        matrix_info%X(i)= matrix_info%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 ppohFVM_Allreduce_R (DNRM20, ppohFVM_sum)
      DNRM2= DNRM20

      RESID= dsqrt(DNRM2/BNRM2)

        if ( RESID.le.TOL   ) exit
        if ( ITER .eq.MAXIT ) solver_info%ERROR= -300

        RHO1 = RHO
      enddo
!C===

   30 continue

!C
!C-- INTERFACE data EXCHANGE
      call ppohFVM_update_1_R (comm_info, matrix_info%X, NP, N)

      deallocate (WW)

      solver_info%RESID     = RESID
      solver_info%ITERactual= ITER

      end subroutine ppohFVM_CG_2_1_00
```

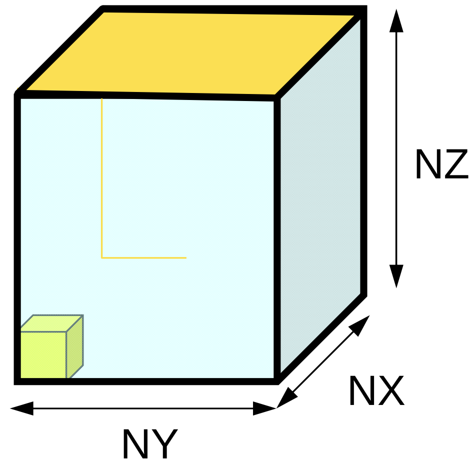
Compute $r^{(0)}= b-[A]x^{(0)}$
for **i= 1, 2, …**
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)} z^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1} p^{(i-1)}$
  endif
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$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

ppohFVM_CG_2_1_00（6/6）

```
!C
!C +-----------------------+
!C | {x}= {x} + ALPHA*{p} |
!C | {r}= {r} - ALPHA*{q} |
!C +-----------------------+
!C===
      do i= 1, N
        matrix_info%X(i)= matrix_info%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 ppohFVM_Allreduce_R (DNRM20, ppohFVM_sum)
      DNRM2= DNRM20

      RESID= dsqrt(DNRM2/BNRM2)

        if ( RESID.le.TOL   ) exit
        if ( ITER .eq.MAXIT ) solver_info%ERROR= -300

        RHO1 = RHO
      enddo
!C===

   30 continue

!C
!C-- INTERFACE data EXCHANGE
      call ppohFVM_update_1_R (comm_info, matrix_info%X, NP, N)

      deallocate (WW)

      solver_info%RESID     = RESID
      solver_info%ITERactual= ITER

      end subroutine ppohFVM_CG_2_1_00
```

簡易可視化方法

- 各領域が規則正しい直方体構造であることを仮定
  - 一様形状である必要はない
  - pmeshで生成されるようなメッシュ
- 最終的に出力するParaView用出力ファイルの全体のメッシュ数を規定
- 各部分領域（MPIプロセス）の従属変数分布から，各領域に割り当てる「可視化用」メッシュ数の決定
  - 八分木で領域ごとに可視化用メッシュ生成
  - 値の変化の多い領域にメッシュ数を多く割当
    - ルールは色々と検討する必要がある
- 各領域で生成した可視化用メッシュを集める



計算例

- 192×96×64節点（=1,179,648節点，1,143,135要素）
- 16～192コア
- Solver計算時間
- 可視化
  - 2,923節点，703要素（192コア） Movie
  - 1,216節点，576要素（16コア） Movie

```
../../pmesh/pcube HEADER
2000              ITER
1.0 1.0           COND, QVOL
1.0e-08           RESID
T                 MESH_ASCI
1000              N_MESH_VIS
```

core #	sec. (speed-up)
16（4x2x2）	9.42（16.0）
32（4x4x2）	4.80（31.4）
64（4x4x4）	2.56（58.9）
96（6x4x4）	1.85（81.5）
128（8x4x4）	1.38（109.）
192（8x6x4）	1.07（141.）

まとめ，演習

- ppOpen-APPL/FVM
  - MPIを知らなくても並列コードを作ることができる
    - ある程度オペレーションを知らないと無理だが，引数を覚えておくことからは少なくとも解放される
  - 機能拡張
    - 線形ソルバー，前処理手法
    - 様々なデータタイプ（X(3,N)，X(N,3)）
    - C言語，AMR，可視化
  - 仕様：今後若干変わる可能性あり（Ver.1.0まで）
- 演習
  - Strong Scaling：問題規模を固定して，コア数を変える
  - Weak Scaling：1コア当たりの問題規模固定
    - 反復回数変わる
  - Hybrid

利用可能なキュー

- 以下の2種類のキューを利用可能
- 1 Tofu（12ノード）を使える
  - **lecture**
    - 12ノード（192コア），15分，アカウント有効期間中利用可能（2月25日（水）1700まで）
    - 全教育ユーザーで共有
  - **tutorial**
    - 12ノード（192コア），15分，講義・演習実施時間帯
    - **lecture**よりは多くのジョブを投入可能（混み具合による）

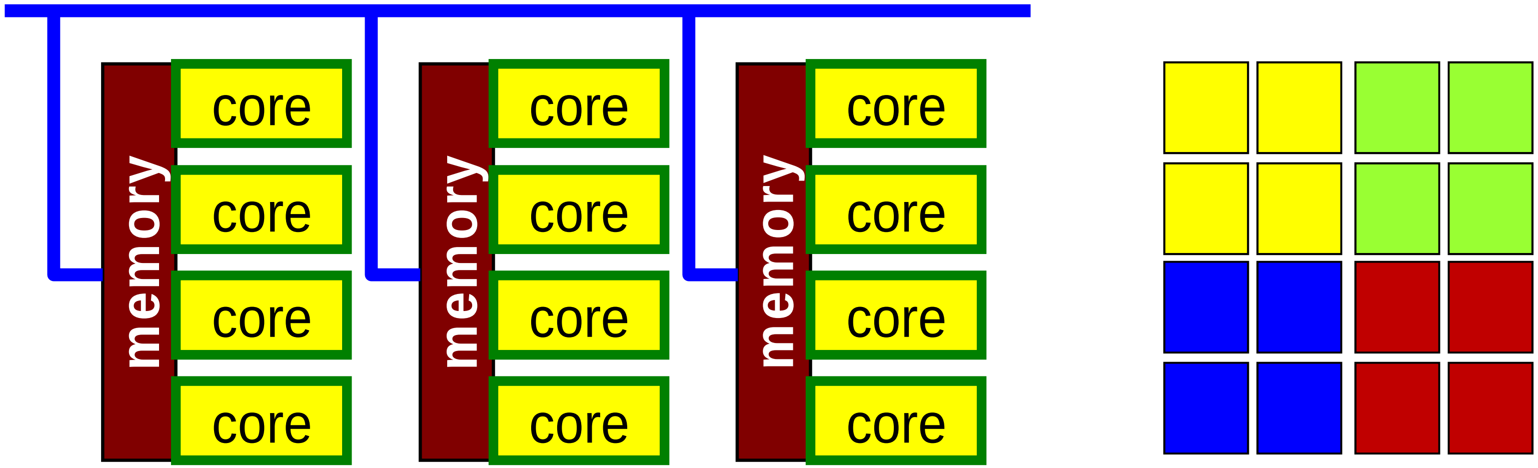
ジョブ投入，確認等

• ジョブの投入	pjsub スクリプト名
• ジョブの確認	pjstat
• ジョブの取り消し・強制終了	pjdel ジョブID
• キューの状態の確認	pjstat --rsc
• キューの詳細構成	pjstat --rsc –x
• 実行中のジョブ数	pjstat --rsc –b
• 同時実行・投入可能数	pjstat --limit

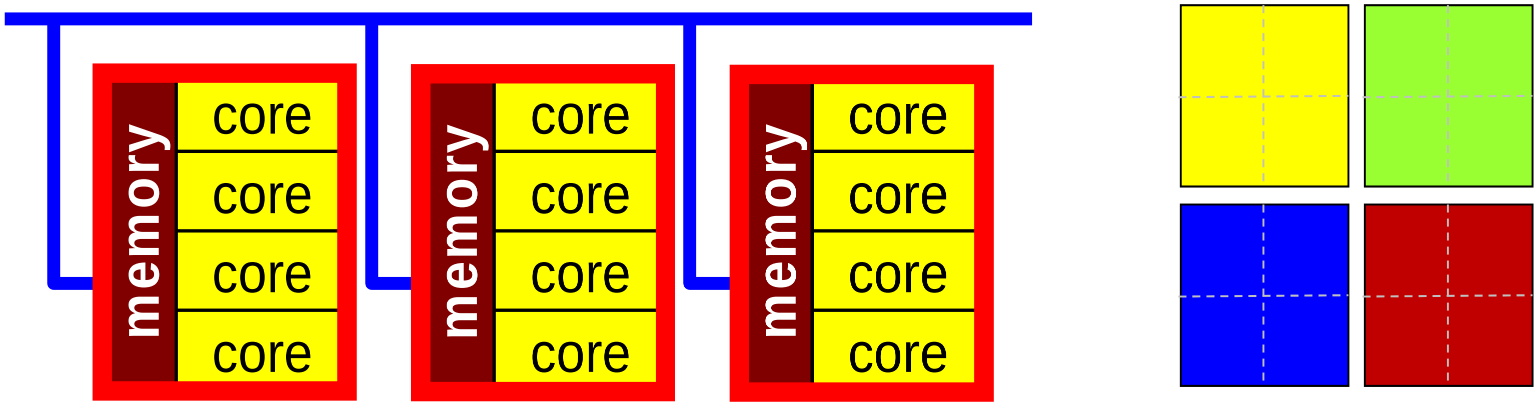
```
[z30088@oakleaf-fx-6 S2-ref]$ pjstat

Oakleaf-FX scheduled stop time: 2012/09/28(Fri) 09:00:00 (Remain: 31days 20:01:46)

JOB_ID     JOB_NAME   STATUS  PROJECT  RSCGROUP  START_DATE       ELAPSE       TOKEN NODE:COORD
334730     go.sh      RUNNING gt61     lecture   08/27 12:58:08   00:00:05       0.0 1
```

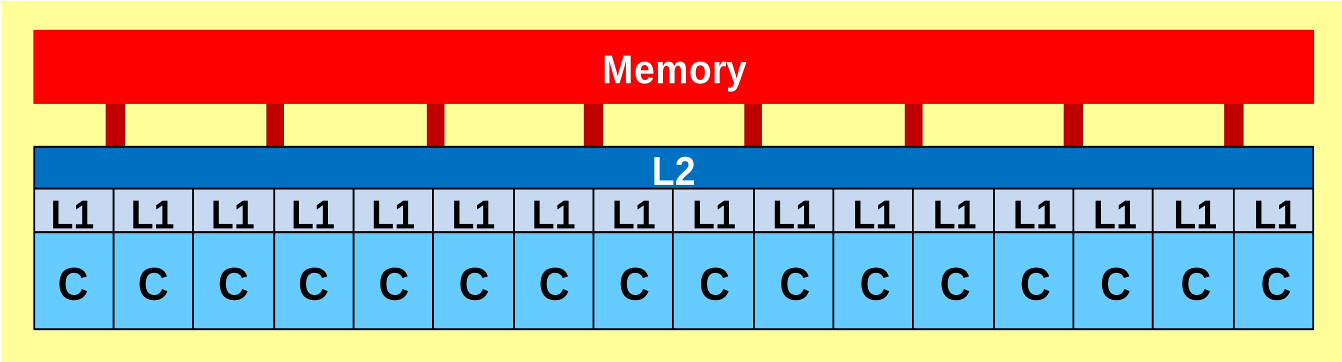
Flat MPI vs. Hybrid

Flat-MPI : Each Core -> Independent



Hybrid : Hierarchal Structure





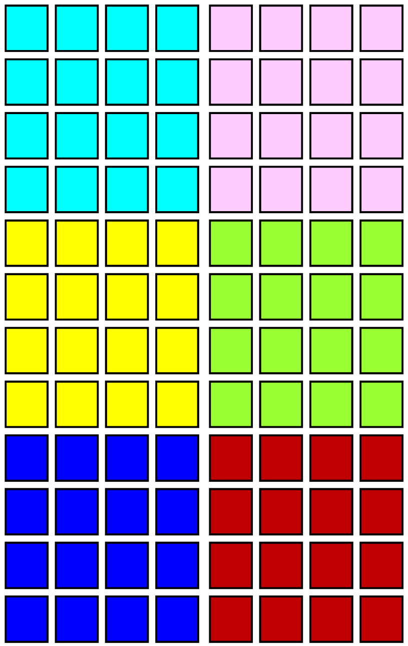
HB M x N

Number of OpenMP threads per a single MPI process

Number of MPI process per a single node

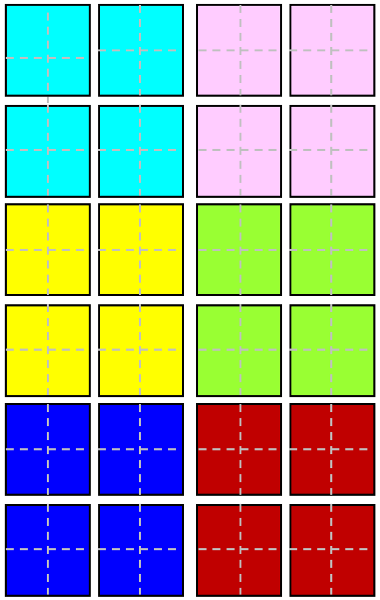
Size (and number) of local data changes according to parallel programming model

example: 6 nodes, 96 cores



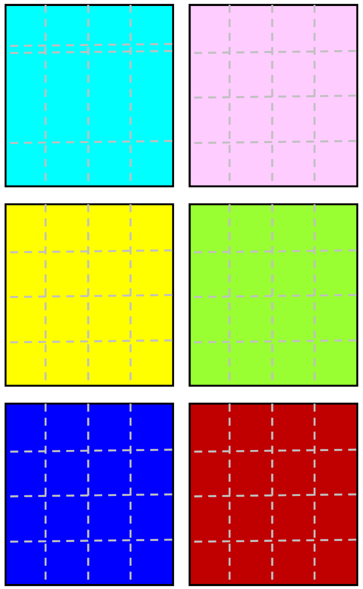
Flat MPI

```
128 192 64
  8  12  1
pcube
```



HB 4x4

```
128 192 64
  4   6  1
pcube
```



HB 16x1

```
128 192 64
  2   3  1
pcube
```

Batch Script (1/2) Env. Var.: OMP_NUM_THREADS

Flat MPI

```
#PJM -L "node=6"
#PJM -L "elapse=00:05:00"
#PJM -j
#PJM -L "rscgrp=tutorial“
#PJM -g “gt00“
#PJM -o "test.lst"
#PJM --mpi "proc=96"

mpiexec ./sol

rm wk.*
```

Hybrid 16×1

```
#!/bin/sh
#PJM -L "node=6"
#PJM -L "elapse=00:05:00"
#PJM -j
#PJM -L "rscgrp=tutorial“
#PJM -g “gt00“
#PJM -o "test.lst"
#PJM --mpi "proc=6"

export OMP_NUM_THREADS=16
mpiexec ./sol

rm wk.*
```

Batch Script (2/2) Env. Var.: OMP_NUM_THREADS

Hybrid 4×4

```
#!/bin/sh
#PJM -L "node=6"
#PJM -L "elapse=00:05:00"
#PJM -j
#PJM -L "rscgrp=tutorial“
#PJM -g “gt00“
#PJM -o "test.lst"
#PJM --mpi "proc=24"

export OMP_NUM_THREADS=4
mpiexec ./sol

rm wk.*
```

Hybrid 8×2

```
#!/bin/sh
#PJM -L "node=6"
#PJM -L "elapse=00:05:00"
#PJM -j
#PJM -L "rscgrp=tutorial“
#PJM -g “gt00“
#PJM -o "test.lst"
#PJM --mpi “proc=12"

export OMP_NUM_THREADS=8
mpiexec ./sol

rm wk.*
```