三次元並列有限要素法（Ⅱ）

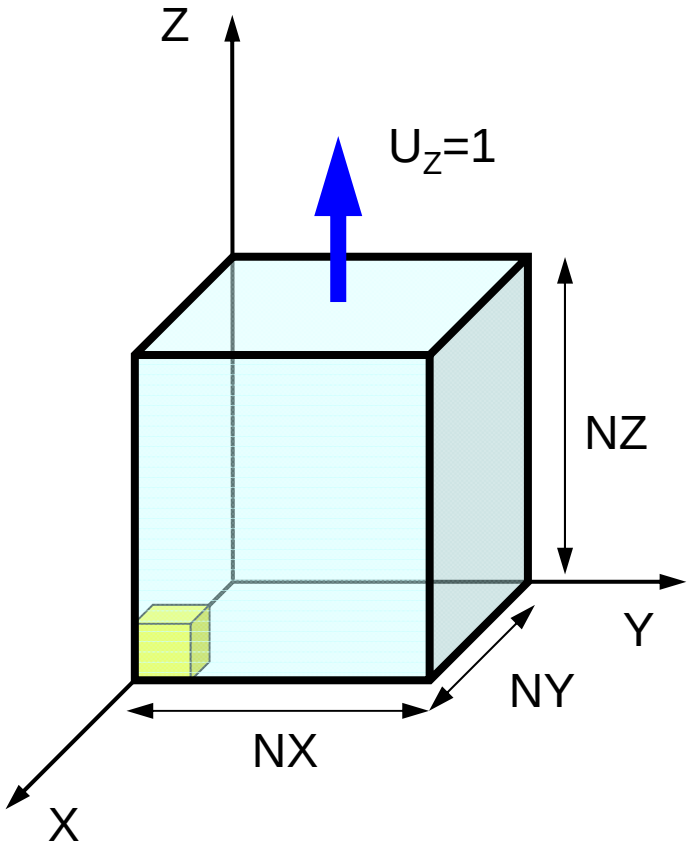
2012年冬学期

中島 研吾

科学技術計算Ⅱ（4820-1028）・コンピュータ科学特別講義（4810-1205）

（並列有限要素法）

対象とする問題



- 弾性体
  - ヤング率　　E
  - ポアソン比　　ν
- 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
  - 対称条件
    - U_X=0@X=0
    - U_Y=0@Y=0
    - U_Z=0@Z=0
  - 強制変位
    - U_Z=1@Z=Z_{max}

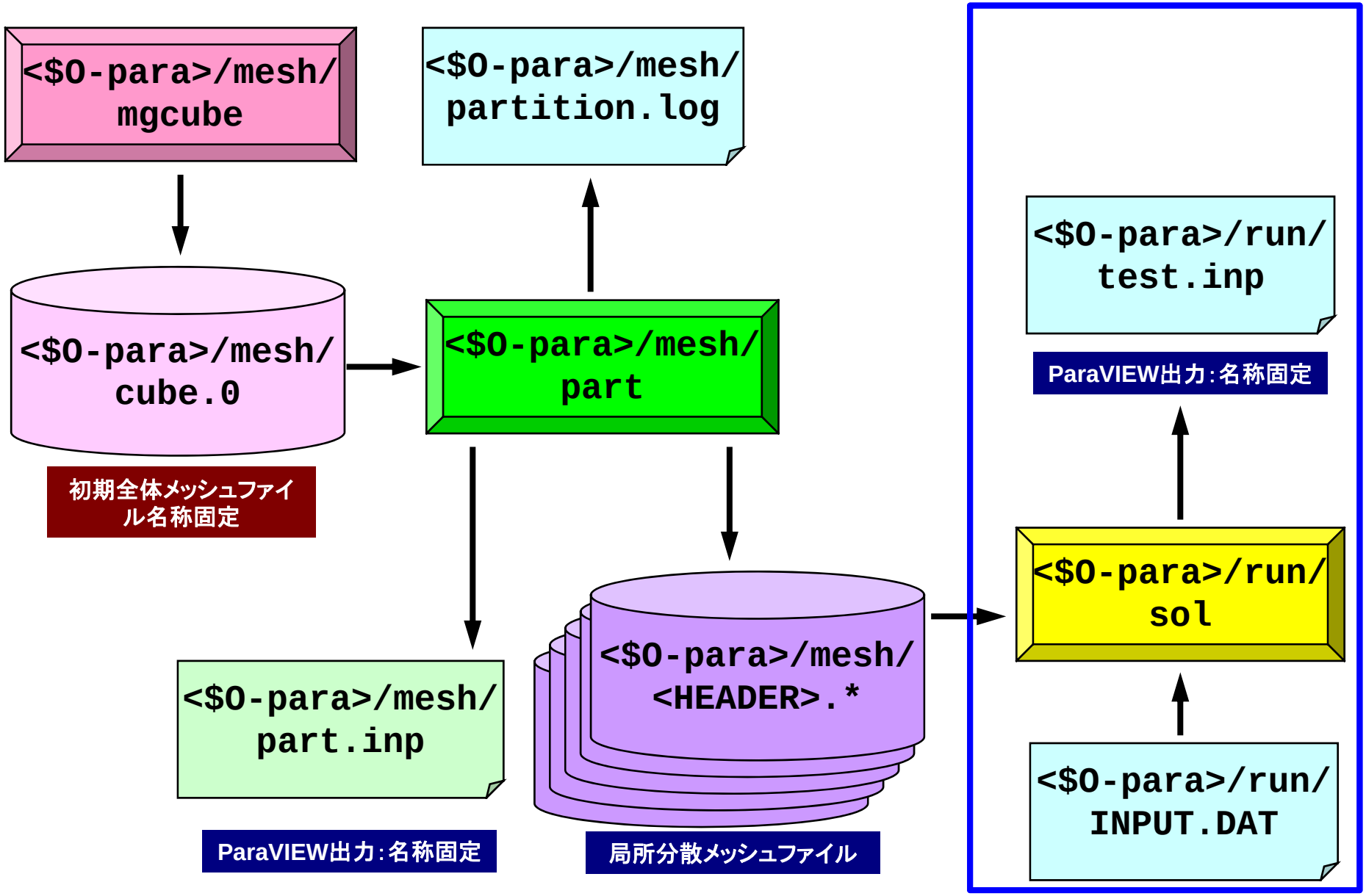
有限要素法の処理

- 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）
- 応力計算

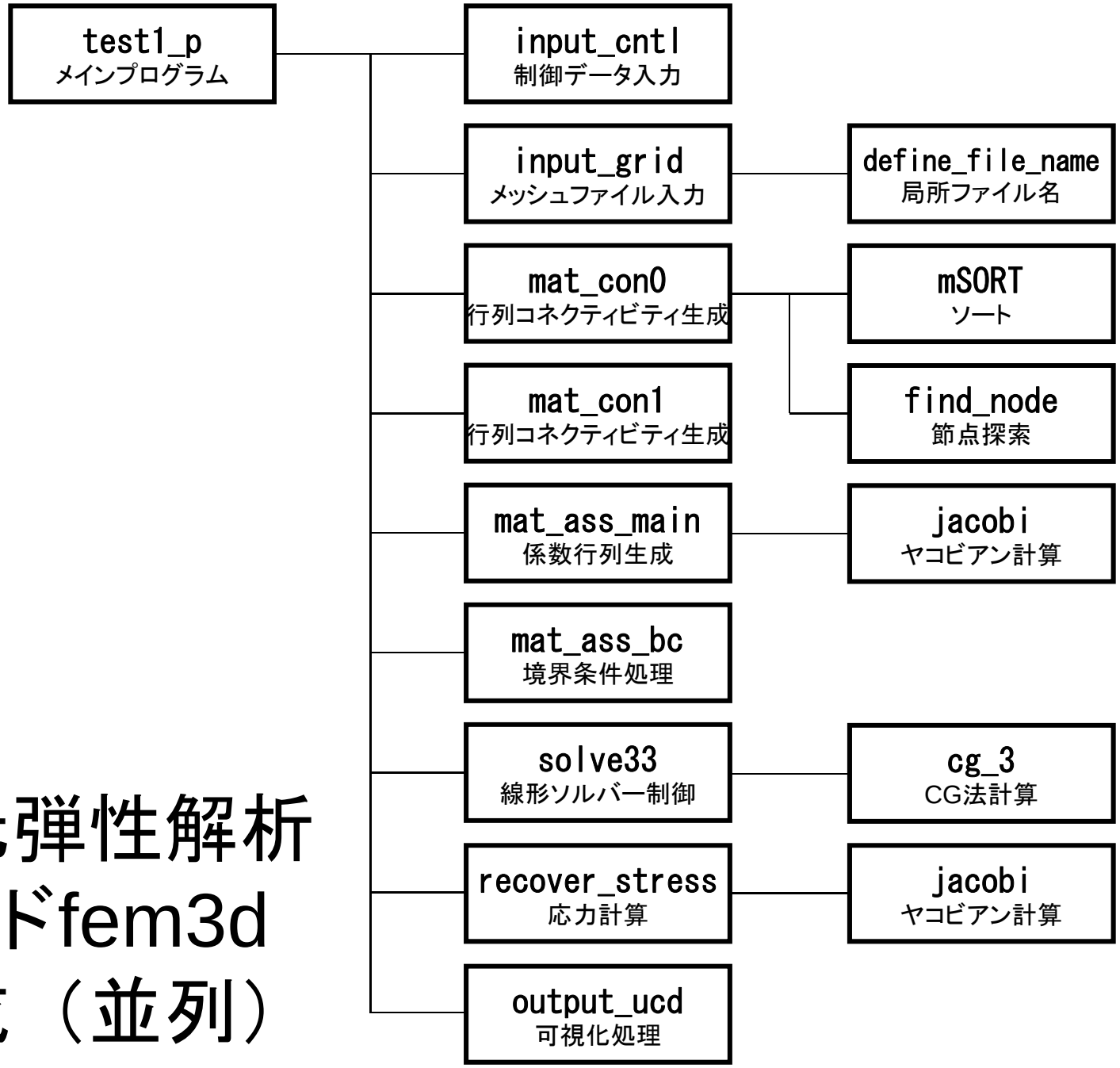
並列有限要素法の手順


<$O-para>/mesh/
mgcube
<$O-para>/mesh/
partition.log
<$O-para>/mesh/
cube.0
<$O-para>/mesh/
part
初期全体メッシュファイル名称固定
<$O-para>/run/
test.inp
ParaVIEW出力:名称固定
<$O-para>/run/
sol
<$O-para>/mesh/
part.inp
<$O-para>/mesh/
<HEADER>.*
<$O-para>/run/
INPUT.DAT
ParaVIEW出力:名称固定
局所分散メッシュファイル

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）
- 応力計算

三次元弾性解析コードfem3dの構成（並列）



全体処理

```
#include <stdio.h>
#include <stdlib.h>
FILE* fp_log;
#define GLOBAL_VALUE_DEFINE
#include "pfem_util.h"
//#include "solver33.h"
extern void PFEM_INIT(int,char**);
extern void INPUT_CNTL();
extern void INPUT_GRID();
extern void MAT_CON0();
extern void MAT_CON1();
extern void MAT_ASS_MAIN();
extern void MAT_ASS_BC();
extern void SOLVE33();
extern void RECOVER_STRESS();
extern void OUTPUT_UCD();
extern void PFEM_FINALIZE();
int main(int argc,char* argv[])
{
        double START_TIME,END_TIME;

        PFEM_INIT(argc,argv);

        INPUT_CNTL();
        INPUT_GRID();

        MAT_CON0();
        MAT_CON1();

        MAT_ASS_MAIN();
        MAT_ASS_BC()  ;

        SOLVE33();

        RECOVER_STRESS();
        OUTPUT_UCD()    ;
        PFEM_FINALIZE() ;
```

Global変数表：pfem_util.h/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	[NODGRPtot+1]	I	各節点グループに含まれる節点数（累積）
NODGRP_ITEM	I	[NODGRP_INDEX[NODGRPTOT+1]]	I	節点グループに含まれる節点
NODGRP_NAME	C80	[NODGRP_INDEX[NODGRPTOT+1]]	I	節点グループ名
NL，NU	I		O	各節点非対角成分数（上三角・下三角）
NPL，NPU	I		O	非対角成分総数（上三角・下三角）
D	R	[9*NP]	O	全体行列：対角ブロック
B,X	R	[3*NP]	O	右辺ベクトル，未知数ベクトル

Global変数表：pfem_util.h/f（2/4）

変数名	種別	サイズ	I/O	内　容
ALUG	R	[9*NP]	O	全体行列：対角ブロックの完全LU分解
AL,AU	R	[9*NPL],[9*NPU]	O	全体行列：上・下三角成分
indexL, indexU	I	[NP+1]	O	全体行列：非零非対角ブロック数
itemL, itemU	I	[NPL], [NPU]	O	全体行列：上・下三角ブロック（列番号）
INL, INU	I	[NP]	O	各節点の上・下三角ブロック数
IAL, IAU	I	[NP][NL],[NP][NU]	O	各節点の上・下三角ブロック（列番号）
IWKX	I	[NP][2]	O	ワーク用配列
METHOD	I		I	反復解法（=1に固定）
PRECOND	I		I	前処理手法（=0：ブロックSSOR, =1：ブロック対角スケーリング）
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）
SIGMA_DIAG	R		I	LU分解時の対角成分係数（=1.0に設定）
pfemIarray	I	[100]	O	諸定数（整数）
pfemRarray	R	[100]	O	諸定数（実数）

Global変数表：pfem_util.h/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]	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
ELAST, POISSON	R		I	ヤング率，ポアソン比
SIGMA_N, TAU_N	R	[N][3]	O	節点における垂直，せん断応力成分

Global変数表：pfem_util.h/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	[NEIBPETOT+1]	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	受信テーブル（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
iterPREmax	I		I	ASDDの繰返数

開始，終了：MPI_Init/Finalize

```
#include "pfem_util.h"
void PFEM_INIT(int argc, char* argv[])
{
        int i;
        MPI_Init(&argc,&argv);
        MPI_Comm_size(MPI_COMM_WORLD,&PETOT);
        MPI_Comm_rank(MPI_COMM_WORLD,&my_rank);

        for(i=0;i<100;i++)  pfemRarray[i]=0.0;
        for(i=0;i<100;i++)  pfemIarray[i]=0;
}
```

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

void  PFEM_FINALIZE()
{
      MPI_Finalize ();

          if( my_rank == 0 ){
                  fprintf(stdout,"* normal terminatio¥n");
                  exit(0);
          }
}
```

制御ファイル入力：INPUT_CNTL

```
#include <stdio.h>
#include <stdlib.h>
#include "pfem_util.h"
/** **/
void INPUT_CNTL()
{
        FILE *fp;
        if( my_rank == 0 ){
                if( (fp=fopen("INPUT.DAT","r")) == NULL){
                        fprintf(stdout,"input file cannot be opened!¥n");
                        exit(1);}
                fscanf(fp,"%s",HEADER);
                fscanf(fp,"%d %d",&METHOD,&PRECOND);
                fscanf(fp,"%d",&iterPREmax);
                fscanf(fp,"%d",&ITER);
                fclose(fp);}
        }

        MPI_Bcast(HEADER ,80,MPI_CHAR,0,MPI_COMM_WORLD);
        MPI_Bcast(&METHOD , 1,MPI_INTEGER,0,MPI_COMM_WORLD);
        MPI_Bcast(&PRECOND, 1,MPI_INTEGER,0,MPI_COMM_WORLD);
        MPI_Bcast(&iterPREmax, 1,MPI_INTEGER,0,MPI_COMM_WORLD);
        MPI_Bcast(&ITER   , 1,MPI_INTEGER,0,MPI_COMM_WORLD);

        SIGMA_DIAG= 1.0;
        SIGMA     = 0.0;
        RESID     = 1.e-8;
        NSET      = 0;

        pfemRarray[0]= RESID;
        pfemRarray[1]= SIGMA_DIAG;
        pfemRarray[2]= SIGMA;

        pfemIarray[0]= ITER;
        pfemIarray[1]= METHOD;
        pfemIarray[2]= PRECOND;
        pfemIarray[3]= NSET;
        pfemIarray[4]= iterPREmax;
}
```

メッシュ入力：INPUT_GRID（1/4）

```
#include <stdio.h>
#include <stdlib.h>
#include "pfem_util.h"
#include "allocate.h"
/*** external functions **/
extern void ERROR_EXIT (int, int);
extern void DEFINE_FILE_NAME(char*,char*,int);
/** **/
void INPUT_GRID()
{
        FILE *fp;
        int i,j,k,ii,kk,kkk,nn,icel,iS,iE,ic0;
        int NTYPE,IMAT;
        int idummy;

        DEFINE_FILE_NAME(HEADER, fname, my_rank);
        if( (fp=fopen(fname,"r")) == NULL){
                fprintf(stdout,"input file cannot be opened!¥n");
                exit(1);}
/**
        NEIB-PE
**/
        fscanf(fp,"%d",&kkk);
        fscanf(fp,"%d",&NEIBPETOT);

        NEIBPE=(int*)allocate_vector(sizeof(int),NEIBPETOT);
        for(i=0;i<NEIBPETOT;i++) fscanf(fp,"%d",&NEIBPE[i]);

        for(i=0;i<NEIBPETOT;i++){
                if( NEIBPE[i] > PETOT-1 ){
                        ERROR_EXIT (202,my_rank);}
        }
```

分散メッシュファイル名：DEFINE_FILE_NAME HEADER+ランク番号

```
#include <stdio.h>
#include <string.h>
void DEFINE_FILE_NAME (char HEADERo[], char filename[],int my_rank)
{
        char string[80];
        sprintf(string,".%-d",my_rank);
        strcpy(filename,HEADERo);
        strcat(filename,string);
}
```

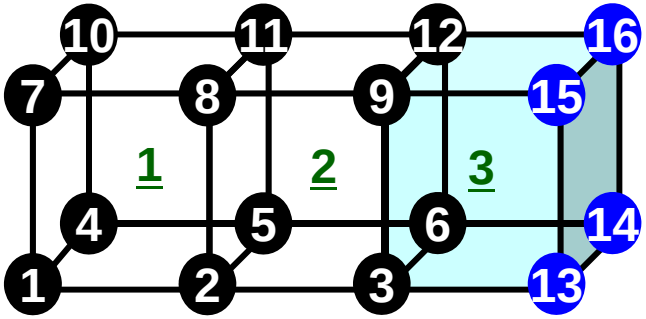
allocate, deallocate関数

```
#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

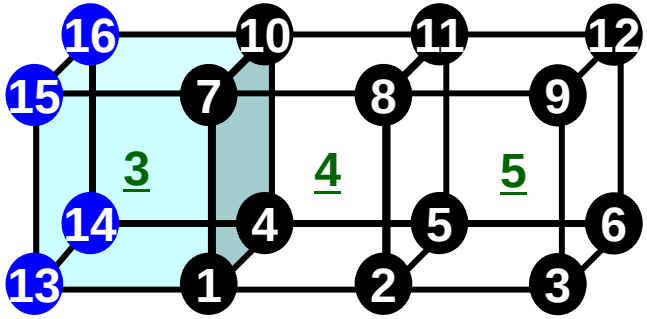
局所番号付け：節点（隣接領域）

aaa.1




```
  1
  1
  0
 16          12
  1     1     0.00     0.00     0.00
  2     1     1.00     0.00     0.00
  3     1     2.00     0.00     0.00
  4     1     0.00     1.00     0.00
  5     1     1.00     1.00     0.00
  6     1     2.00     1.00     0.00
  7     1     0.00     0.00     1.00
  8     1     1.00     0.00     1.00
  9     1     2.00     0.00     1.00
 10     1     0.00     1.00     1.00
 11     1     1.00     1.00     1.00
 12     1     2.00     1.00     1.00
  1     0     3.00     0.00     0.00
  4     0     3.00     1.00     0.00
  7     0     3.00     0.00     1.00
 10     0     3.00     1.00     1.00
```

aaa.0




```
  0          領域ID
  1          隣接領域数  NEIBPETOT
  1          隣接領域ID  NEIBPE(neib)
 16          12
  1     0     3.00     0.00     0.00
  2     0     4.00     0.00     0.00
  3     0     5.00     0.00     0.00
  4     0     3.00     1.00     0.00
  5     0     4.00     1.00     0.00
  6     0     5.00     1.00     0.00
  7     0     3.00     0.00     1.00
  8     0     4.00     0.00     1.00
  9     0     5.00     0.00     1.00
 10     0     3.00     1.00     1.00
 11     0     4.00     1.00     1.00
 12     0     5.00     1.00     1.00
  3     1     2.00     0.00     0.00
  6     1     2.00     1.00     0.00
  9     1     2.00     0.00     1.00
 12     1     2.00     1.00     1.00
```

メッシュ入力：INPUT_GRID（2/4）

```
/**
        NODE
**/
        fscanf(fp,"%d %d",&NP,&N);

        XYZ    =(KREAL**)allocate_matrix(sizeof(KREAL),NP,3);
        NODE_ID=(KINT **)allocate_matrix(sizeof(KINT ),NP,2);

        for(i=0;i<NP;i++){
                for(j=0;j<3;j++){
                        XYZ[i][j]=0.0;
                }
        }

        for(i=0;i<NP;i++){

          fscanf(fp,"%d %d %lf %lf %lf",&NODE_ID[i][0],&NODE_ID[i][1],&XYZ[i][0],&XYZ[i][1],&XYZ[i][2]);
        }
/**
        ELEMENT
**/
        fscanf(fp,"%d %d",&ICELTOT,&ICELTOT_INT);

        ICELNOD=(KINT**)allocate_matrix(sizeof(KINT),ICELTOT,8);
        intELEM_list=(KINT*)allocate_vector(sizeof(KINT),ICELTOT);
        ELEM_ID=(KINT**)allocate_matrix(sizeof(KINT),ICELTOT,2);

        for(i=0;i<ICELTOT;i++) fscanf(fp,"%d",&NTYPE);

        for(icel=0;icel<ICELTOT;icel++){
                fscanf(fp,"%d %d %d %d %d %d %d %d %d %d %d",
                &ELEM_ID[icel][0],&ELEM_ID[icel][1],
                &IMAT,
                &ICELNOD[icel][0],&ICELNOD[icel][1],&ICELNOD[icel][2],&ICELNOD[icel][3],
                &ICELNOD[icel][4],&ICELNOD[icel][5],&ICELNOD[icel][6],&ICELNOD[icel][7]);
        }

        for(ic0=0;ic0<ICELTOT_INT;ic0++) fscanf(fp,"%d",&intELEM_list[ic0]);
```

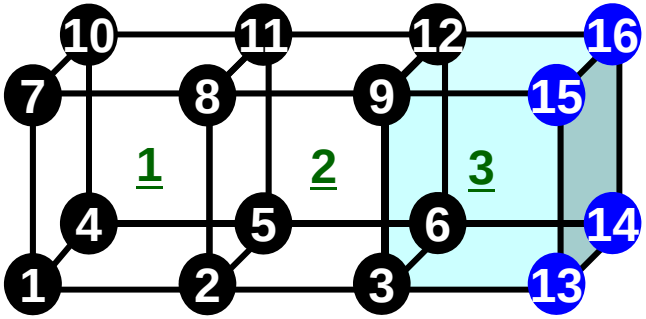
局所番号付け：節点

- 局所番号は各領域「1」から番号付け
  - 1CPUの場合と同じプログラムを使用可能：SPMD
  - 要素番号も同じように「1」から番号付け

- 内点⇒外点という順番で番号付け
- Double　Numbering
  - 本来の所属領域での局所節点番号
  - 所属領域番号

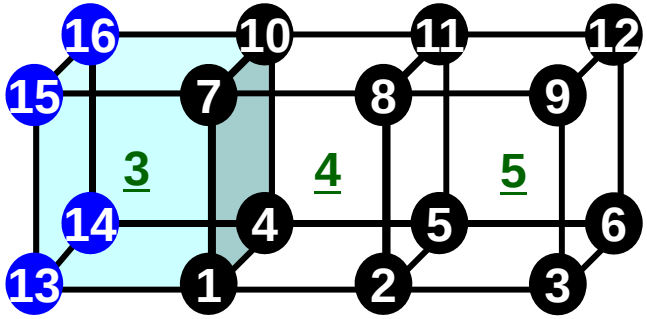
局所番号付け：節点

aaa.1




```
  1
  1
  0
 16          12
  1     1     0.00      0.00      0.00
  2     1     1.00      0.00      0.00
  3     1     2.00      0.00      0.00
  4     1     0.00      1.00      0.00
  5     1     1.00      1.00      0.00
  6     1     2.00      1.00      0.00
  7     1     0.00      0.00      1.00
  8     1     1.00      0.00      1.00
  9     1     2.00      0.00      1.00
 10     1     0.00      1.00      1.00
 11     1     1.00      1.00      1.00
 12     1     2.00      1.00      1.00
  1     0     3.00      0.00      0.00
  4     0     3.00      1.00      0.00
  7     0     3.00      0.00      1.00
 10     0     3.00      1.00      1.00
```

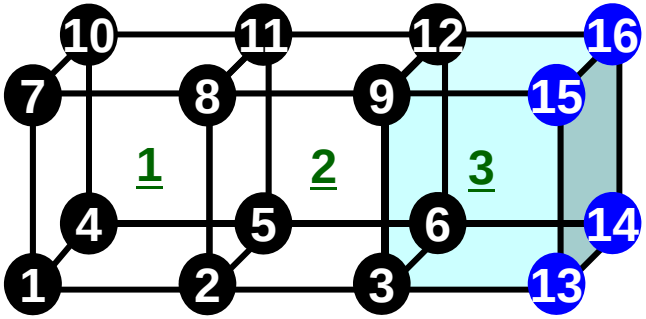
aaa.0




```
  0
  1
  1
 16          12   （総節点数，内点数）
  1     0     3.00      0.00      0.00
  2     0     4.00      0.00      0.00
  3     0     5.00      0.00      0.00
  4     0     3.00      1.00      0.00
  5     0     4.00      1.00      0.00
  6     0     5.00      1.00      0.00
  7     0     3.00      0.00      1.00
  8     0     4.00      0.00      1.00
  9     0     5.00      0.00      1.00
 10     0     3.00      1.00      1.00
 11     0     4.00      1.00      1.00
 12     0     5.00      1.00      1.00
  3     1     2.00      0.00      0.00
  6     1     2.00      1.00      0.00
  9     1     2.00      0.00      1.00
 12     1     2.00      1.00      1.00
```

局所番号付け：節点

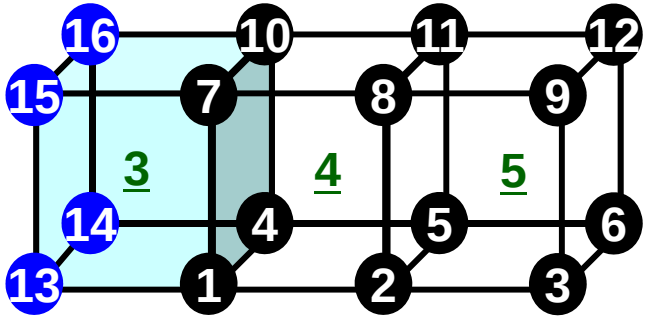
aaa.1




```
  1
  1
  0
 16          12
  1     1     0.00      0.00      0.00   ①
  2     1     1.00      0.00      0.00   ②
  3     1     2.00      0.00      0.00   ③
  4     1     0.00      1.00      0.00   ④
  5     1     1.00      1.00      0.00   ⑤
  6     1     2.00      1.00      0.00   ⑥
  7     1     0.00      0.00      1.00   ⑦
  8     1     1.00      0.00      1.00   ⑧
  9     1     2.00      0.00      1.00   ⑨
 10     1     0.00      1.00      1.00   ⑩
 11     1     1.00      1.00      1.00   ⑪
 12     1     2.00      1.00      1.00   ⑫
  1     0     3.00      0.00      0.00   ⑬
  4     0     3.00      1.00      0.00   ⑭
  7     0     3.00      0.00      1.00   ⑮
 10     0     3.00      1.00      1.00   ⑯
```

所属領域とそこでの番号　　座標

aaa.0

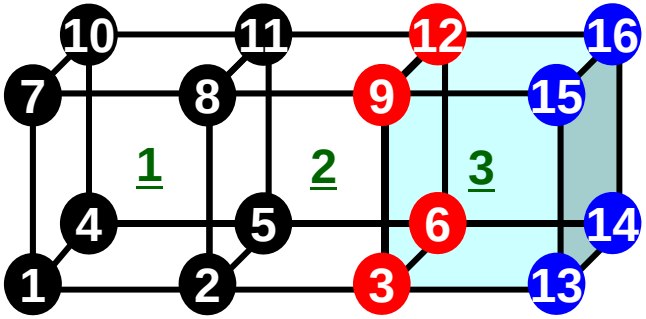



```
  0
  1
  1
 16          12
  1     0     3.00      0.00      0.00   ①
  2     0     4.00      0.00      0.00   ②
  3     0     5.00      0.00      0.00   ③
  4     0     3.00      1.00      0.00   ④
  5     0     4.00      1.00      0.00   ⑤
  6     0     5.00      1.00      0.00   ⑥
  7     0     3.00      0.00      1.00   ⑦
  8     0     4.00      0.00      1.00   ⑧
  9     0     5.00      0.00      1.00   ⑨
 10     0     3.00      1.00      1.00   ⑩
 11     0     4.00      1.00      1.00   ⑪
 12     0     5.00      1.00      1.00   ⑫
  3     1     2.00      0.00      0.00   ⑬
  6     1     2.00      1.00      0.00   ⑭
  9     1     2.00      0.00      1.00   ⑮
 12     1     2.00      1.00      1.00   ⑯
```

所属領域とそこでの番号　　座標

局所番号付け：節点

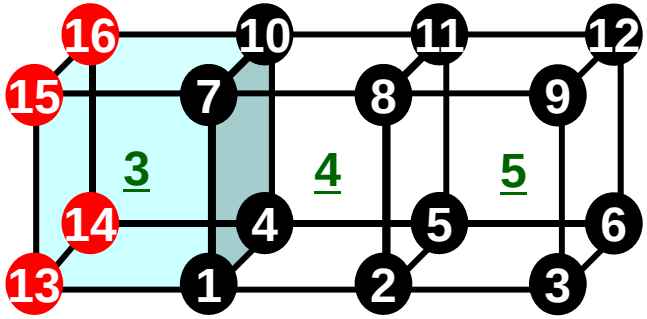
aaa.1




```
  1
  1
  0
 16          12
  1     1     0.00      0.00      0.00   ①
  2     1     1.00      0.00      0.00   ②
  3     1     2.00      0.00      0.00   ③
  4     1     0.00      1.00      0.00   ④
  5     1     1.00      1.00      0.00   ⑤
  6     1     2.00      1.00      0.00   ⑥
  7     1     0.00      0.00      1.00   ⑦
  8     1     1.00      0.00      1.00   ⑧
  9     1     2.00      0.00      1.00   ⑨
 10     1     0.00      1.00      1.00   ⑩
 11     1     1.00      1.00      1.00   ⑪
 12     1     2.00      1.00      1.00   ⑫
  1     0     3.00      0.00      0.00   ⑬
  4     0     3.00      1.00      0.00   ⑭
  7     0     3.00      0.00      1.00   ⑮
 10     0     3.00      1.00      1.00   ⑯
```

所属領域とそこでの番号　　座標

aaa.0




```
  0
  1
  1
 16          12
  1     0     3.00      0.00      0.00   ①
  2     0     4.00      0.00      0.00   ②
  3     0     5.00      0.00      0.00   ③
  4     0     3.00      1.00      0.00   ④
  5     0     4.00      1.00      0.00   ⑤
  6     0     5.00      1.00      0.00   ⑥
  7     0     3.00      0.00      1.00   ⑦
  8     0     4.00      0.00      1.00   ⑧
  9     0     5.00      0.00      1.00   ⑨
 10     0     3.00      1.00      1.00   ⑩
 11     0     4.00      1.00      1.00   ⑪
 12     0     5.00      1.00      1.00   ⑫
  3     1     2.00      0.00      0.00   ⑬
  6     1     2.00      1.00      0.00   ⑭
  9     1     2.00      0.00      1.00   ⑮
 12     1     2.00      1.00      1.00   ⑯
```

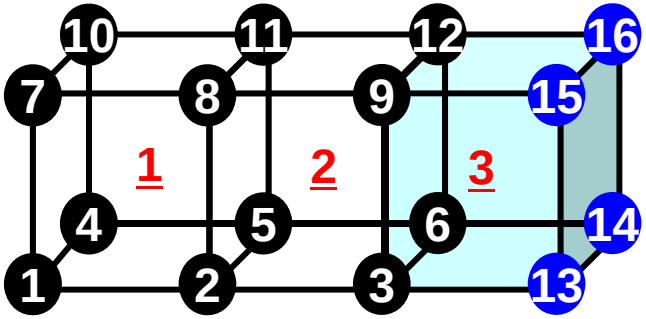
所属領域とそこでの番号　　座標

メッシュ入力：INPUT_GRID（2/4）

```
/**
        NODE
**/
        fscanf(fp,"%d %d",&NP,&N);

        XYZ    =(KREAL**)allocate_matrix(sizeof(KREAL),NP,3);
        NODE_ID=(KINT **)allocate_matrix(sizeof(KINT ),NP,2);

        for(i=0;i<NP;i++){
                for(j=0;j<3;j++){
                        XYZ[i][j]=0.0;
                }
        }

        for(i=0;i<NP;i++){

          fscanf(fp,"%d %d %lf %lf %lf",&NODE_ID[i][0],&NODE_ID[i][1],&XYZ[i][0],&XYZ[i][1],&XYZ[i][2]);
        }
/**
        ELEMENT
**/
        fscanf(fp,"%d %d",&ICELTOT,&ICELTOT_INT);

        ICELNOD=(KINT**)allocate_matrix(sizeof(KINT),ICELTOT,8);
        intELEM_list=(KINT*)allocate_vector(sizeof(KINT),ICELTOT);
        ELEM_ID=(KINT**)allocate_matrix(sizeof(KINT),ICELTOT,2);

        for(i=0;i<ICELTOT;i++) fscanf(fp,"%d",&NTYPE);

        for(icel=0;icel<ICELTOT;icel++){
                fscanf(fp,"%d %d %d %d %d %d %d %d %d %d %d",
                &ELEM_ID[icel][0],&ELEM_ID[icel][1],
                &IMAT,
                &ICELNOD[icel][0],&ICELNOD[icel][1],&ICELNOD[icel][2],&ICELNOD[icel][3],
                &ICELNOD[icel][4],&ICELNOD[icel][5],&ICELNOD[icel][6],&ICELNOD[icel][7]);
        }

        for(ic0=0;ic0<ICELTOT_INT;ic0++) fscanf(fp,"%d",&intELEM_list[ic0]);
```

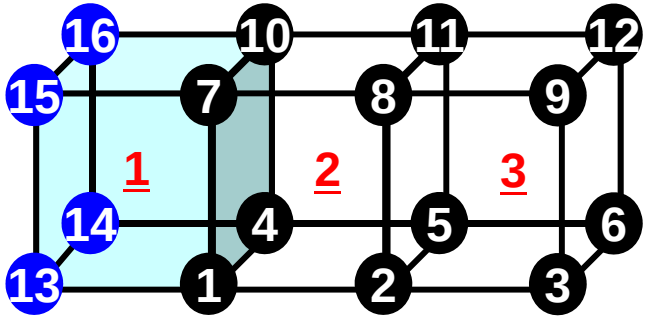
局所番号付け：要素

aaa.1




```
  3   2
361 361 361
  1   1   1   1   2   5   4   7   8  11  10
  2   1   1   2   3   6   5   8   9  12  11
  1   0   1   3  13  14   6   9  15  16  12
  1   2
```

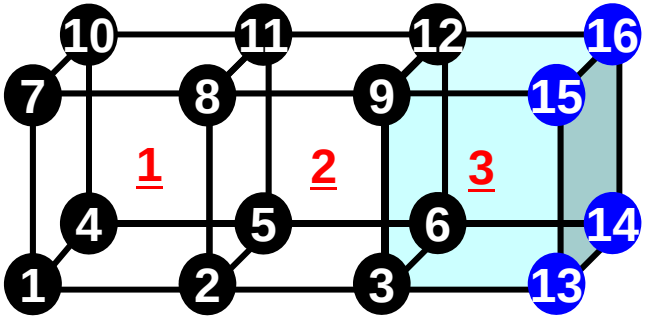
aaa.0



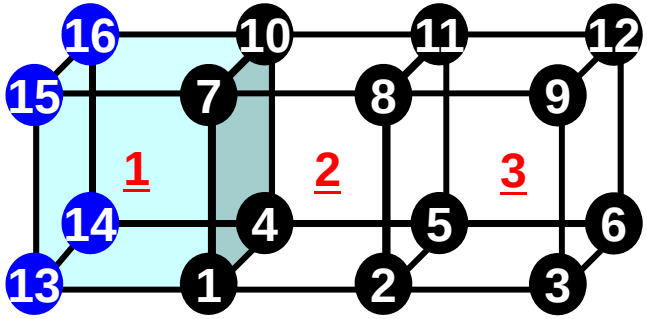

```
  3   3
361 361 361
  1   0   1  13   1   4  14  15   7  10  16
  2   0   1   1   2   5   4   7   8  11  10
  3   0   1   2   3   6   5   8   9  12  11
  1   2   3
```

局所番号付け：要素

aaa.1

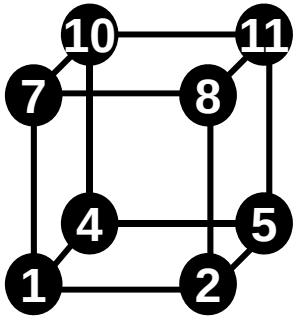


aaa.0




```
  3   2
361 361 361
  1   1   1   1   2   5   4   7   8  11  10
  2   1   1   2   3   6   5   8   9  12  11
  1   0   1   3  13  14   6   9  15  16  12
  1   2
```

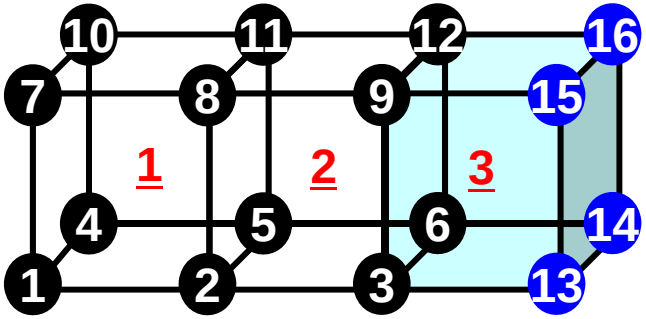
```
  3   3   （全要素，領域所属要素）
361 361 361
  1   0   1  13   1   4  14  15   7  10  16
  2   0   1   1   2   5   4   7   8  11  10
  3   0   1   2   3   6   5   8   9  12  11
  1   2   3
```



- 要素が所属する領域
  - 8個の節点の所属する領域によって決定
  - 全て「内点」であれば，節点と同じ領域
  - 「外点」を含む場合は，節点の所属領域番号の最も若い領域に属する
  - 本ケースのオーバーラップ要素は「0」領域に所属
  - OUTPUT_UCDで利用

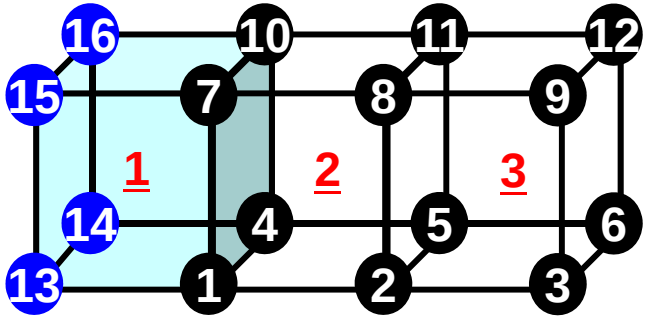
局所番号付け：要素

aaa.1




```
  3    2
361  361  361
  1    1    1    1    2    5    4    7    8   11   10
  2    1    1    2    3    6    5    8    9   12   11
  1    0    1    3   13   14    6    9   15   16   12
  1    2
```

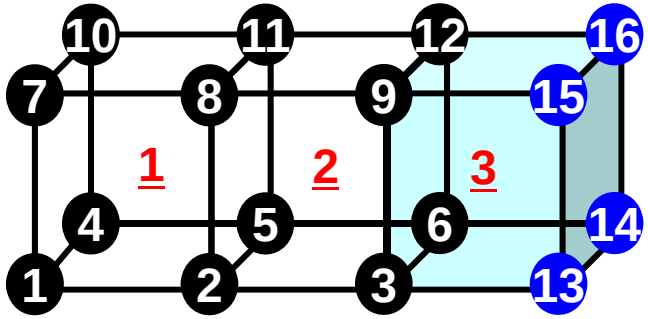
aaa.0



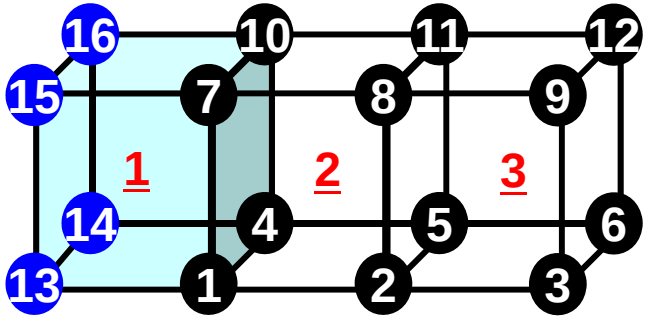

```
  3    3
361  361  361   （要素タイプ，全要素）
  1    0    1   13    1    4   14   15    7   10   16
  2    0    1    1    2    5    4    7    8   11   10
  3    0    1    2    3    6    5    8    9   12   11
  1    2    3
```

局所番号付け：要素

aaa.1



aaa.0



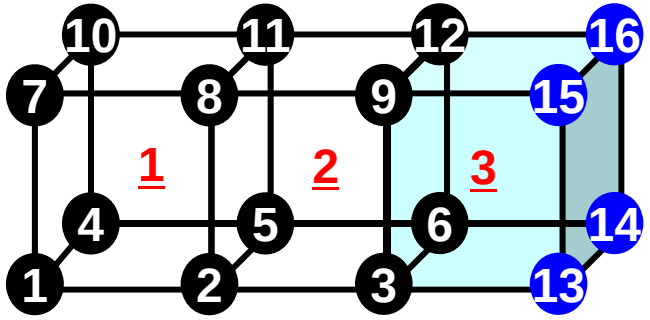

```
  3    2
361 361 361
  1    1    1    1    2    5    4    7    8   11   10   1
  2    1    1    2    3    6    5    8    9   12   11   2
  1    0    1    3   13   14    6    9   15   16   12   3
  1    2
```

```
  3    3
361 361 361
  1    0    1   13    1    4   14   15    7   10   16   1
  2    0    1    1    2    5    4    7    8   11   10   2
  3    0    1    2    3    6    5    8    9   12   11   3
  1    2    3
```

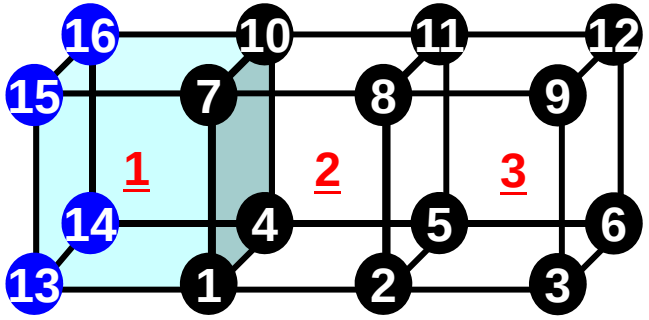
- 要素についてもDouble Numbering
  - 本来の所属領域での局所要素番号
  - 所属領域番号
- 材料番号
- 8個の節点
- 以降の計算では下線付の「局所要素番号」を使用

局所番号付け：要素

aaa.1



aaa.0




```
  3    2
361  361  361
  1    1    1    1    2    5    4    7    8   11   10   1
  2    1    1    2    3    6    5    8    9   12   11   2
  1    0    1    3   13   14    6    9   15   16   12   3
  1    2
```

```
  3    3
361  361  361
  1    0    1   13    1    4   14   15    7   10   16   1
  2    0    1    1    2    5    4    7    8   11   10   2
  3    0    1    2    3    6    5    8    9   12   11   3
  1    2    3
```

- aaa.1
  - 1，2　の要素が「領域所属要素」
- aaa.0
  - 1，2，3　の要素が「領域所属要素」

メッシュ入力：INPUT_GRID（3/4）

```
/**
        COMMUNICATION table
**/
        IMPORT_INDEX=(int*)allocate_vector(sizeof(int),NEIBPETOT+1);
        EXPORT_INDEX=(int*)allocate_vector(sizeof(int),NEIBPETOT+1);

        for(i=0;i<NEIBPETOT+1;++i) IMPORT_INDEX[i]=0;
        for(i=0;i<NEIBPETOT+1;++i) EXPORT_INDEX[i]=0;

        if( PETOT != 1 ) {
                for(i=1;i<=NEIBPETOT;i++) fscanf(fp,"%d",&IMPORT_INDEX[i]);
                nn=IMPORT_INDEX[NEIBPETOT];
                if( nn > 0 ) IMPORT_ITEM=(int*)allocate_vector(sizeof(int),nn);
                for(i=0;i<nn;i++) fscanf(fp,"%d %d",&IMPORT_ITEM[i],&idummy);

                for(i=1;i<=NEIBPETOT;i++) fscanf(fp,"%d",&EXPORT_INDEX[i]);
                nn=EXPORT_INDEX[NEIBPETOT];
                if( nn > 0 ) EXPORT_ITEM=(int*)allocate_vector(sizeof(int),nn);
                for(i=0;i<nn;i++) fscanf(fp,"%d",&EXPORT_ITEM[i]);}
/**
        NODE grp.  info.
**/
        fscanf(fp,"%d",&NODGRPtot);

        NODGRP_INDEX=(KINT*  )allocate_vector(sizeof(KINT),NODGRPtot+1);
        NODGRP_NAME =(CHAR80*)allocate_vector(sizeof(CHAR80),NODGRPtot);
        for(i=0;i<NODGRPtot+1;i++) NODGRP_INDEX[i]=0;

        for(i=0;i<NODGRPtot;i++) fscanf(fp,"%d",&NODGRP_INDEX[i+1]);
        nn=NODGRP_INDEX[NODGRPtot];
        NODGRP_ITEM=(KINT*)allocate_vector(sizeof(KINT),nn);

        for(k=0;k<NODGRPtot;k++){
                iS= NODGRP_INDEX[k];
                iE= NODGRP_INDEX[k+1];
                fscanf(fp,"%s",NODGRP_NAME[k].name);
                nn= iE - iS;
                if( nn != 0 ){
                        for(kk=iS;kk<iE;kk++) fscanf(fp,"%d",&NODGRP_ITEM[kk]);}
        }
```

領域間通信 一般化された通信テーブル

- 「通信」とは「外点」の情報を，その「外点」が本来属している領域から得ることである。

- 「通信テーブル」とは領域間の外点の関係の情報を記述したもの。
  - 「送信テーブル（export）」，「受信テーブル（import）」がある。

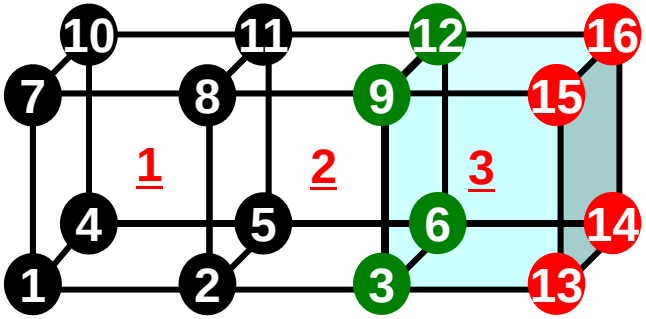
- 送信側：「境界点」として送る
- 受信側：「外点」として受け取る

一般化された通信テーブル：送信

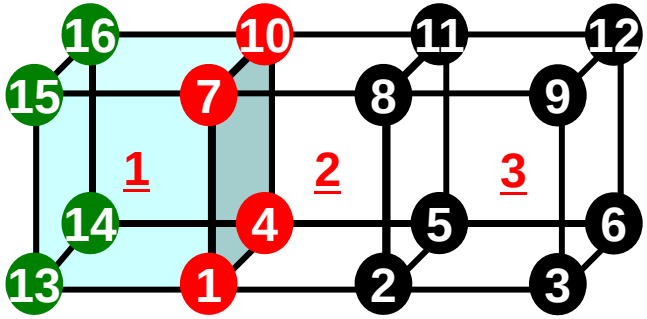
- 送信相手
  - NeibPETot, NeibPE[neib]
- それぞれの送信相手に送るメッセージサイズ
  - export_index[neib], neib= 0, NeibPETot
- 「境界点」番号
  - export_item[k], k= 0, export_index[NeibPETot]-1
- それぞれの送信相手に送るメッセージ
  - SendBuf[k], k= 0, export_index[NeibPETot]-1

通信テーブル(送信)

aaa.1



aaa.0

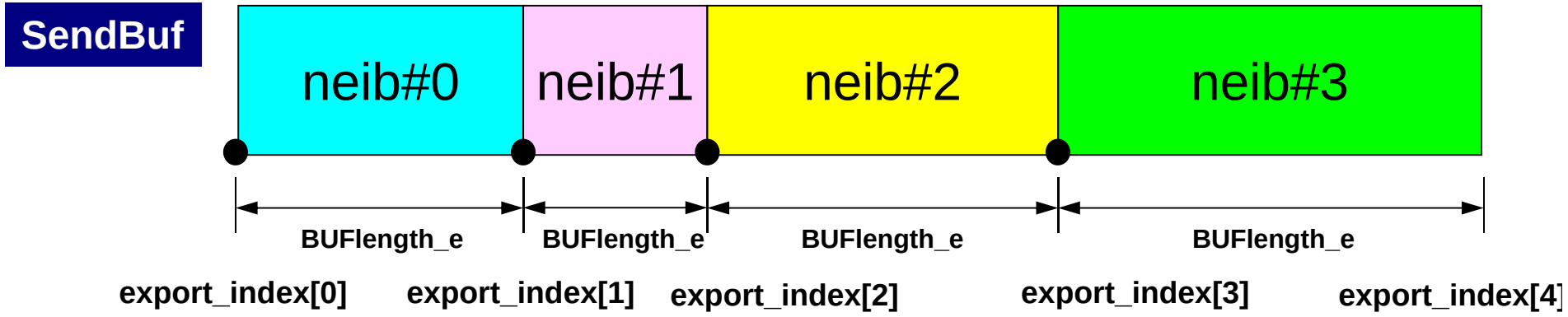



```
 4
13          0
14          0
15          0
16          0
 4
 3
 6
 9
12
```

```
 4
13          1
14          1
15          1
16          1
 4          export_index(neib)：送信節点数
 1          export_item：節点番号
 4
 7
10
```

- export_index 各隣接領域に送信する外点の数（累積数）
  - 現在：隣接領域数は1
- export_item　境界点の番号

送信（MPI_Isend/Irecv/Waitall）



export_index[neib]～export_index[neib+1]-1番目のexport_itemがneib番目の隣接領域に送信される

```
for (neib=0; neib<NeibPETot;neib++){
   for (k=export_index[neib];k<export_index[neib+1];k++){
      kk= export_item[k];
      SendBuf[k]= VAL[kk];
   }
}

for (neib=0; neib<NeibPETot; neib++){
  tag= 0;
  iS_e= export_index[neib];
  iE_e= export_index[neib+1];
  BUFlength_e= iE_e - iS_e

  ierr= MPI_Isend
        (&SendBuf[iS_e], BUFlength_e, MPI_DOUBLE, NeibPE[neib], 0,
         MPI_COMM_WORLD, &ReqSend[neib])
}

MPI_Waitall(NeibPETot, ReqSend, StatSend);
```

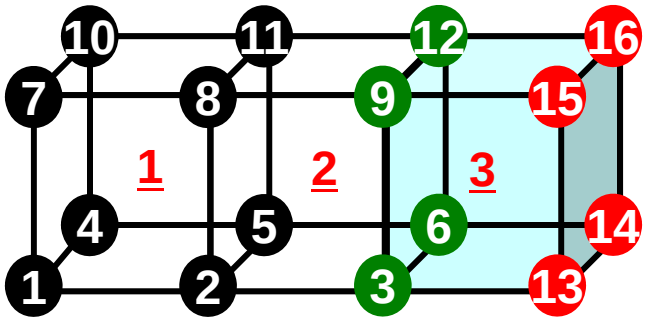
送信バッファへの代入

一般化された通信テーブル：受信

- 受信相手
  - NeibPETot，NeibPE[neib]
- それぞれの受信相手から受け取るメッセージサイズ
  - import_index[neib], neib= 0, NeibPETot
- 「外点」番号
  - import_item[k], k= 0, import_index[NeibPETot]-1
- それぞれの受信相手から受け取るメッセージ
  - RecvBuf[k], k= 0, import_index[NeibPETot]-1

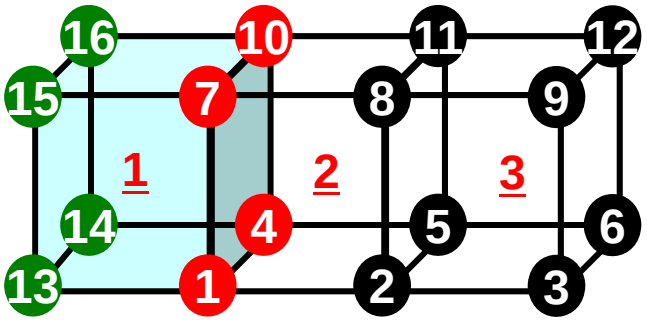
通信テーブル(受信)

aaa.1




```
 4
13          0
14          0
15          0
16          0
 4
 3
 6
 9
12
```

aaa.0




```
 4            import_index(neib)受信節点数
13          1  import_item  節点番号，領域
14          1
15          1
16          1
 4           export_index(neib)
 1           export_item
 4
 7
10
```

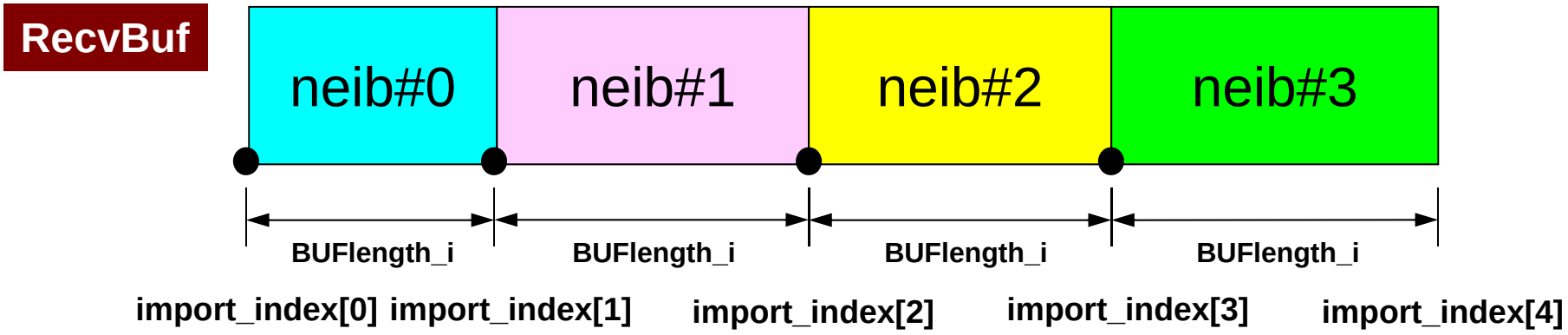
- import_index 各隣接領域から受信する外点の数（累積数）
  - 現在：隣接領域数は1
- import_item　外点の番号，所属領域

受信（MPI_Isend/Irecv/Waitall）

```
for (neib=0; neib<NeibPETot; neib++){
   tag= 0;
   iS_i= import_index[neib];
   iE_i= import_index[neib+1];
   BUFlength_i= iE_i - iS_i

   ierr= MPI_Irecv
         (&RecvBuf[iS_i], BUFlength_i, MPI_DOUBLE, NeibPE[neib], 0,
          MPI_COMM_WORLD, &ReqRecv[neib])
}

MPI_Waitall(NeibPETot, ReqRecv, StatRecv);

for (neib=0; neib<NeibPETot;neib++){
    for (k=import_index[neib];k<import_index[neib+1];k++){
       kk= import_item[k];
       VAL[kk]= RecvBuf[k];
    }
}
```

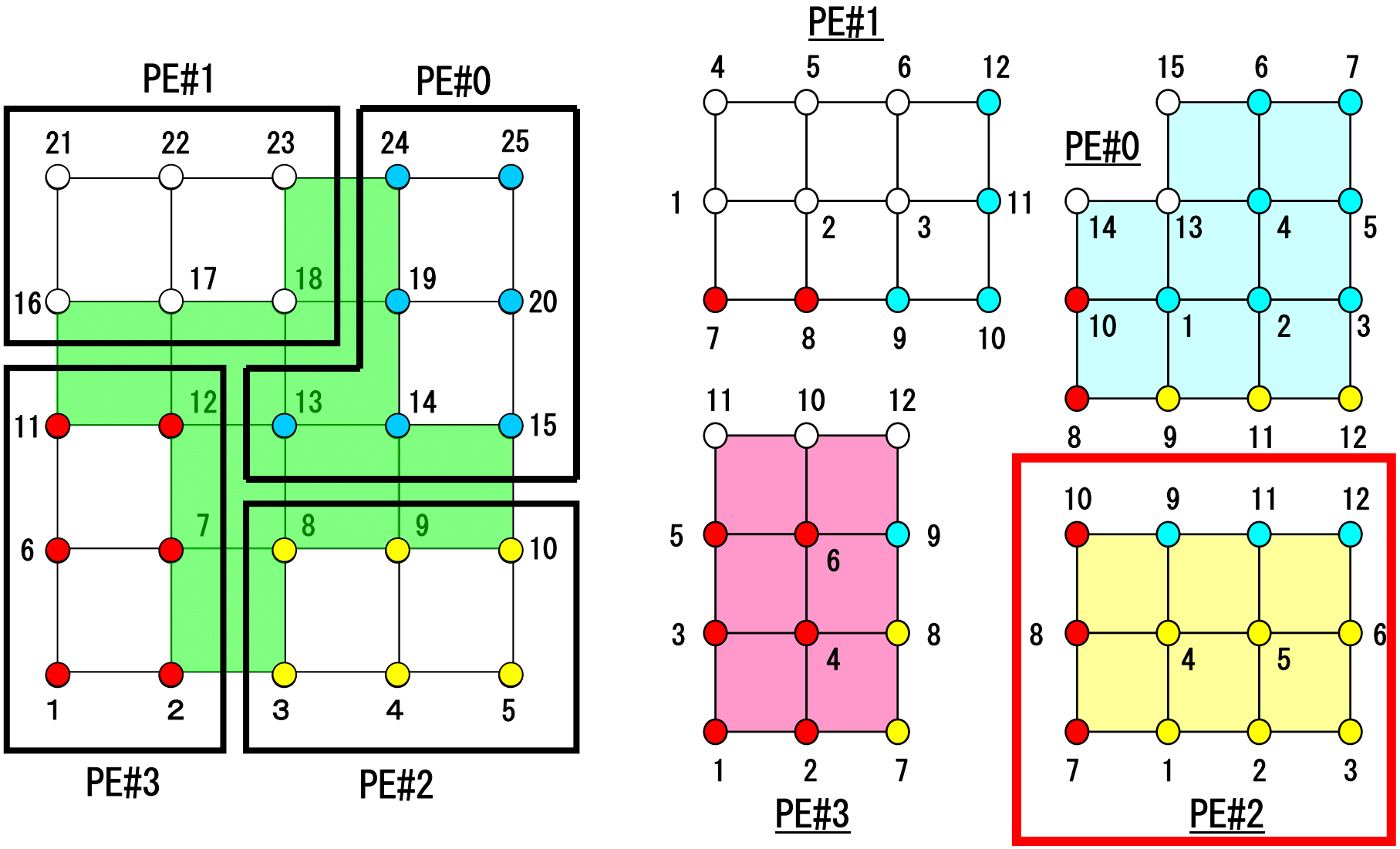
受信バッファからの代入

import_index[neib]～import_index[neib+1]-1番目のimport_itemがneib番目の隣接領域から受信される

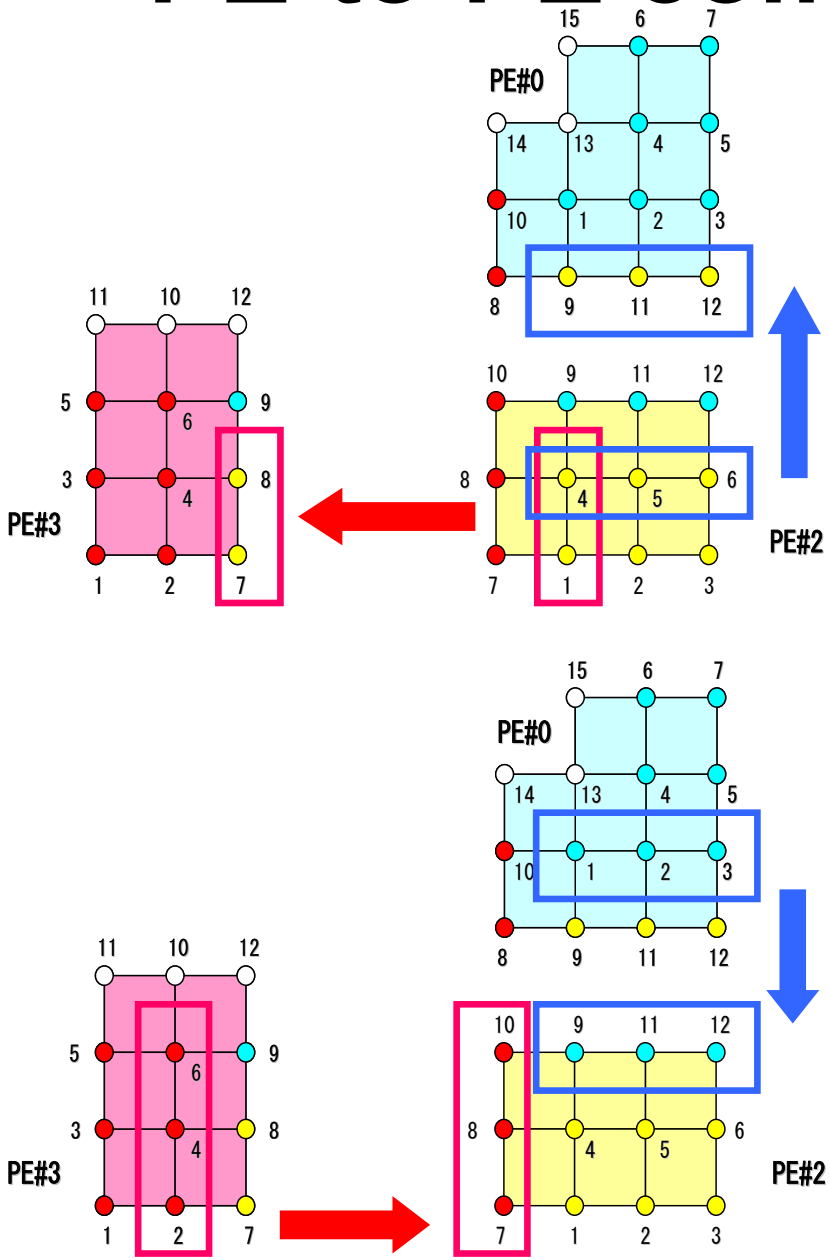


Node-based Partitioning

internal nodes - elements - external nodes

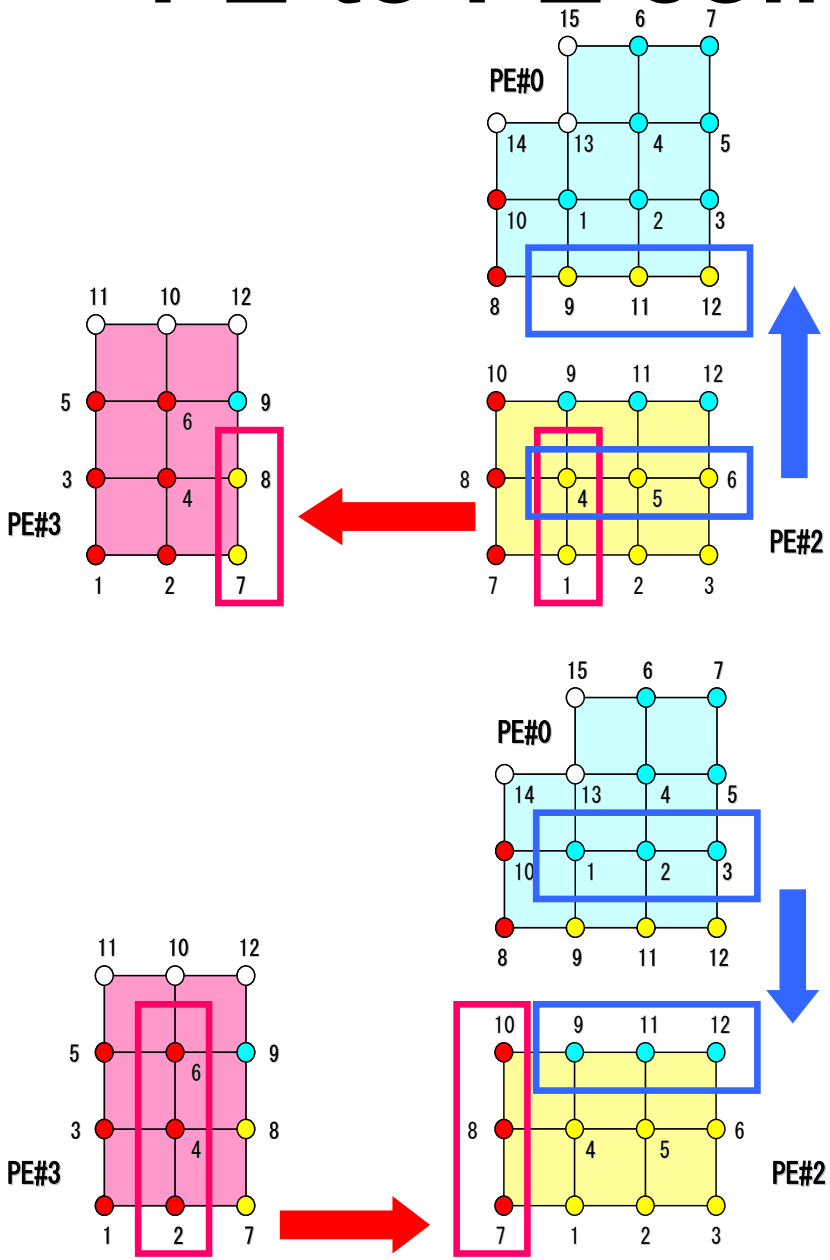


PE-to-PE comm. : Local Data




```
          2
          2
          3          0
（中略）
          3          6
          7          3
          8          3
         10          3
          9          0
         11          0
         12          0
          2          5
          1
          4
          4
          5
          6
```

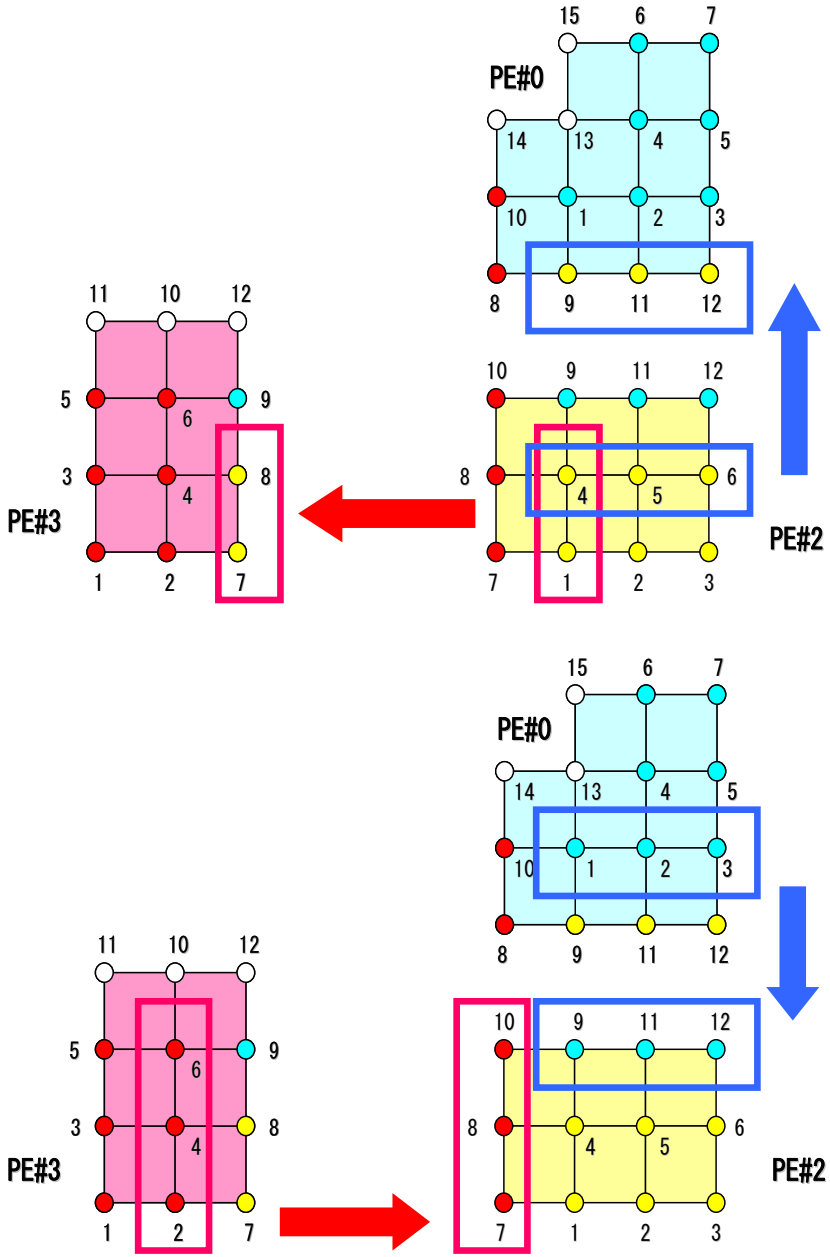
PE-to-PE comm. : Local Data




```
          2                    領域ID

          2                    隣接領域数
          3          0         隣接領域
（中略）
          3          6
          7          3
          8          3
         10          3
          9          0
         11          0
         12          0
          2          5
          1
          4
          4
          5
          6
```

```
NEIBPE= 2
NEIBPE[0]=3, NEIBPE[1]= 0
```

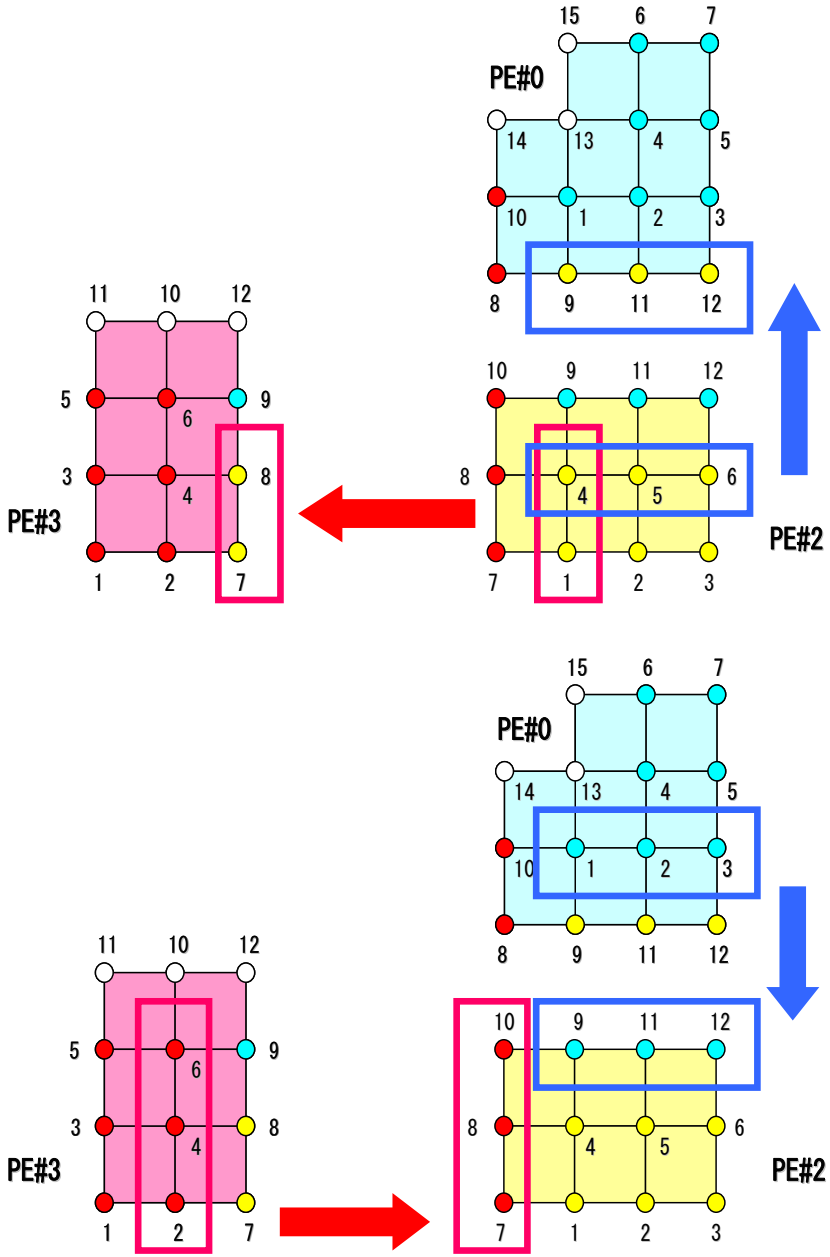
PE-to-PE comm. : SEND




```
          2
          2
          3          0
（中略）
          3          6
          7          3
          8          3
         10          3
          9          0
         11          0
         12          0
          2          5   export_index
          1
          4
          4
          5
          6
```

```
export_index[0]= 0
export_index[1]= 2
export_index[2]= 2+3 = 5

export_item[0-4]=1,4,4,5,6

4番の節点は2つの領域に送られる
```

PE-to-PE comm. : RECV




```
                2
                2
                3              0
（中略）
                3              6  import_index
                7              3
                8              3
               10              3
                9              0
               11              0
               12              0
                2              5
                1
                4
                4
                5
                6
```

```
import_index[0]= 0
import_index[1]= 3
import_index[2]= 3+3 = 6

import_item[0-5]=7,8,10,9,11,12
```

メッシュ入力：INPUT_GRID（4/4）

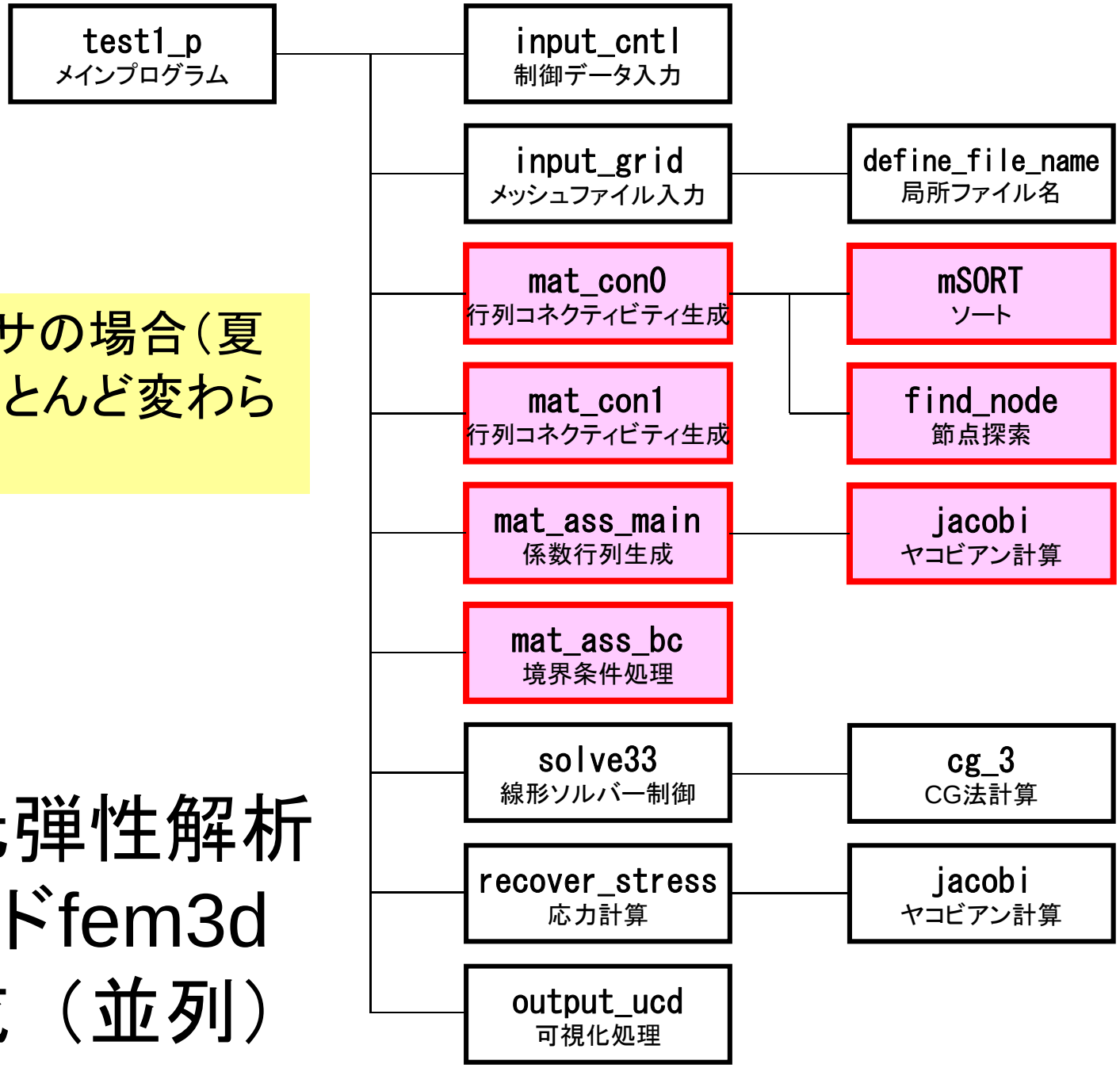
```
/**
        ELEMENT grp. info.
**/
        fscanf(fp,"%d",&ELMGRPtot);

        ELMGRP_INDEX=(KINT*  )allocate_vector(sizeof(int),ELMGRPtot+1);
        ELMGRP_NAME =(CHAR80*)allocate_vector(sizeof(CHAR80),ELMGRPtot);
        for(i=0;i<ELMGRPtot+1;i++) ELMGRP_INDEX[i]=0;
        for(i=0;i<ELMGRPtot;i++) fscanf(fp,"%d",&ELMGRP_INDEX[i+1]);
        nn=ELMGRP_INDEX[ELMGRPtot];
        ELMGRP_ITEM=(KINT *)allocate_vector(sizeof(KINT),nn);

        for(k=0;k<ELMGRPtot;k++){
                iS=ELMGRP_INDEX[k];
                iE=ELMGRP_INDEX[k+1];
                fscanf(fp,"%s",ELMGRP_NAME[k].name);
                nn= iE - iS ;
                if( nn != 0 ){
                        for(kk=iS;kk<iE;kk++) fscanf(fp,"%d",&ELMGRP_ITEM[kk]);} }
/**
        SURFACE grp. info.
**/
        fscanf(fp,"%d",&SUFGRPtot);

        SUFGRP_INDEX=(KINT*  )allocate_vector(sizeof(KINT),SUFGRPtot+1);
        SUFGRP_NAME =(CHAR80*)allocate_vector(sizeof(CHAR80),SUFGRPtot);
        for(i=0;i<SUFGRPtot+1;i++) SUFGRP_INDEX[i]=0;
        for(i=0;i<SUFGRPtot;i++) fscanf(fp,"%d",&SUFGRP_INDEX[i+1]);
        nn= SUFGRP_INDEX[SUFGRPtot];
        SUFGRP_ITEM=(KINT*)allocate_vector(sizeof(KINT),2*nn);

        for(k=0;k<SUFGRPtot;k++){
                iS=SUFGRP_INDEX[k];
                iE=SUFGRP_INDEX[k+1];
                fscanf(fp,"%s",SUFGRP_NAME[k].name);
                nn= iE - iS;
                if( nn != 0 ){
                        for(kk=iS;kk<iE;kk++) fscanf(fp,"%d",&SUFGRP_ITEM[2*kk  ]);
                        for(kk=iS;kk<iE;kk++) fscanf(fp,"%d",&SUFGRP_ITEM[2*kk+1]);} }
        fclose(fp);}
```

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）
- 応力計算

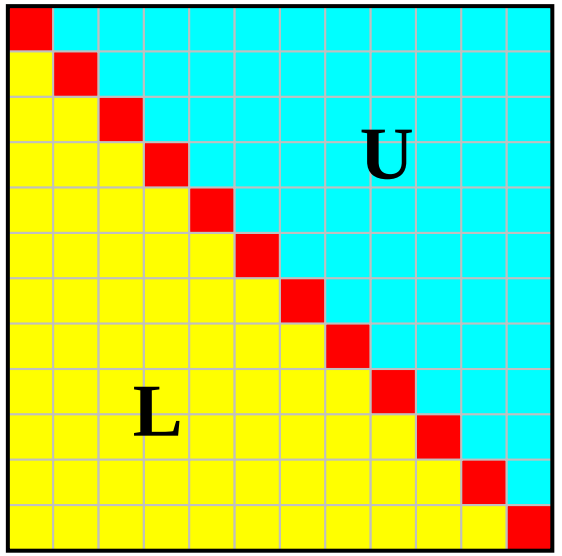
1プロセッサの場合（夏学期）とほとんど変わらない

三次元弾性解析コードfem3dの構成（並列）



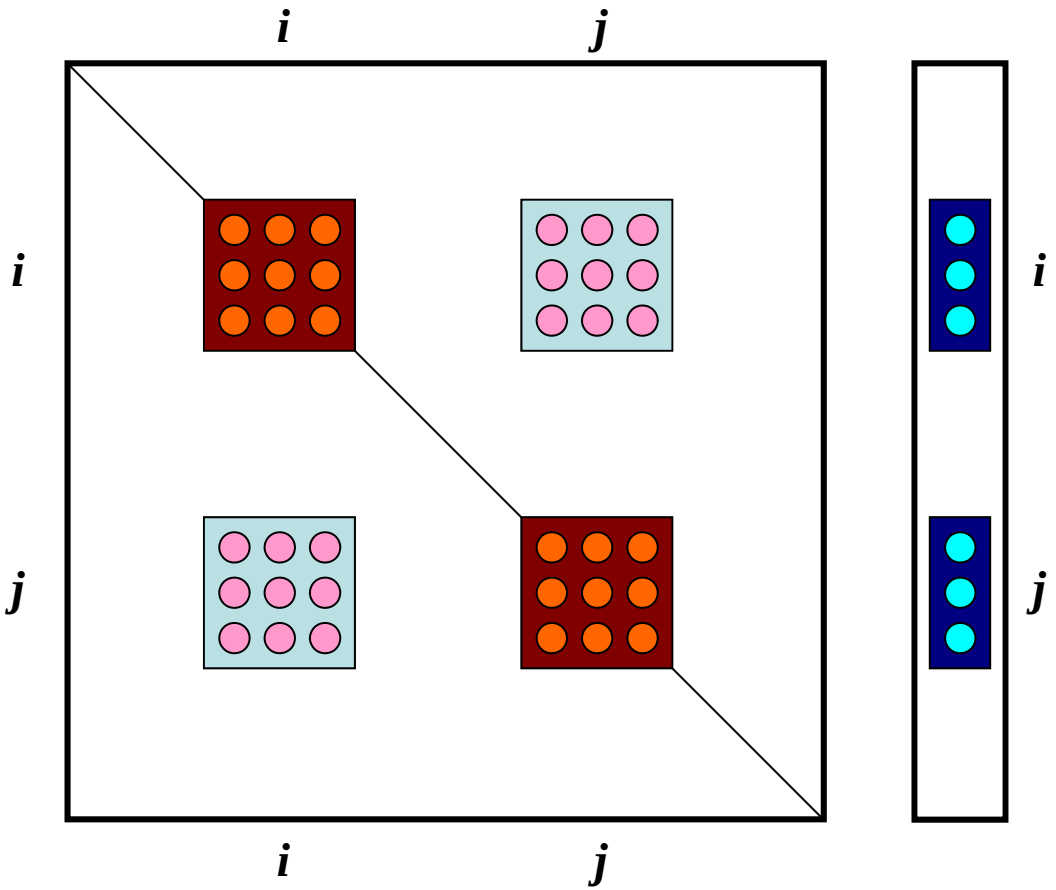
fem3d：いくつかの特徴

- 非対角成分
  - 上三角，下三角を分けて記憶
    - indexL，itemL，AL
    - indexU，itemU，AU
- ブロックとして記憶
  - ベクトル：1節点3成分
  - 行列：各ブロック9成分
  - 行列の各成分ではなく，節点上の3変数に基づくブロックとして処理する



ブロックとして記憶（1/3）

- 記憶容量が減る
  - index, itemに関する記憶容量を数十分の1に削減できる



ブロックとして記憶（2/3）

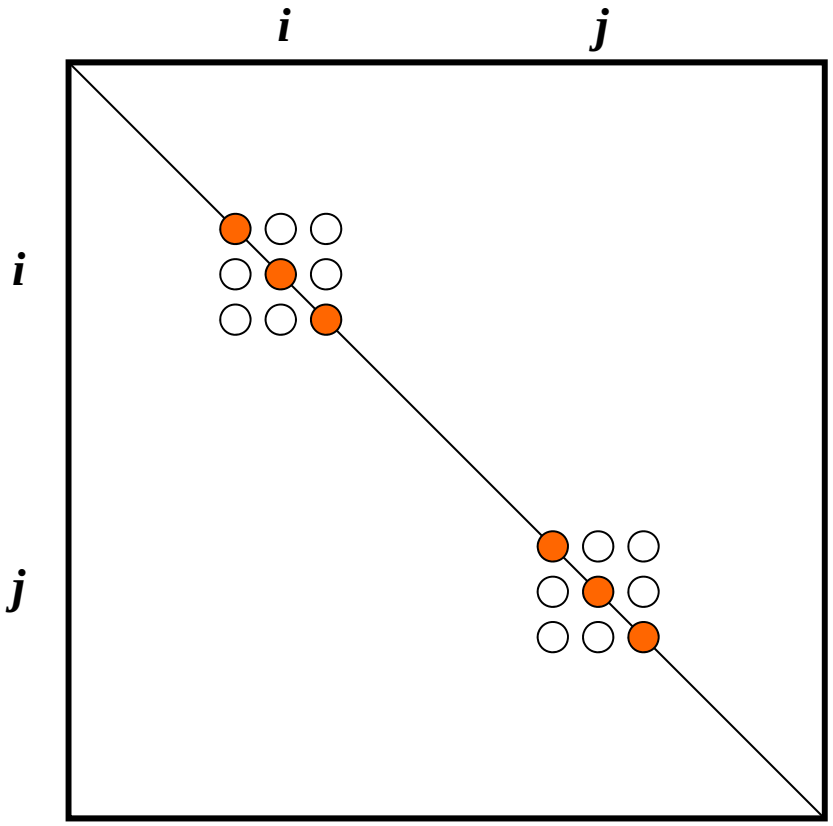
- 計算効率
  - 間接参照（メモリに負担）と計算の比が小さくなる
  - ベクトル，スカラー共に効く：2倍以上の性能

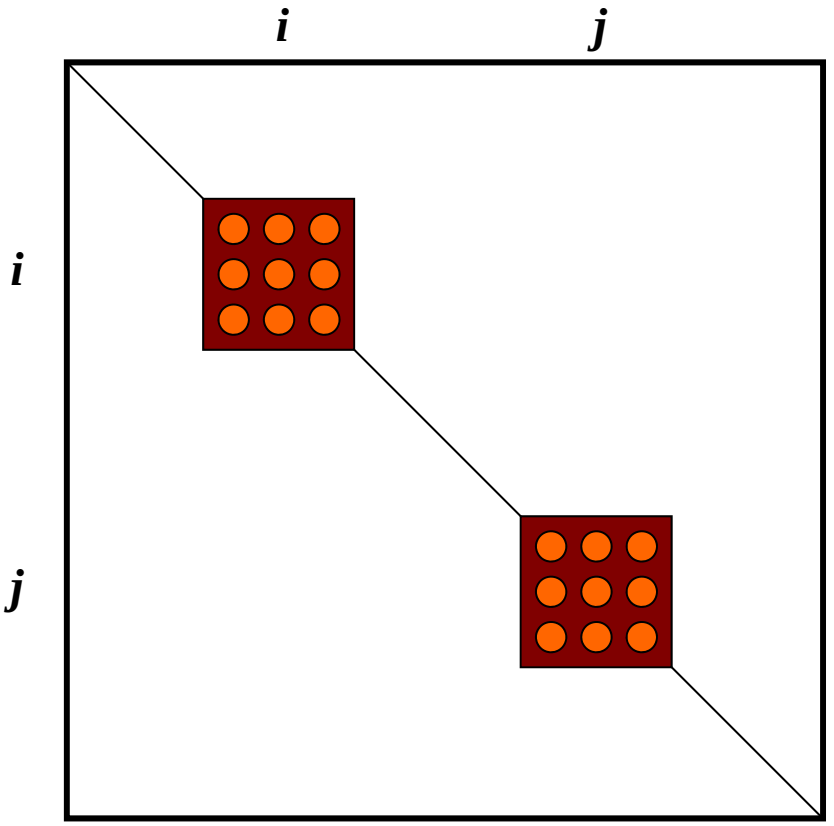
```
do i= 1, 3*N
  Y(i)= D(i)*X(i)
  do k= index(i-1)+1, index(i)
    kk= item(k)
    Y(i)= Y(i) + AMAT(k)*X(k)
  enddo
enddo
```

```
do i= 1, N
  X1= X(3*i-2)
  X2= X(3*i-1)
  X3= X(3*i)
  Y(3*i-2)= D(9*i-8)*X1+D(9*i-7)*X2+D(9*i-6)*X3
  Y(3*i-1)= D(9*i-5)*X1+D(9*i-4)*X2+D(9*i-3)*X3
  Y(3*I  )= D(9*i-2)*X1+D(9*i-1)*X2+D(9*I  )*X3
  do k= index(i-1)+1, index(i)
    kk= item(k)
    X1= X(3*kk-2)
    X2= X(3*kk-1)
    X3= X(3*kk)
    Y(3*i-2)= Y(3*i-2)+AMAT(9*k-8)*X1+AMAT(9*k-7)*X2 &
                                     +AMAT(9*k-6)*X3
    Y(3*i-1)= Y(3*i-1)+AMAT(9*k-5)*X1+AMAT(9*k-4)*X2 &
                                     +AMAT(9*k-3)*X3
    Y(3*I  )= Y(3*I  )+AMAT(9*k-2)*X1+AMAT(9*k-1)*X2 &
                                     +AMAT(9*k  )*X3
  enddo
enddo
```

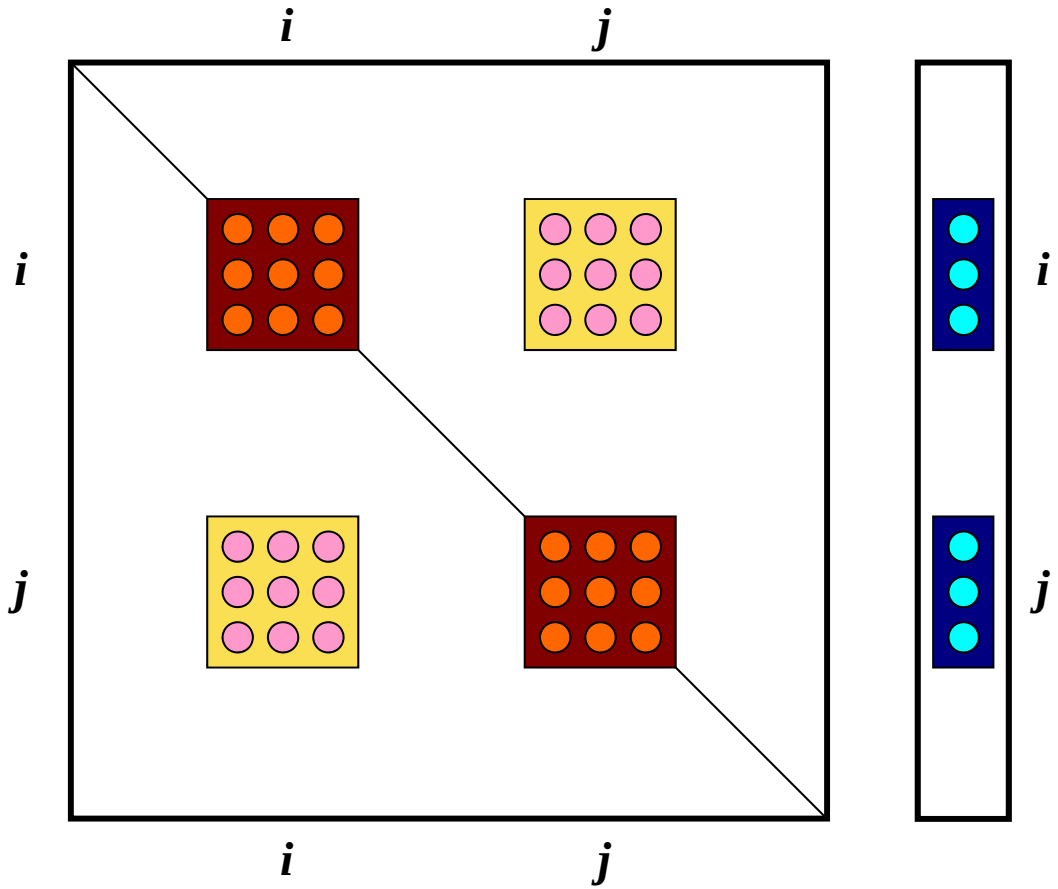
ブロックとして記憶（3/3）

- 計算の安定化
  - 対角成分で割るのではなく，対角ブロックの完全LU分解を求めて解く
  - 特に悪条件問題で有効

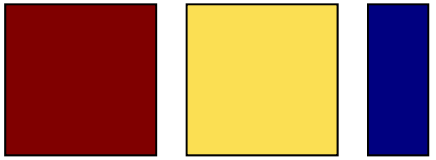




用語の定義



ブロック（節点）:



成分（自由度）:

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
- INL[NP]，INU[NP]，IAL[NP][NL]，IAU[NP][NU]を使って非ゼロ非対角ブロック数（上下）を予備的に勘定する

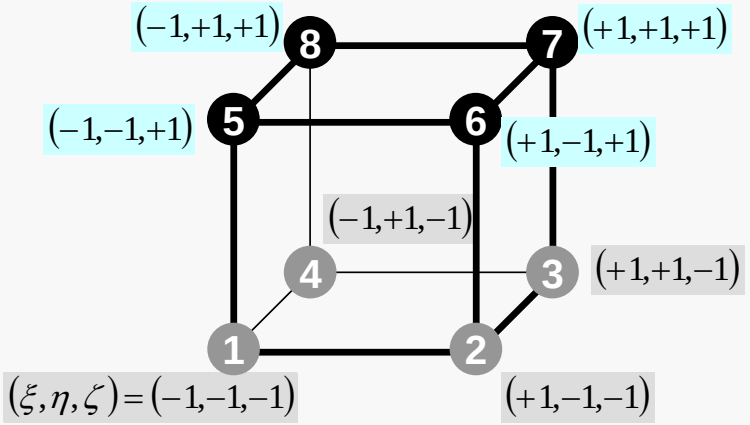
全体処理

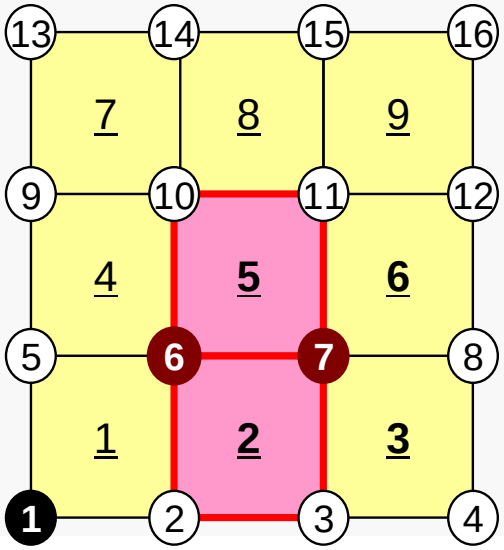
```
#include <stdio.h>
#include <stdlib.h>
FILE* fp_log;
#define GLOBAL_VALUE_DEFINE
#include "pfem_util.h"
//#include "solver33.h"
extern void PFEM_INIT(int,char**);
extern void INPUT_CNTL();
extern void INPUT_GRID();
extern void MAT_CON0();
extern void MAT_CON1();
extern void MAT_ASS_MAIN();
extern void MAT_ASS_BC();
extern void SOLVE33();
extern void RECOVER_STRESS();
extern void OUTPUT_UCD();
extern void PFEM_FINALIZE();
int main(int argc,char* argv[])
{
        double START_TIME,END_TIME;

        PFEM_INIT(argc,argv);

        INPUT_CNTL();
        INPUT_GRID();

        MAT_CON0();
        MAT_CON1();

        MAT_ASS_MAIN();
        MAT_ASS_BC()  ;

        SOLVE33();

        RECOVER_STRESS();
        OUTPUT_UCD()    ;
        PFEM_FINALIZE() ;
```

MAT_CON0: INL,INU, IAL, IAU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）

```
#include <stdio.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE *fp_log;
extern void mSORT(int*, int*, int);
static void FIND_TS_NODE (int,int);

void MAT_CON0()
{
        int i,j,k,icel,in;
        int in1,in2,in3,in4,in5,in6,in7,in8;
        int NN;

        N2= 256; (この変数は使用しない)
        NU= 26;
        NL= 26;

        INL=(KINT* )allocate_vector(sizeof(KINT),NP);
        IAL=(KINT**)allocate_matrix(sizeof(KINT),NP,NL);
        INU=(KINT* )allocate_vector(sizeof(KINT),NP);
        IAU=(KINT**)allocate_matrix(sizeof(KINT),NP,NU);

        for(i=0;i<NP;i++) INL[i]=0;
        for(i=0;i<NP;i++) for(j=0;j<NL;j++) IAL[i][j]=0;
        for(i=0;i<NP;i++) INU[i]=0;
        for(i=0;i<NP;i++) for(j=0;j<NU;j++) IAU[i][j]=0;
```

NU, NL:
各節点における
(上下)非ゼロ非対角
ブロックの最大数
(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:
⇒夏学期レポート課題

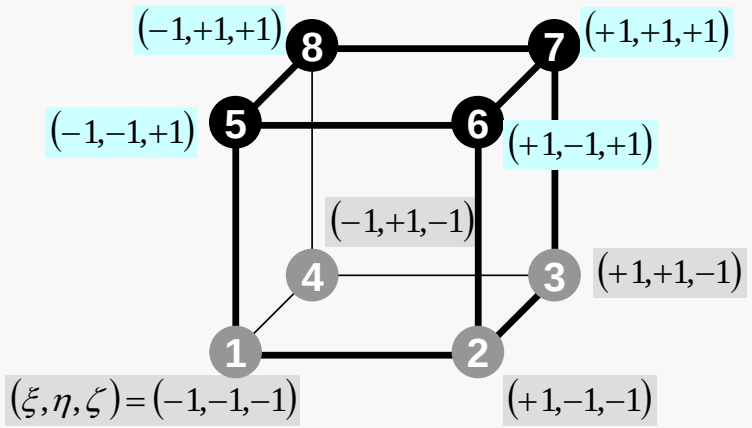
行列コネクティビティ生成：MAT_CON0（1/4）

```
#include <stdio.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE *fp_log;
extern void mSORT(int*, int*, int);
static void FIND_TS_NODE (int,int);

void MAT_CON0()
{
        int i,j,k,icel,in;
        int in1,in2,in3,in4,in5,in6,in7,in8;
        int NN;

        N2= 256; (この変数は使用しない)
        NU= 26;
        NL= 26;

        INL=(KINT* )allocate_vector(sizeof(KINT),NP);
        IAL=(KINT**)allocate_matrix(sizeof(KINT),NP,NL);
        INU=(KINT* )allocate_vector(sizeof(KINT),NP);
        IAU=(KINT**)allocate_matrix(sizeof(KINT),NP,NU);

        for(i=0;i<NP;i++) INL[i]=0;
        for(i=0;i<NP;i++) for(j=0;j<NL;j++) IAL[i][j]=0;
        for(i=0;i<NP;i++) INU[i]=0;
        for(i=0;i<NP;i++) for(j=0;j<NU;j++) IAU[i][j]=0;
```

変数名	サイズ	内　容
INL, INU	[NP]	各節点の上・下三角ブロック数
IAL, IAU	[NP][NL], [NP][NU]	各節点の上・下三角ブロック（列番号）

行列コネクティビティ生成：MAT_CON0（2/4）

```
for( icel=0;icel< ICELTOT;icel++){
    in1=ICELNOD[icel][0];
    in2=ICELNOD[icel][1];
    in3=ICELNOD[icel][2];
    in4=ICELNOD[icel][3];
    in5=ICELNOD[icel][4];
    in6=ICELNOD[icel][5];
    in7=ICELNOD[icel][6];
    in8=ICELNOD[icel][7];

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

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

    FIND_TS_NODE (in3,in1);
    FIND_TS_NODE (in3,in2);
    FIND_TS_NODE (in3,in4);
    FIND_TS_NODE (in3,in5);
    FIND_TS_NODE (in3,in6);
    FIND_TS_NODE (in3,in7);
    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

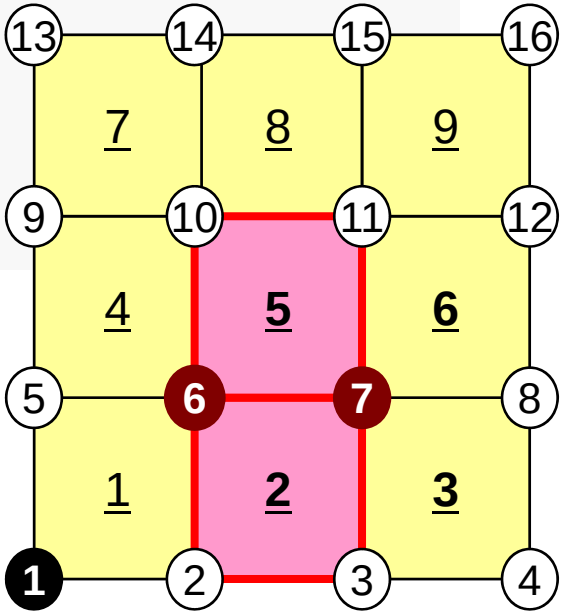
```
static void FIND_TS_NODE (int ip1,int ip2)

{
        int kk,icou;
        if( ip1 > ip2 ){
                for(kk=0;kk<INL[ip1-1];kk++){
                        if(ip2 == IAL[ip1-1][kk]) return;
                }
                icou=INL[ip1-1]+1;
                IAL[ip1-1][icou-1]=ip2;
                INL[ip1-1]=icou;
                return;
        }

        if( ip2 > ip1 ) {
                for(kk=0;kk<INU[ip1-1];kk++){
                        if(ip2 == IAU[ip1-1][kk]) return;
                }
                icou=INU[ip1-1]+1;
                IAU[ip1-1][icou-1]=ip2;
                INU[ip1-1]=icou;
                return;
        }
}
```

変数名	サイズ	内　容
INL, INU	[NP]	各節点の上・下三角ブロック数
IAL, IAU	[NP][NL], [NP][NU]	各節点の上・下三角ブロック （列番号）

節点探索：FIND_TS_NODE
一次元ではこの部分は手動

```
static void FIND_TS_NODE (int ip1,int ip2)

{
        int kk,icou;
        if( ip1 > ip2 ){
                for(kk=0;kk<INL[ip1-1];kk++){
                        if(ip2 == IAL[ip1-1][kk]) return;
                }
                icou=INL[ip1-1]+1;
                IAL[ip1-1][icou-1]=ip2;
                INL[ip1-1]=icou;
                return;
        }

        if( ip2 > ip1 ) {
                for(kk=0;kk<INU[ip1-1];kk++){
                        if(ip2 == IAU[ip1-1][kk]) return;
                }
                icou=INU[ip1-1]+1;
                IAU[ip1-1][icou-1]=ip2;
                INU[ip1-1]=icou;
                return;
        }
}
```

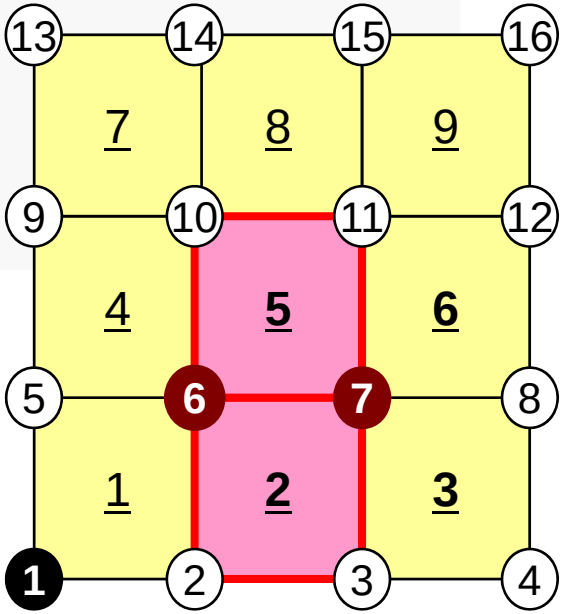
既にIAL，IAUに含まれている場合は，次のペアへ



節点探索：FIND_TS_NODE
一次元ではこの部分は手動

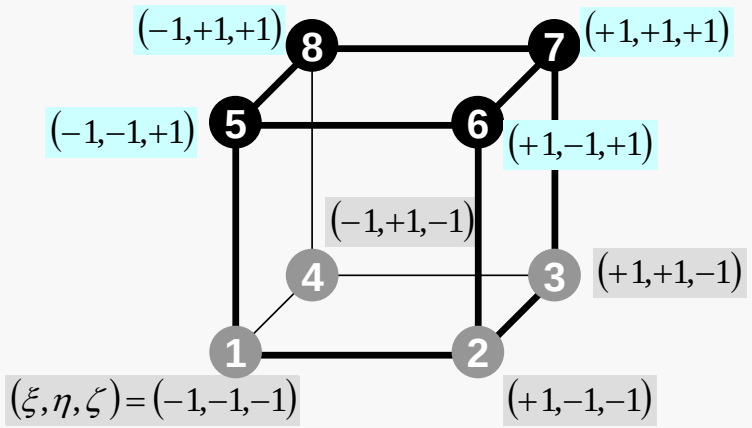
```
static void FIND_TS_NODE (int ip1,int ip2)

{
        int kk,icou;
        if( ip1 > ip2 ){
                for(kk=0;kk<INL[ip1-1];kk++){
                        if(ip2 == IAL[ip1-1][kk]) return;
                }
                icou=INL[ip1-1]+1;
                IAL[ip1-1][icou-1]=ip2;
                INL[ip1-1]=icou;
                return;
        }

        if( ip2 > ip1 ) {
                for(kk=0;kk<INU[ip1-1];kk++){
                        if(ip2 == IAU[ip1-1][kk]) return;
                }
                icou=INU[ip1-1]+1;
                IAU[ip1-1][icou-1]=ip2;
                INU[ip1-1]=icou;
                return;
        }
}
```

IAL，IAUに含まれていない場合は，INL・INUに1を加えてIAL，IAUに格納（節点番号は1から始まる）



行列コネクティビティ生成：MAT_CON0（3/4）

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

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

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

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



行列コネクティビティ生成：MAT_CON0（4/4）

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

        for(in=0;in<N;in++){
                NN=INL[in];
                for (k=0;k<NN;k++){
                        NCOL1[k]=IAL[in][k];
                }
                mSORT(NCOL1,NCOL2,NN);
                for(k=NN;k>0;k--){
                        IAL[in][NN-k]= NCOL1[NCOL2[k-1]-1];
                }

                NN=INU[in];
                for (k=0;k<NN;k++){
                        NCOL1[k]=IAU[in][k];
                }
                mSORT(NCOL1,NCOL2,NN);
                for(k=NN;k>0;k--){
                        IAU[in][NN-k]= NCOL1[NCOL2[k-1]-1];
                }
        }

}
```

各節点において，
IAL[i][k], IAU[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

```
#include <stdio.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE* fp_log;
void MAT_CON1()
{       int i,k,kk;

        indexL=(KINT*)allocate_vector(sizeof(KINT),NP+1);
        indexU=(KINT*)allocate_vector(sizeof(KINT),NP+1);
        for(i=0;i<NP+1;i++) indexL[i]=0;
        for(i=0;i<NP+1;i++) indexU[i]=0;

        for(i=0;i<NP;i++){
                indexL[i+1]=indexL[i]+INL[i];
                indexU[i+1]=indexU[i]+INU[i];
        }
        NPL=indexL[NP];
        NPU=indexU[NP];

        itemL=(KINT*)allocate_vector(sizeof(KINT),NPL);
        itemU=(KINT*)allocate_vector(sizeof(KINT),NPU);

        for(i=0;i<NP;i++){
                for(k=0;k<INL[i];k++){
                        kk=k+indexL[i];
                        itemL[kk]=IAL[i][k]-1;
                }
                for(k=0;k<INU[i];k++){
                        kk=k+indexU[i];
                        itemU[kk]=IAU[i][k]-1;
                }
        }
        deallocate_vector(INL);
        deallocate_vector(INU);
        deallocate_vector(IAL);
        deallocate_vector(IAU);
}
```

C

$$\mathrm{indexL}[i+1]=\sum_{k=0}^{i}\mathrm{INL}[k]$$

$$\mathrm{indexU}[i+1]=\sum_{k=0}^{i}\mathrm{INU}[k]$$

$$\mathrm{indexL}[0]=\mathrm{indexU}[0]=0$$

FORTRAN

$$\mathrm{indexL}[i]=\sum_{k=1}^{i}\mathrm{INL}[k]$$

$$\mathrm{indexU}[i]=\sum_{k=1}^{i}\mathrm{INU}[k]$$

$$\mathrm{indexL}[0]=\mathrm{indexU}[0]=0$$

CRS形式への変換：MAT_CON1

```
#include <stdio.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE* fp_log;
void MAT_CON1()
{       int i,k,kk;

        indexL=(KINT*)allocate_vector(sizeof(KINT),NP+1);
        indexU=(KINT*)allocate_vector(sizeof(KINT),NP+1);
        for(i=0;i<NP+1;i++) indexL[i]=0;
        for(i=0;i<NP+1;i++) indexU[i]=0;

        for(i=0;i<NP;i++){
                indexL[i+1]=indexL[i]+INL[i];
                indexU[i+1]=indexU[i]+INU[i];
        }
        NPL=indexL[NP];
        NPU=indexU[NP];

        itemL=(KINT*)allocate_vector(sizeof(KINT),NPL);
        itemU=(KINT*)allocate_vector(sizeof(KINT),NPU);

        for(i=0;i<NP;i++){
                for(k=0;k<INL[i];k++){
                        kk=k+indexL[i];
                        itemL[kk]=IAL[i][k]-1;
                }
                for(k=0;k<INU[i];k++){
                        kk=k+indexU[i];
                        itemU[kk]=IAU[i][k]-1;
                }
        }
        deallocate_vector(INL);
        deallocate_vector(INU);
        deallocate_vector(IAL);
        deallocate_vector(IAU);
}
```

NPL=indexL[NP]
itemLのサイズ
非ゼロ非対角ブロック（下三角）総数

NPU=indexU[NP]
itemUのサイズ
非ゼロ非対角ブロック（上三角）総数

CRS形式への変換：MAT_CON1

```
#include <stdio.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE* fp_log;
void MAT_CON1()
{       int i,k,kk;

        indexL=(KINT*)allocate_vector(sizeof(KINT),NP+1);
        indexU=(KINT*)allocate_vector(sizeof(KINT),NP+1);
        for(i=0;i<NP+1;i++) indexL[i]=0;
        for(i=0;i<NP+1;i++) indexU[i]=0;

        for(i=0;i<NP;i++){
                indexL[i+1]=indexL[i]+INL[i];
                indexU[i+1]=indexU[i]+INU[i];
        }
        NPL=indexL[NP];
        NPU=indexU[NP];

        itemL=(KINT*)allocate_vector(sizeof(KINT),NPL);
        itemU=(KINT*)allocate_vector(sizeof(KINT),NPU);

        for(i=0;i<NP;i++){
                for(k=0;k<INL[i];k++){
                        kk=k+indexL[i];
                        itemL[kk]=IAL[i][k]-1;
                }
                for(k=0;k<INU[i];k++){
                        kk=k+indexU[i];
                        itemU[kk]=IAU[i][k]-1;
                }
        }
        deallocate_vector(INL);
        deallocate_vector(INU);
        deallocate_vector(IAL);
        deallocate_vector(IAU);
}
```

itemL，itemUに「0」から始まる節点番号を記憶

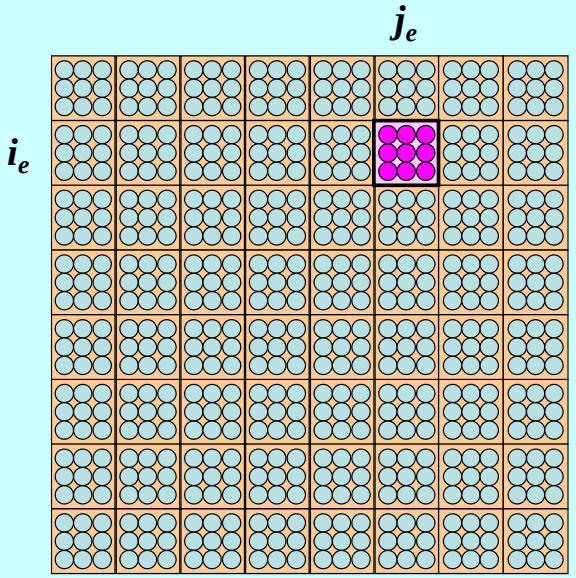
全体処理

```
#include <stdio.h>
#include <stdlib.h>
FILE* fp_log;
#define GLOBAL_VALUE_DEFINE
#include "pfem_util.h"
//#include "solver33.h"
extern void PFEM_INIT(int,char**);
extern void INPUT_CNTL();
extern void INPUT_GRID();
extern void MAT_CON0();
extern void MAT_CON1();
extern void MAT_ASS_MAIN();
extern void MAT_ASS_BC();
extern void SOLVE33();
extern void RECOVER_STRESS();
extern void OUTPUT_UCD();
extern void PFEM_FINALIZE();
int main(int argc,char* argv[])
{
        double START_TIME,END_TIME;

        PFEM_INIT(argc,argv);

        INPUT_CNTL();
        INPUT_GRID();

        MAT_CON0();
        MAT_CON1();

        MAT_ASS_MAIN();
        MAT_ASS_BC()  ;

        SOLVE33();

        RECOVER_STRESS();
        OUTPUT_UCD()    ;
        PFEM_FINALIZE() ;
```

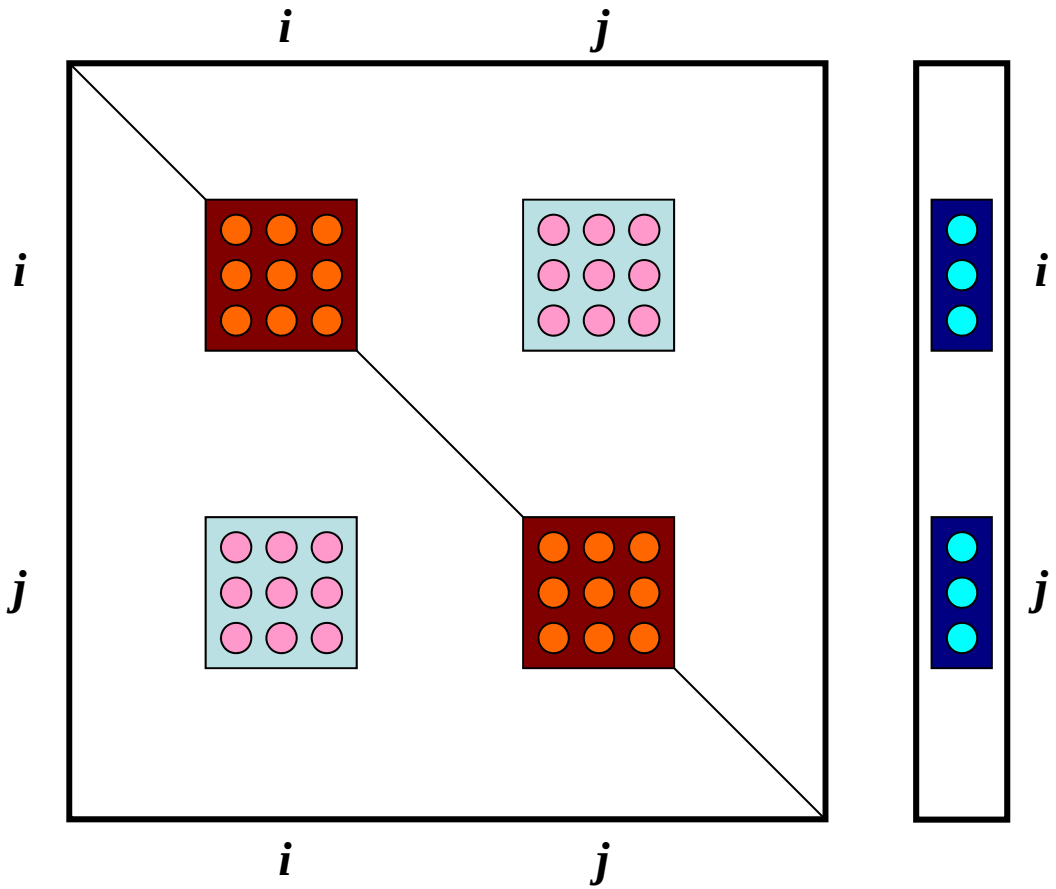
MAT_ASS_MAIN：全体構成

```
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
      Aip, jpのitemL,itemU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
```



ブロックとして記憶



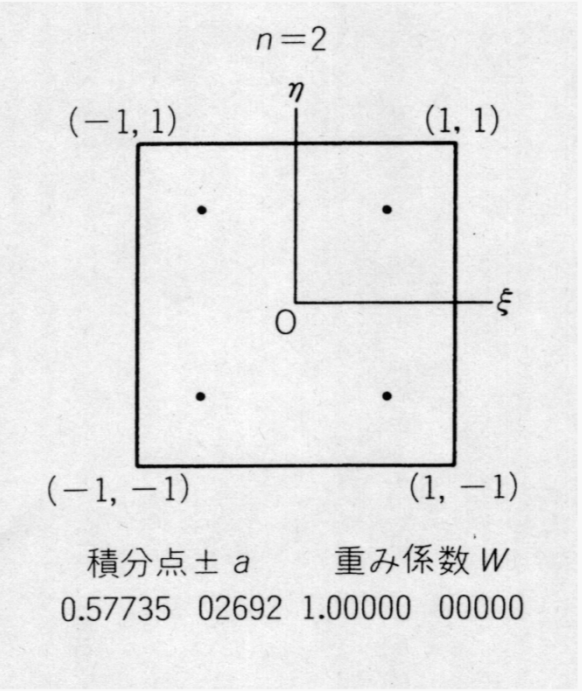
係数行列：MAT_ASS_MAIN（1/7）

```
#include <stdio.h>
#include <math.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE *fp_log;
extern void JACOBI();
void MAT_ASS_MAIN()
{
            int i,k,kk;
            int ip,jp,kp;
            int ipn,jpn,kpn;
            int icel;
            int ie,je;
            int iiS,iiE;
            int in1,in2,in3,in4,in5,in6,in7,in8;
            double valA,valB,valX;
            double QP1,QM1,EP1,EM1,TP1,TM1;
            double X1,X2,X3,X4,X5,X6,X7,X8;
            double Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8;
            double Z1,Z2,Z3,Z4,Z5,Z6,Z7,Z8;
            double PNXi,PNYi,PNZi,PNXj,PNYj,PNZj;
            double VOL;
            double coef;
            double a11,a12,a13,a21,a22,a23,a31,a32,a33;

            KINT nodLOCAL[8];

            AL=(KREAL*) allocate_vector(sizeof(KREAL),9*NPL);          下三角ブロック
            AU=(KREAL*) allocate_vector(sizeof(KREAL),9*NPU);          上三角ブロック
            B =(KREAL*) allocate_vector(sizeof(KREAL),3*NP );          右辺ベクトル
            D =(KREAL*) allocate_vector(sizeof(KREAL),9*NP);           対角ブロック
            X =(KREAL*) allocate_vector(sizeof(KREAL),3*NP);           未知数ベクトル

            for(i=0;i<9*NPL;i++) AL[i]=0.0;
            for(i=0;i<9*NPU;i++) AU[i]=0.0;
            for(i=0;i<3*NP;i++) B[i]=0.0;
            for(i=0;i<9*NP;i++) D[i]=0.0;
            for(i=0;i<3*NP;i++) X[i]=0.0;
```

係数行列：MAT ASS MAIN（2/7）

POS: 積分点座標
WEI: 重み係数

```
      WEI[0]= 1.0000000000e0;
      WEI[1]= 1.0000000000e0;

      POS[0]= -0.5773502692e0;
      POS[1]=  0.5773502692e0;

/***
      INIT.
      PNQ   - 1st-order derivative of shape function by QSI
      PNE   - 1st-order derivative of shape function by ETA
      PNT   - 1st-order derivative of shape function by ZET
***/
      valA=            POISSON /      (1.e0-POISSON);
      valB= (1.e0-2.e0*POISSON)/(2.e0*(1.e0-POISSON));
      valX= ELAST*(1.e0-POISSON)/((1.e0+POISSON)*(1.e0-2.e0*POISSON));

      valA= valA * valX;
      valB= valB * valX;

      for(ip=0;ip<2;ip++){
        for(jp=0;jp<2;jp++){
          for(kp=0;kp<2;kp++){
            QP1= 1.e0 + POS[ip];
            QM1= 1.e0 - POS[ip];
            EP1= 1.e0 + POS[jp];
            EM1= 1.e0 - POS[jp];
            TP1= 1.e0 + POS[kp];
            TM1= 1.e0 - POS[kp];

            SHAPE[ip][jp][kp][0]= 08th * QM1 * EM1 * TM1;
            SHAPE[ip][jp][kp][1]= 08th * QP1 * EM1 * TM1;
            SHAPE[ip][jp][kp][2]= 08th * QP1 * EP1 * TM1;
            SHAPE[ip][jp][kp][3]= 08th * QM1 * EP1 * TM1;
            SHAPE[ip][jp][kp][4]= 08th * QM1 * EM1 * TP1;
            SHAPE[ip][jp][kp][5]= 08th * QP1 * EM1 * TP1;
            SHAPE[ip][jp][kp][6]= 08th * QP1 * EP1 * TP1;
            SHAPE[ip][jp][kp][7]= 08th * QM1 * EP1 * TP1;
```



係数行列：MAT ASS MAIN（2/7）

```
            WEI[0]= 1.0000000000e0;
            WEI[1]= 1.0000000000e0;

            POS[0]= -0.5773502692e0;
            POS[1]=  0.5773502692e0;

/***
            INIT.
            PNQ   - 1st-order derivative of shape function by QSI
            PNE   - 1st-order derivative of shape function by ETA
            PNT   - 1st-order derivative of shape function by ZET
***/
            valA=            POISSON /      (1.e0-POISSON);
            valB= (1.e0-2.e0*POISSON)/(2.e0*(1.e0-POISSON));
            valX= ELAST*(1.e0-POISSON)/((1.e0+POISSON)*(1.e0-2.e0*POISSON));

            valA= valA * valX;
            valB= valB * valX;

            for(ip=0;ip<2;ip++){
                for(jp=0;jp<2;jp++){
                    for(kp=0;kp<2;kp++){
                        QP1= 1.e0 + POS[ip];
                        QM1= 1.e0 - POS[ip];
                        EP1= 1.e0 + POS[jp];
                        EM1= 1.e0 - POS[jp];
                        TP1= 1.e0 + POS[kp];
                        TM1= 1.e0 - POS[kp];

                        SHAPE[ip][jp][kp][0]= 08th * QM1 * EM1 * TM1;
                        SHAPE[ip][jp][kp][1]= 08th * QP1 * EM1 * TM1;
                        SHAPE[ip][jp][kp][2]= 08th * QP1 * EP1 * TM1;
                        SHAPE[ip][jp][kp][3]= 08th * QM1 * EP1 * TM1;
                        SHAPE[ip][jp][kp][4]= 08th * QM1 * EM1 * TP1;
                        SHAPE[ip][jp][kp][5]= 08th * QP1 * EM1 * TP1;
                        SHAPE[ip][jp][kp][6]= 08th * QP1 * EP1 * TP1;
                        SHAPE[ip][jp][kp][7]= 08th * QM1 * EP1 * TP1;
```

ひずみ⇒応力関係

$$\begin{Bmatrix}\sigma_x\\ \sigma_y\\ \sigma_z\\ \tau_{xy}\\ \tau_{yz}\\ \tau_{zx}\end{Bmatrix} = \frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0\\ \nu & 1-\nu & \nu & 0 & 0 & 0\\ \nu & \nu & 1-\nu & 0 & 0 & 0\\ 0 & 0 & 0 & \frac{1}{2}(1-2\nu) & 0 & 0\\ 0 & 0 & 0 & 0 & \frac{1}{2}(1-2\nu) & 0\\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2}(1-2\nu)\end{bmatrix}\begin{Bmatrix}\varepsilon_x\\ \varepsilon_y\\ \varepsilon_z\\ \gamma_{xy}\\ \gamma_{yz}\\ \gamma_{zx}\end{Bmatrix}$$

$$[D]$$

$$\{\sigma\} = [D]\{\varepsilon\}$$

係数行列：MAT ASS MAIN（2/7）

```
            WEI[0]= 1.0000000000e0;
            WEI[1]= 1.0000000000e0;

            POS[0]= -0.5773502692e0;
            POS[1]=  0.5773502692e0;

/***
            INIT.
            PNQ   - 1st-order derivative of shape function by QSI
            PNE   - 1st-order derivative of shape function by ETA
            PNT   - 1st-order derivative of shape function by ZET
***/
            valA=            POISSON /      (1.e0-POISSON);
            valB= (1.e0-2.e0*POISSON)/(2.e0*(1.e0-POISSON));
            valX= ELAST*(1.e0-POISSON)/((1.e0+POISSON)*(1.e0-2.e0*POISSON));

            valA= valA * valX;
            valB= valB * valX;

            for(ip=0;ip<2;ip++){
                for(jp=0;jp<2;jp++){
                    for(kp=0;kp<2;kp++){
                        QP1= 1.e0 + POS[ip];
                        QM1= 1.e0 - POS[ip];
                        EP1= 1.e0 + POS[jp];
                        EM1= 1.e0 - POS[jp];
                        TP1= 1.e0 + POS[kp];
                        TM1= 1.e0 - POS[kp];

                        SHAPE[ip][jp][kp][0]= 08th * QM1 * EM1 * TM1;
                        SHAPE[ip][jp][kp][1]= 08th * QP1 * EM1 * TM1;
                        SHAPE[ip][jp][kp][2]= 08th * QP1 * EP1 * TM1;
                        SHAPE[ip][jp][kp][3]= 08th * QM1 * EP1 * TM1;
                        SHAPE[ip][jp][kp][4]= 08th * QM1 * EM1 * TP1;
                        SHAPE[ip][jp][kp][5]= 08th * QP1 * EM1 * TP1;
                        SHAPE[ip][jp][kp][6]= 08th * QP1 * EP1 * TP1;
                        SHAPE[ip][jp][kp][7]= 08th * QM1 * EP1 * TP1;
```

$$\mathrm{valX} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}$$

$$\mathrm{valA} = \frac{\nu}{(1-\nu)} \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}$$

$$\mathrm{valB} = \frac{(1-2\nu)}{2(1-\nu)} \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}$$

係数行列：MAT ASS MAIN（2/7）

```
            WEI[0]= 1.0000000000e0;
            WEI[1]= 1.0000000000e0;

            POS[0]= -0.5773502692e0;
            POS[1]=  0.5773502692e0;

/***
            INIT.
            PNQ   - 1st-order derivative of shape function by QSI
            PNE   - 1st-order derivative of shape function by ETA
            PNT   - 1st-order derivative of shape function by ZET
***/
            valA=            POISSON /      (1.e0-POISSON);
            valB= (1.e0-2.e0*POISSON)/(2.e0*(1.e0-POISSON));
            valX= ELAST*(1.e0-POISSON)/((1.e0+POISSON)*(1.e0-2.e0*POISSON));

            valA= valA * valX;
            valB= valB * valX;

            for(ip=0;ip<2;ip++){
                for(jp=0;jp<2;jp++){
                    for(kp=0;kp<2;kp++){
                        QP1= 1.e0 + POS[ip];
                        QM1= 1.e0 - POS[ip];
                        EP1= 1.e0 + POS[jp];
                        EM1= 1.e0 - POS[jp];
                        TP1= 1.e0 + POS[kp];
                        TM1= 1.e0 - POS[kp];

                        SHAPE[ip][jp][kp][0]= 08th * QM1 * EM1 * TM1;
                        SHAPE[ip][jp][kp][1]= 08th * QP1 * EM1 * TM1;
                        SHAPE[ip][jp][kp][2]= 08th * QP1 * EP1 * TM1;
                        SHAPE[ip][jp][kp][3]= 08th * QM1 * EP1 * TM1;
                        SHAPE[ip][jp][kp][4]= 08th * QM1 * EM1 * TP1;
                        SHAPE[ip][jp][kp][5]= 08th * QP1 * EM1 * TP1;
                        SHAPE[ip][jp][kp][6]= 08th * QP1 * EP1 * TP1;
                        SHAPE[ip][jp][kp][7]= 08th * QM1 * EP1 * TP1;
```

$$\mathrm{valX} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}$$

$$\mathrm{valA} = \frac{\nu}{(1-\nu)} \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} = \frac{E\nu}{(1+\nu)(1-2\nu)}$$

$$\mathrm{valB} = \frac{(1-2\nu)}{2(1-\nu)} \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} = \frac{E}{2(1+\nu)}$$

ひずみ⇒応力関係

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{Bmatrix} = \begin{bmatrix} \mathrm{valX} & \mathrm{valA} & \mathrm{valA} & 0 & 0 & 0 \\ \mathrm{valA} & \mathrm{valX} & \mathrm{valA} & 0 & 0 & 0 \\ \mathrm{valA} & \mathrm{valA} & \mathrm{valX} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{valB} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathrm{valB} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathrm{valB} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{Bmatrix}$$

$$[D]$$

$$\{\sigma\} = [D]\{\varepsilon\}$$

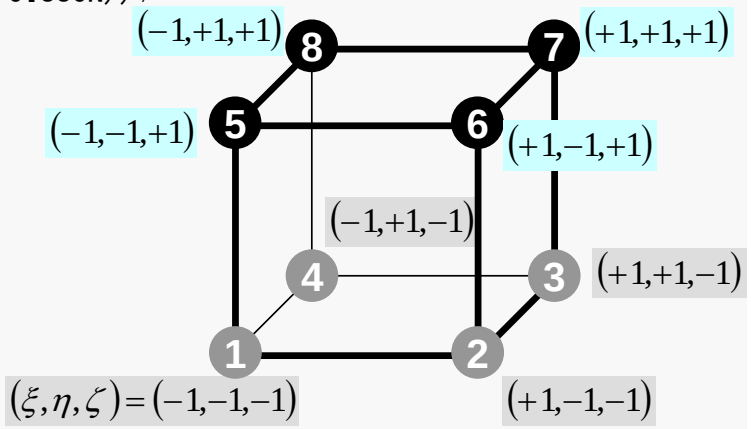
係数行列：MAT ASS MAIN（2/7）

```
            WEI[0]= 1.0000000000e0;
            WEI[1]= 1.0000000000e0;

            POS[0]= -0.5773502692e0;
            POS[1]=  0.5773502692e0;

/***
            INIT.
            PNQ   - 1st-order derivative of shape function by QSI
            PNE   - 1st-order derivative of shape function by ETA
            PNT   - 1st-order derivative of shape function by ZET
***/
            valA=            POISSON /      (1.e0-POISSON);
            valB= (1.e0-2.e0*POISSON)/(2.e0*(1.e0-POISSON));
            valX= ELAST*(1.e0-POISSON)/((1.e0+POISSON)*(1.e0-2.e0*POISSON));

            valA= valA * valX;
            valB= valB * valX;

            for(ip=0;ip<2;ip++){
                for(jp=0;jp<2;jp++){
                    for(kp=0;kp<2;kp++){
                        QP1= 1.e0 + POS[ip];
                        QM1= 1.e0 - POS[ip];
                        EP1= 1.e0 + POS[jp];
                        EM1= 1.e0 - POS[jp];
                        TP1= 1.e0 + POS[kp];
                        TM1= 1.e0 - POS[kp];

                        SHAPE[ip][jp][kp][0]= 08th * QM1 * EM1 * TM1;
                        SHAPE[ip][jp][kp][1]= 08th * QP1 * EM1 * TM1;
                        SHAPE[ip][jp][kp][2]= 08th * QP1 * EP1 * TM1;
                        SHAPE[ip][jp][kp][3]= 08th * QM1 * EP1 * TM1;
                        SHAPE[ip][jp][kp][4]= 08th * QM1 * EM1 * TP1;
                        SHAPE[ip][jp][kp][5]= 08th * QP1 * EM1 * TP1;
                        SHAPE[ip][jp][kp][6]= 08th * QP1 * EP1 * TP1;
                        SHAPE[ip][jp][kp][7]= 08th * QM1 * EP1 * TP1;
```

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

係数行列：MAT ASS MAIN（2/7）

```
            WEI[0]= 1.0000000000e0;
            WEI[1]= 1.0000000000e0;

            POS[0]= -0.5773502692e0;
            POS[1]=  0.5773502692e0;

/***
            INIT.
            PNQ   - 1st-order derivative of shape function by QSI
            PNE   - 1st-order derivative of shape function by ETA
            PNT   - 1st-order derivative of shape function by ZET
***/
            valA=            POISSON /      (1.e0-POISSON);
            valB= (1.e0-2.e0*POISSON)/(2.e0*(1.e0-POISSON));
            valX= ELAST*(1.e0-POISSON)/((1.e0+POISSON)*(1.e0-2.e0*POISSON));

            valA= valA * valX;
            valB= valB * valX;

            for(ip=0;ip<2;ip++){
                for(jp=0;jp<2;jp++){
                    for(kp=0;kp<2;kp++){
                        QP1= 1.e0 + POS[ip];
                        QM1= 1.e0 - POS[ip];
                        EP1= 1.e0 + POS[jp];
                        EM1= 1.e0 - POS[jp];
                        TP1= 1.e0 + POS[kp];
                        TM1= 1.e0 - POS[kp];

                        SHAPE[ip][jp][kp][0]= 0.8e0 * QM1 * EM1 * TM1;
                        SHAPE[ip][jp][kp][1]= 0.8e0 * QP1 * EM1 * TM1;
                        SHAPE[ip][jp][kp][2]= 0.8e0 * QP1 * EP1 * TM1;
                        SHAPE[ip][jp][kp][3]= 0.8e0 * QM1 * EP1 * TM1;
                        SHAPE[ip][jp][kp][4]= 0.8e0 * QM1 * EM1 * TP1;
                        SHAPE[ip][jp][kp][5]= 0.8e0 * QP1 * EM1 * TP1;
                        SHAPE[ip][jp][kp][6]= 0.8e0 * QP1 * EP1 * TP1;
                        SHAPE[ip][jp][kp][7]= 0.8e0 * QM1 * EP1 * TP1;
```



係数行列：MAT ASS MAIN（2/7）

```
            WEI[0]= 1.0000000000e0;
            WEI[1]= 1.0000000000e0;

            POS[0]= -0.5773502692e0;
            POS[1]=  0.5773502692e0;

/***
            INIT.
            PNQ   - 1st-order derivative of shape function by QSI
            PNE   - 1st-order derivative of shape function by ETA
            PNT   - 1st-order derivative of shape function by ZET
***/
            valA=             POISSON /      (1.e0-POISSON);
            valB= (1.e0-2.e0*POISSON)/(2.e0*(1.e0-POISSON));
            valX= ELAST*(1.e0-POISSON)/((1.e0+POISSON)*(1.e0-2.e0*POISS

            valA= valA * valX;
            valB= valB * valX;

            for(ip=0;ip<2;ip++){
                for(jp=0;jp<2;jp++){
                    for(kp=0;kp<2;kp++){
                        QP1= 1.e0 + POS[ip];
                        QM1= 1.e0 - POS[ip];
                        EP1= 1.e0 + POS[jp];
                        EM1= 1.e0 - POS[jp];
                        TP1= 1.e0 + POS[kp];
                        TM1= 1.e0 - POS[kp];

                        SHAPE[ip][jp][kp][0]= 08th * QM1 * EM1 * TM1;
                        SHAPE[ip][jp][kp][1]= 08th * QP1 * EM1 * TM1;
                        SHAPE[ip][jp][kp][2]= 08th * QP1 * EP1 * TM1;
                        SHAPE[ip][jp][kp][3]= 08th * QM1 * EP1 * TM1;
                        SHAPE[ip][jp][kp][4]= 08th * QM1 * EM1 * TP1;
                        SHAPE[ip][jp][kp][5]= 08th * QP1 * EM1 * TP1;
                        SHAPE[ip][jp][kp][6]= 08th * QP1 * EP1 * TP1;
                        SHAPE[ip][jp][kp][7]= 08th * QM1 * 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（3/7）

```
                PNQ[jp][kp][0]= - 08th * EM1 * TM1;
                PNQ[jp][kp][1]= + 08th * EM1 * TM1;
                PNQ[jp][kp][2]= + 08th * EP1 * TM1;
                PNQ[jp][kp][3]= - 08th * EP1 * TM1;
                PNQ[jp][kp][4]= - 08th * EM1 * TP1;
                PNQ[jp][kp][5]= + 08th * EM1 * TP1;
                PNQ[jp][kp][6]= + 08th * EP1 * TP1;
                PNQ[jp][kp][7]= - 08th * EP1 * TP1;
                PNE[ip][kp][0]= - 08th * QM1 * TM1;
                PNE[ip][kp][1]= - 08th * QP1 * TM1;
                PNE[ip][kp][2]= + 08th * QP1 * TM1;
                PNE[ip][kp][3]= + 08th * QM1 * TM1;
                PNE[ip][kp][4]= - 08th * QM1 * TP1;
                PNE[ip][kp][5]= - 08th * QP1 * TP1;
                PNE[ip][kp][6]= + 08th * QP1 * TP1;
                PNE[ip][kp][7]= + 08th * QM1 * TP1;
                PNT[ip][jp][0]= - 08th * QM1 * EM1;
                PNT[ip][jp][1]= - 08th * QP1 * EM1;
                PNT[ip][jp][2]= - 08th * QP1 * EP1;
                PNT[ip][jp][3]= - 08th * QM1 * EP1;
                PNT[ip][jp][4]= + 08th * QM1 * EM1;
                PNT[ip][jp][5]= + 08th * QP1 * EM1;
                PNT[ip][jp][6]= + 08th * QP1 * EP1;
                PNT[ip][jp][7]= + 08th * QM1 * EP1;
                            }
                }
}

for( icel=0;icel< ICELTOT;icel++){
                in1=ICELNOD[icel][0];
                in2=ICELNOD[icel][1];
                in3=ICELNOD[icel][2];
                in4=ICELNOD[icel][3];
                in5=ICELNOD[icel][4];
                in6=ICELNOD[icel][5];
                in7=ICELNOD[icel][6];
                in8=ICELNOD[icel][7];
```

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

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

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

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

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

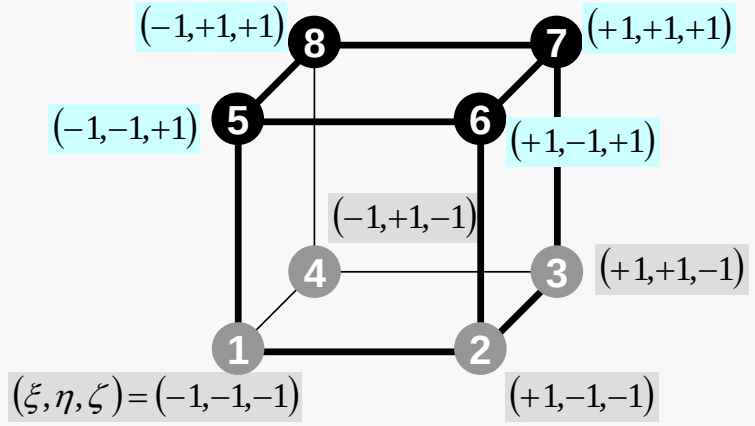
$$\frac{\partial N_3}{\partial \xi}(\xi_i,\eta_j,\zeta_k)=+\frac{1}{8}(1+\eta_j)(1-\zeta_k)$$

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

(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（3/7）

```
                PNQ[jp][kp][0]= - 08th * EM1 * TM1;
                PNQ[jp][kp][1]= + 08th * EM1 * TM1;
                PNQ[jp][kp][2]= + 08th * EP1 * TM1;
                PNQ[jp][kp][3]= - 08th * EP1 * TM1;
                PNQ[jp][kp][4]= - 08th * EM1 * TP1;
                PNQ[jp][kp][5]= + 08th * EM1 * TP1;
                PNQ[jp][kp][6]= + 08th * EP1 * TP1;
                PNQ[jp][kp][7]= - 08th * EP1 * TP1;
                PNE[ip][kp][0]= - 08th * QM1 * TM1;
                PNE[ip][kp][1]= - 08th * QP1 * TM1;
                PNE[ip][kp][2]= + 08th * QP1 * TM1;
                PNE[ip][kp][3]= + 08th * QM1 * TM1;
                PNE[ip][kp][4]= - 08th * QM1 * TP1;
                PNE[ip][kp][5]= - 08th * QP1 * TP1;
                PNE[ip][kp][6]= + 08th * QP1 * TP1;
                PNE[ip][kp][7]= + 08th * QM1 * TP1;
                PNT[ip][jp][0]= - 08th * QM1 * EM1;
                PNT[ip][jp][1]= - 08th * QP1 * EM1;
                PNT[ip][jp][2]= - 08th * QP1 * EP1;
                PNT[ip][jp][3]= - 08th * QM1 * EP1;
                PNT[ip][jp][4]= + 08th * QM1 * EM1;
                PNT[ip][jp][5]= + 08th * QP1 * EM1;
                PNT[ip][jp][6]= + 08th * QP1 * EP1;
                PNT[ip][jp][7]= + 08th * QM1 * EP1;
                            }
                }
}

for( icel=0;icel< ICELTOT;icel++){
                in1=ICELNOD[icel][0];
                in2=ICELNOD[icel][1];
                in3=ICELNOD[icel][2];
                in4=ICELNOD[icel][3];
                in5=ICELNOD[icel][4];
                in6=ICELNOD[icel][5];
                in7=ICELNOD[icel][6];
                in8=ICELNOD[icel][7];
```



係数行列：MAT ASS MAIN（4/7）

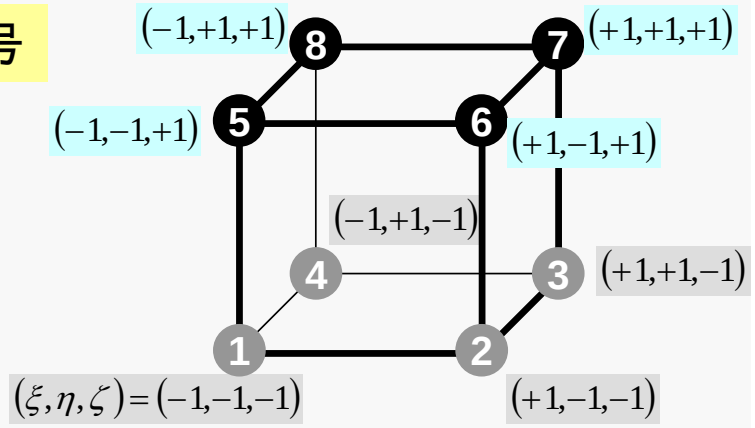
```
/**
 ** JACOBIAN & INVERSE JACOBIAN
**/
                        nodLOCAL[0]= in1;
                        nodLOCAL[1]= in2;
                        nodLOCAL[2]= in3;
                        nodLOCAL[3]= in4;
                        nodLOCAL[4]= in5;
                        nodLOCAL[5]= in6;
                        nodLOCAL[6]= in7;
                        nodLOCAL[7]= in8;

                        X1=XYZ[in1-1][0];
                        X2=XYZ[in2-1][0];
                        X3=XYZ[in3-1][0];
                        X4=XYZ[in4-1][0];
                        X5=XYZ[in5-1][0];
                        X6=XYZ[in6-1][0];
                        X7=XYZ[in7-1][0];
                        X8=XYZ[in8-1][0];

                        Y1=XYZ[in1-1][1];
                        Y2=XYZ[in2-1][1];
                        Y3=XYZ[in3-1][1];
                        Y4=XYZ[in4-1][1];
                        Y5=XYZ[in5-1][1];
                        Y6=XYZ[in6-1][1];
                        Y7=XYZ[in7-1][1];
                        Y8=XYZ[in8-1][1];

                        Z1=XYZ[in1-1][2];
                        Z2=XYZ[in2-1][2];
                        Z3=XYZ[in3-1][2];
                        Z4=XYZ[in4-1][2];
                        Z5=XYZ[in5-1][2];
                        Z6=XYZ[in6-1][2];
                        Z7=XYZ[in7-1][2];
                        Z8=XYZ[in8-1][2];

                        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

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（5/7）

```
/**
          CONSTRUCT the GLOBAL MATRIX
**/
                    for(ie=0;ie<8;ie++){
                        ip=nodLOCAL[ie];

                        for(je=0;je<8;je++){
                            jp=nodLOCAL[je];

                            kk=0;
                            if( jp > ip ){
                                iiS=indexU[ip-1];
                                iiE=indexU[ip  ];
                                for( k=iiS;k<iiE;k++){
                                    if( itemU[k] == jp-1 ){
                                        kk=k;
                                        break;
                                                        }
                                                    }
                                        }

                            if( jp < ip ){
                                iiS=indexL[ip-1];
                                iiE=indexL[ip  ];
                                for( k=iiS;k<iiE;k++){
                                    if( itemL[k] == jp-1 ){
                                        kk=k;
                                        break;
                                                        }
                                                    }
                                        }

                            PNXi= 0.e0;
                            PNYi= 0.e0;
                            PNZi= 0.e0;
                            PNXj= 0.e0;
                            PNYj= 0.e0;
                            PNZj= 0.e0;

                            VOL= 0.e0;
```

全体行列の非対角ブロック

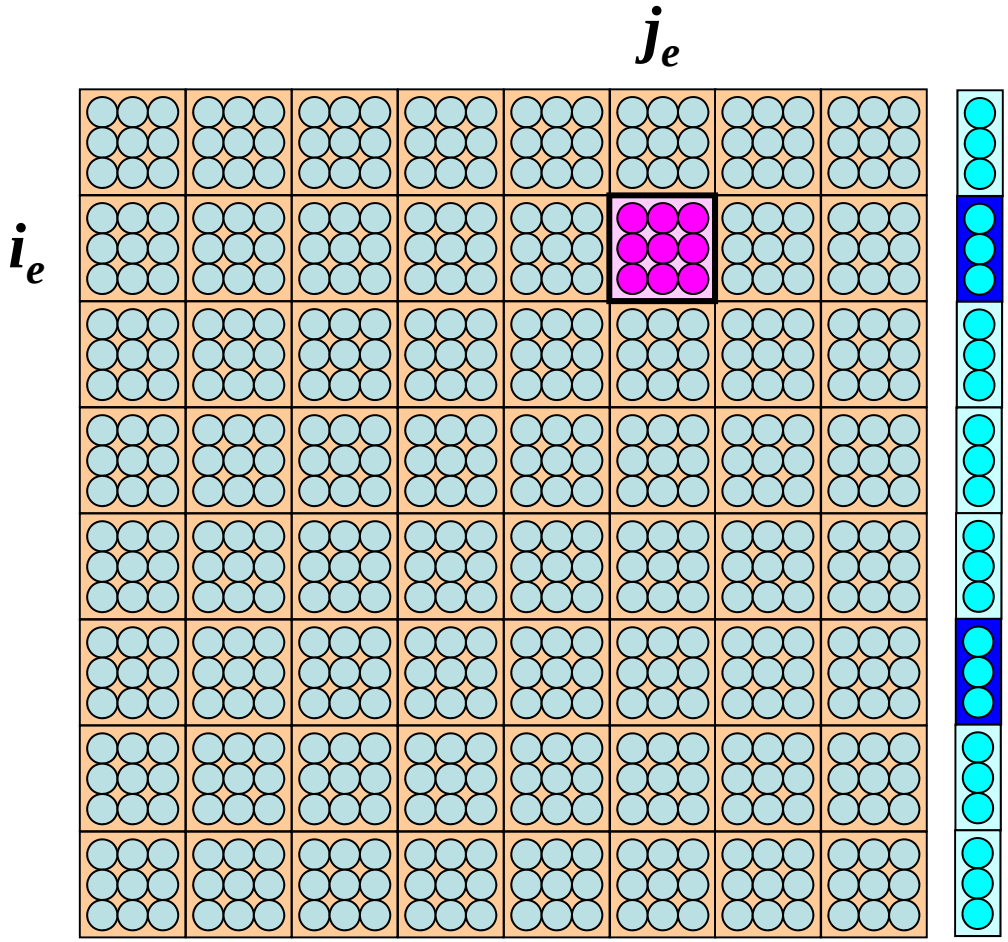
$$A_{ip,jp}$$

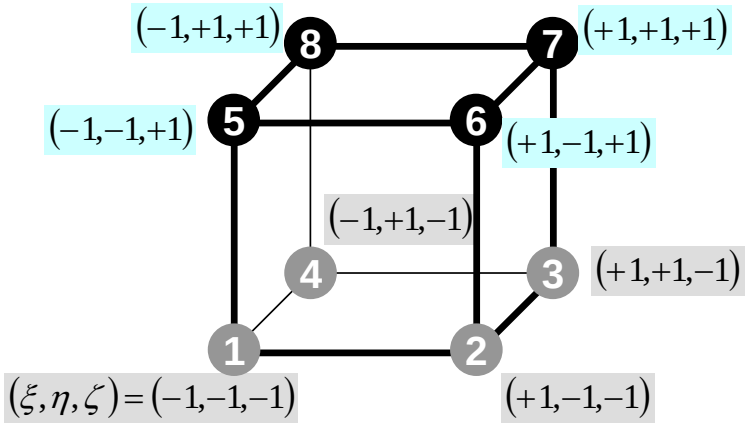
kk：itemL, itemUにおけるアドレス
　0から始まる
k: 0から始まる

ip,jp: 1から始まる

要素マトリクス：24×24行列

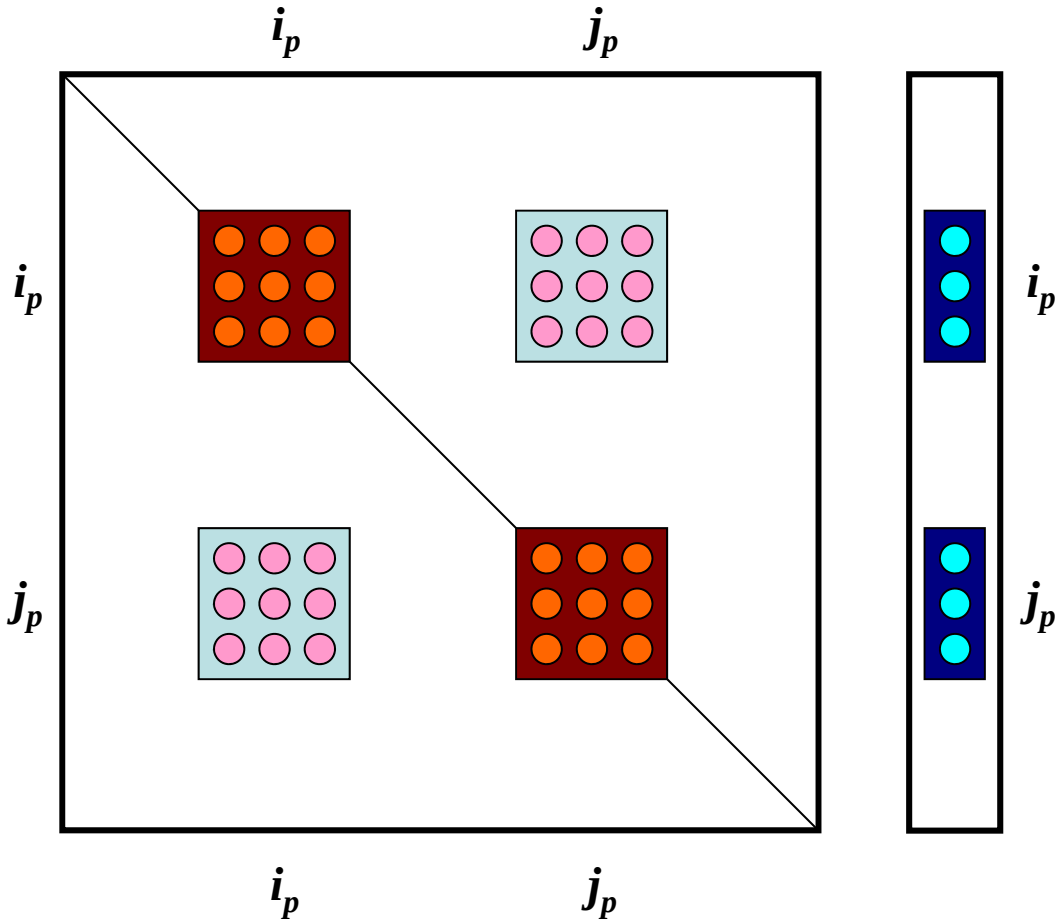
各節点上の（u,v,w）成分が物理的にも強くカップルしているので3自由度をまとめて扱う：8×8行列





$$\begin{bmatrix} a_{i_e j_e 11} & a_{i_e j_e 12} & a_{i_e j_e 13} \\ a_{i_e j_e 21} & a_{i_e j_e 22} & a_{i_e j_e 23} \\ a_{i_e j_e 31} & a_{i_e j_e 32} & a_{i_e j_e 33} \end{bmatrix} (i_e, j_e = 1 \ldots 8)$$

全体マトリクス



係数行列：MAT ASS MAIN（5/7）

```
/**
        CONSTRUCT the GLOBAL MATRIX
**/
                for(ie=0;ie<8;ie++){
                    ip=nodLOCAL[ie];

                    for(je=0;je<8;je++){
                        jp=nodLOCAL[je];

                        kk=0;
                        if( jp > ip ){
                            iiS=indexU[ip-1];
                            iiE=indexU[ip  ];
                            for( k=iiS;k<=iiE;k++){
                                if( itemU[k] == jp-1 ){
                                    kk=k;
                                    break;
                                }
                            }
                        }

                        if( jp < ip ){
                            iiS=indexL[ip-1];
                            iiE=indexL[ip  ];
                            for( k=iiS;k<iiE;k++){
                                if( itemL[k] == jp-1 ){
                                    kk=k;
                                    break;
                                }
                            }
                        }

                        PNXi= 0.e0;
                        PNYi= 0.e0;
                        PNZi= 0.e0;
                        PNXj= 0.e0;
                        PNYj= 0.e0;
                        PNZj= 0.e0;

                        VOL= 0.e0;
```

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：itemL，itemUにおけるアドレス
　0から始まる
k：0から始まる

ip,jp：1から始まる

係数行列：MAT_ASS_MAIN（6/7）

```
for(kpn=0;kpn<2;kpn++){
    for(jpn=0;jpn<2;jpn++){
        for(ipn=0;ipn<2;ipn++){
            coef= -fabs(DETJ[ipn][jpn][kpn])*WEI[ipn]*WEI[jpn]*WEI[kpn];

            VOL+=coef;
            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];

            a11= (valX*PNXi*PNXj+valB*(PNYi*PNYj+PNZi*PNZj))*coef;
            a22= (valX*PNYi*PNYj+valB*(PNZi*PNZj+PNXi*PNXj))*coef;
            a33= (valX*PNZi*PNZj+valB*(PNXi*PNXj+PNYi*PNYj))*coef;

            a12= (valA*PNXi*PNYj + valB*PNXj*PNYi)*coef;
            a13= (valA*PNXi*PNZj + valB*PNXj*PNZi)*coef;
            a21= (valA*PNYi*PNXj + valB*PNYj*PNXi)*coef;
            a23= (valA*PNYi*PNZj + valB*PNYj*PNZi)*coef;
            a31= (valA*PNZi*PNXj + valB*PNZj*PNXi)*coef;
            a32= (valA*PNZi*PNYj + valB*PNZj*PNYi)*coef;

            if (jp > ip) {
              AU[9*kk  ]+=a11;
              AU[9*kk+1]+=a12;
              AU[9*kk+2]+=a13;
              AU[9*kk+3]+=a21;
              AU[9*kk+4]+=a22;
              AU[9*kk+5]+=a23;
              AU[9*kk+6]+=a31;
              AU[9*kk+7]+=a32;
              AU[9*kk+8]+=a33;
              }
```

係数行列：MAT ASS MAIN（6/7）

$$-\int_V \left\{ D\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + b\left(\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right) \right\} dV =$$

$$-\iiint \left\{ D\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + b\left(\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right) \right\} dxdydz =$$

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

$$f(\xi,\eta,\zeta)$$

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

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

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

```
for (kpn=0;kpn<2;kpn++) {
    for (jpn=0;jpn<2;jpn++) {
        for (ipn=0;ipn<2;ipn++) {
            coef= -fabs(DETJ[ipn][jpn][kpn])*WEI[ipn]*WEI[jpn]*WEI[kpn];

                                                    NYi*PNYj+PNZi*PNZj))*coef;
                                                    NZi*PNZj+PNXi*PNXj))*coef;
                                                    NXi*PNXj+PNYi*PNYj))*coef;

                                                    PNXj*PNYi)*coef;
                                                    PNXj*PNZi)*coef;
                                                    PNYj*PNXi)*coef;
                                                    PNYj*PNZi)*coef;
a31= (valA*PNZi*PNXj + valB*PNZj*PNXi)*coef;
a32= (valA*PNZi*PNYj + valB*PNZj*PNYi)*coef;

if (jp > ip) {
    AU[9*kk  ]+=a11;
    AU[9*kk+1]+=a12;
    AU[9*kk+2]+=a13;
    AU[9*kk+3]+=a21;
    AU[9*kk+4]+=a22;
    AU[9*kk+5]+=a23;
    AU[9*kk+6]+=a31;
    AU[9*kk+7]+=a32;
    AU[9*kk+8]+=a33;
```

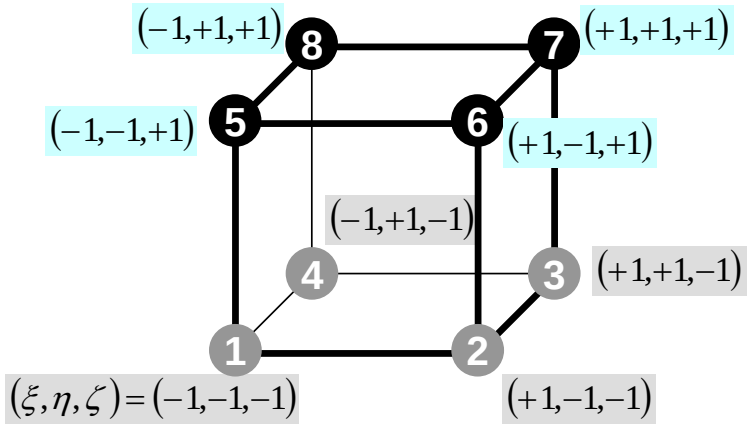
係数行列：MAT_ASS_MAIN（6/7）

```
for(kpn=0;kpn<2;kpn++){
  for(jpn=0;jpn<2;jpn++){
    for(ipn=0;ipn<2;ipn++){
      coef= -fabs(DETJ[ipn][jpn][kpn])*WEI[ipn]*WEI[jpn]*WEI[kpn];

      VOL+=coef;
      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];

      a11= (valX*PNXi*PNXj+valB*(PNYi*PNYj+PNZi*PNZj))*coef;
      a22= (valX*PNYi*PNYj+valB*(PNZi*PNZj+PNXi*PNXj))*coef;
      a33= (valX*PNZi*PNZj+valB*(PNXi*PNXj+PNYi*PNYj))*coef;

      a12= (valA*PNXi*PNYj + valB*PNXj*PNYi)*coef;
      a13= (valA*PNXi*PNZj + valB*PNXj*PNZi)*coef;
      a21= (valA*PNYi*PNXj + valB*PNYj*PNXi)*coef;
      a23= (valA*PNYi*PNZj + valB*PNYj*PNZi)*coef;
      a31= (valA*PNZi*PNXj + valB*PNZj*PNXi)*coef;
      a32= (valA*PNZi*PNYj + valB*PNZj*PNYi)*coef;
```

$$\begin{Bmatrix}\sigma_x\\ \sigma_y\\ \sigma_z\\ \tau_{xy}\\ \tau_{yz}\\ \tau_{zx}\end{Bmatrix} = \begin{bmatrix}\mathrm{valX} & \mathrm{valA} & \mathrm{valA} & 0 & 0 & 0\\ \mathrm{valA} & \mathrm{valX} & \mathrm{valA} & 0 & 0 & 0\\ \mathrm{valA} & \mathrm{valA} & \mathrm{valX} & 0 & 0 & 0\\ 0 & 0 & 0 & \mathrm{valB} & 0 & 0\\ 0 & 0 & 0 & 0 & \mathrm{valB} & 0\\ 0 & 0 & 0 & 0 & 0 & \mathrm{valB}\end{bmatrix} \begin{Bmatrix}\varepsilon_x\\ \varepsilon_y\\ \varepsilon_z\\ \gamma_{xy}\\ \gamma_{yz}\\ \gamma_{zx}\end{Bmatrix}$$

要素マトリクス：i-j成分, X方向（1/3）

$$\begin{bmatrix} a_{ij\,11} & a_{ij\,12} & a_{ij\,13} \\ a_{ij\,21} & a_{ij\,22} & a_{ij\,23} \\ a_{ij\,31} & a_{ij\,32} & a_{ij\,33} \end{bmatrix} \quad (i, j = 1 \ldots 8)$$



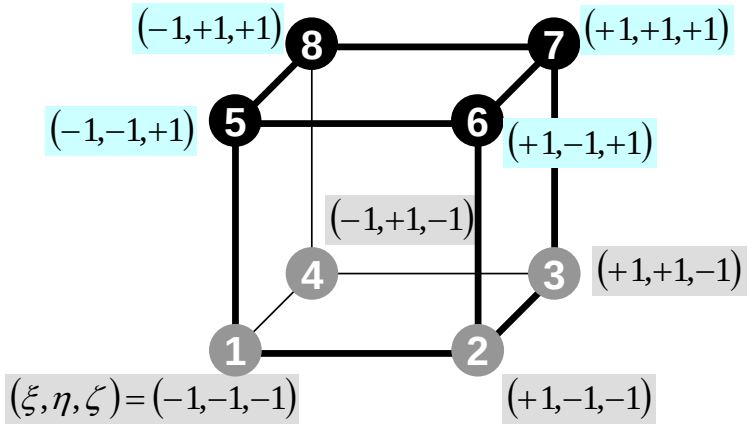
$$a_{ij\,11} = -\int_V \left\{ \mathrm{valX} \cdot N_{i,x} \cdot N_{j,x} + \mathrm{valB} \cdot \left(N_{i,y} \cdot N_{j,y} + N_{i,z} \cdot N_{j,z} \right) \right\} dV$$

$$a_{ij\,12} = -\int_V \left\{ \mathrm{valA} \cdot N_{i,x} \cdot N_{j,y} + \mathrm{valB} \cdot N_{i,y} \cdot N_{j,x} \right\} dV$$

$$a_{ij\,13} = -\int_V \left\{ \mathrm{valA} \cdot N_{i,x} \cdot N_{j,z} + \mathrm{valB} \cdot N_{i,z} \cdot N_{j,x} \right\} dV$$

要素マトリクス：i-j成分, Y方向（2/3）

$$\begin{bmatrix} a_{ij\,11} & a_{ij\,12} & a_{ij\,13} \\ a_{ij\,21} & a_{ij\,22} & a_{ij\,23} \\ a_{ij\,31} & a_{ij\,32} & a_{ij\,33} \end{bmatrix} \quad (i, j = 1 \ldots 8)$$



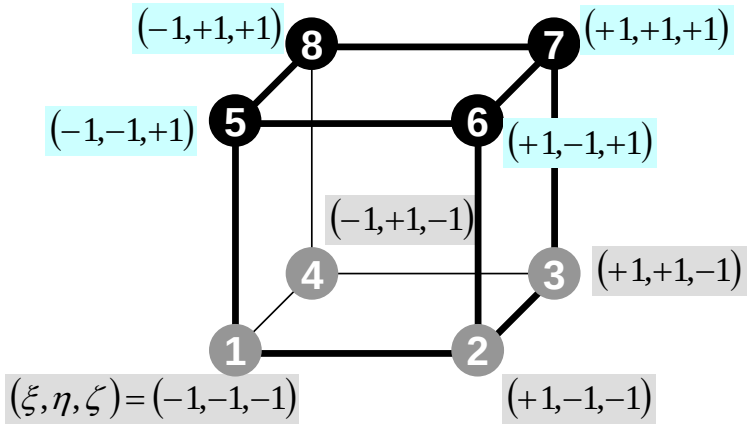
$$a_{ij\,21} = -\int_V \left\{ \mathrm{valA} \cdot N_{i,y} \cdot N_{j,x} + \mathrm{valB} \cdot N_{i,x} \cdot N_{j,y} \right\} dV$$

$$a_{ij\,22} = -\int_V \left\{ \mathrm{valX} \cdot N_{i,y} \cdot N_{j,y} + \mathrm{valB} \cdot \left(N_{i,z} \cdot N_{j,z} + N_{i,x} \cdot N_{j,x} \right) \right\} dV$$

$$a_{ij\,23} = -\int_V \left\{ \mathrm{valA} \cdot N_{i,y} \cdot N_{j,z} + \mathrm{valB} \cdot N_{i,z} \cdot N_{j,y} \right\} dV$$

要素マトリクス：i-j成分, Z方向（3/3）

$$\begin{bmatrix} a_{ij\,11} & a_{ij\,12} & a_{ij\,13} \\ a_{ij\,21} & a_{ij\,22} & a_{ij\,23} \\ a_{ij\,31} & a_{ij\,32} & a_{ij\,33} \end{bmatrix} \quad (i, j = 1 \ldots 8)$$



$$a_{ij\,31} = -\int_V \left\{ \mathrm{valA} \cdot N_{i,z} \cdot N_{j,x} + \mathrm{valB} \cdot N_{i,x} \cdot N_{j,z} \right\} dV$$

$$a_{ij\,32} = -\int_V \left\{ \mathrm{valA} \cdot N_{i,z} \cdot N_{j,y} + \mathrm{valB} \cdot N_{i,y} \cdot N_{j,z} \right\} dV$$

$$a_{ij\,33} = -\int_V \left\{ \mathrm{valD} \cdot N_{i,z} \cdot N_{j,z} + \mathrm{valB} \cdot \left(N_{i,x} \cdot N_{j,x} + N_{i,y} \cdot N_{j,y} \right) \right\} dV$$

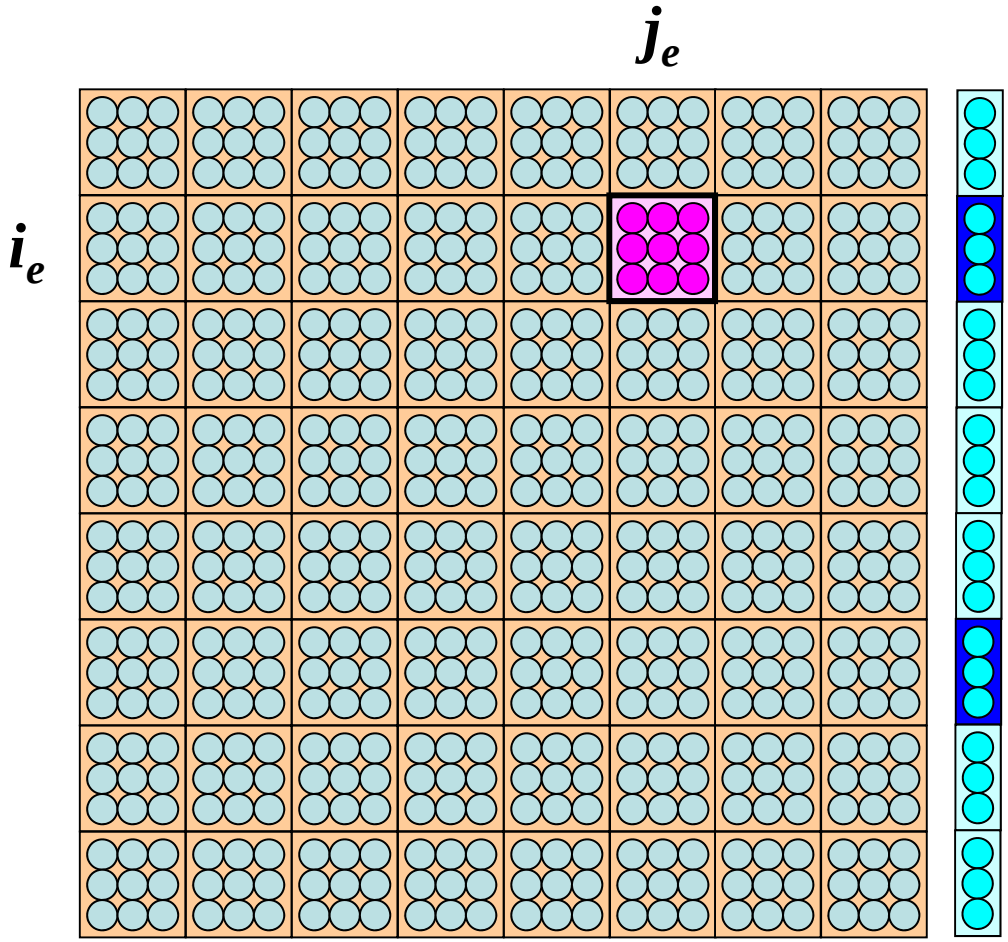
係数行列：MAT_ASS_MAIN（6/7）

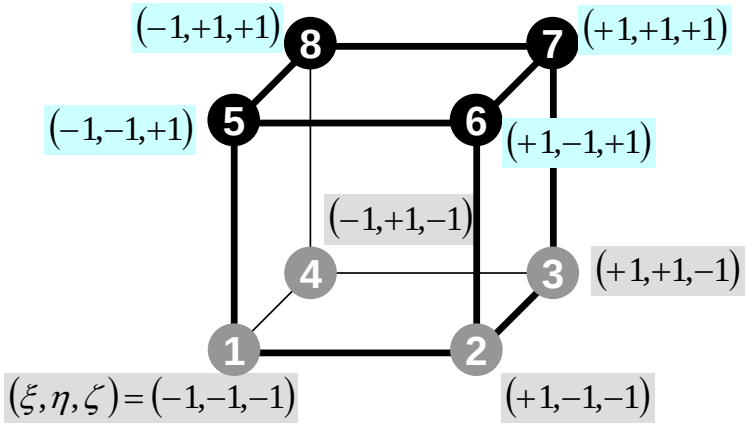
```
for(kpn=0;kpn<2;kpn++){
  for(jpn=0;jpn<2;jpn++){
    for(ipn=0;ipn<2;ipn++){
      coef= -fabs(DETJ[ipn][jpn][kpn])*WEI[ipn]*WEI[jpn]*WEI[kpn];

      VOL+=coef;
      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];

      a11= (valX*PNXi*PNXj+valB*(PNYi*PNYj+PNZi*PNZj))*coef;
      a22= (valX*PNYi*PNYj+valB*(PNZi*PNZj+PNXi*PNXj))*coef;
      a33= (valX*PNZi*PNZj+valB*(PNXi*PNXj+PNYi*PNYj))*coef;

      a12= (valA*PNXi*PNYj + valB*PNXj*PNYi)*coef;
      a13= (valA*PNXi*PNZj + valB*PNXj*PNZi)*coef;
      a21= (valA*PNYi*PNXj + valB*PNYj*PNXi)*coef;
      a23= (valA*PNYi*PNZj + valB*PNYj*PNZi)*coef;
      a31= (valA*PNZi*PNXj + valB*PNZj*PNXi)*coef;
      a32= (valA*PNZi*PNYj + valB*PNZj*PNYi)*coef;

      if (jp > ip) {
        AU[9*kk  ]+=a11;
        AU[9*kk+1]+=a12;
        AU[9*kk+2]+=a13;
        AU[9*kk+3]+=a21;
        AU[9*kk+4]+=a22;
        AU[9*kk+5]+=a23;
        AU[9*kk+6]+=a31;
        AU[9*kk+7]+=a32;
        AU[9*kk+8]+=a33;
        }
```

kk：itemL，itemUにおけるアドレス
　0から始まる

要素マトリクス：24×24行列

各節点上の（u,v,w）成分が物理的にも強くカップルしているので3自由度をまとめて扱う：8×8行列





$$\begin{bmatrix} a_{i_e j_e 11} & a_{i_e j_e 12} & a_{i_e j_e 13} \\ a_{i_e j_e 21} & a_{i_e j_e 22} & a_{i_e j_e 23} \\ a_{i_e j_e 31} & a_{i_e j_e 32} & a_{i_e j_e 33} \end{bmatrix} (i_e, j_e = 1 \ldots 8)$$

係数行列：MAT_ASS_MAIN（7/7）

```
                                    if (jp < ip) {
                                        AL[9*kk  ]+=a11;
                                        AL[9*kk+1]+=a12;
                                        AL[9*kk+2]+=a13;
                                        AL[9*kk+3]+=a21;
                                        AL[9*kk+4]+=a22;
                                        AL[9*kk+5]+=a23;
                                        AL[9*kk+6]+=a31;
                                        AL[9*kk+7]+=a32;
                                        AL[9*kk+8]+=a33;
                                        }

                                    if (jp == ip) {
                                        D[9*ip  ]+=a11;
                                        D[9*ip+1]+=a12;
                                        D[9*ip+2]+=a13;
                                        D[9*ip+3]+=a21;
                                        D[9*ip+4]+=a22;
                                        D[9*ip+5]+=a23;
                                        D[9*ip+6]+=a31;
                                        D[9*ip+7]+=a32;
                                        D[9*ip+8]+=a33;
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
}
```

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
```

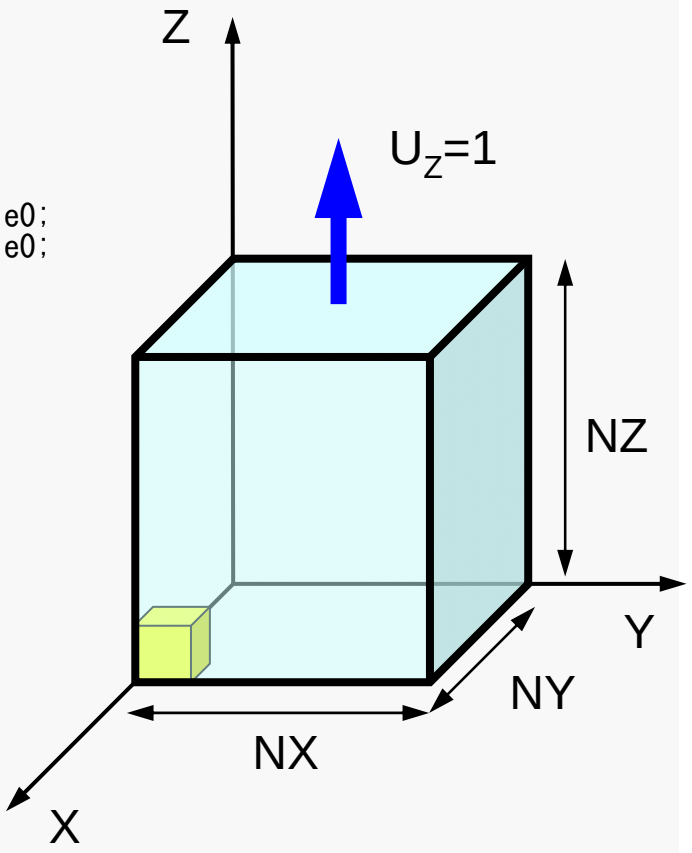
```
do i= 1, NP 節点ループ
  if (IWKX(i,1).eq.1) then  マークされた節点だったら
    対応する右辺ベクトル（B）の成分，対角ブロック（D）の成分の修正（行・列）
    do k= indexL(i-1)+1, indexL(i)  下三角ブロック
      対応する非対角（下三角）ブロック（AL）の成分の修正（行）
    enddo
    do k= indexU(i-1)+1, indexU(i)  上三角ブロック
      対応する非対角（上三角）ブロック（AU）の成分の修正（行）
    enddo
  endif
enddo

do i= 1, NP 節点ループ
  do k= indexL(i-1)+1, indexL(i)      下三角ブロック
    if (IWKX(itemL(k),1).eq.1) then  対応する非対角ブロックの節点がマークされていたら
      対応する右辺ベクトル，非対角（下三角）ブロック（AL）の成分の修正（列）
    endif
  enddo
  do k= indexU(i-1)+1, indexU(i)      上三角ブロック
    if (IWKX(itemU(k),1).eq.1) then  対応する非対角ブロックの節点がマークされていたら
      対応する右辺ベクトル，非対角（上三角）ブロック（AU）の成分の修正（列）
    endif
  enddo
enddo
```

境界条件：MAT ASS BC（1/9）

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE *fp_log;
void MAT_ASS_BC()
{
            int i,j,k,in,ib,ib0,icel;
            int in1,in2,in3,in4,in5,in6,in7,in8;
            int iq1,iq2,iq3,iq4,iq5,iq6,iq7,iq8;
            int iS,iE;
            double STRESS,VAL;

            IWKX=(KINT**) allocate_matrix(sizeof(KINT),NP,2);
            for(i=0;i<N;i++) for(j=0;j<2;j++) IWKX[i][j]=0;
```

境界条件：MAT ASS BC（2/9）

```
/**
        Z=Zmax
**/
        for(in=0;in<NP;in++) IWKX[in][0]=0;

        ib0=-1;

        for( ib0=0;ib0<NODGRPtot;ib0++){
                if( strcmp(NODGRP_NAME[ib0].name,"Zmax") == 0 ) break;
        }

        for( ib=NODGRP_INDEX[ib0];ib<NODGRP_INDEX[ib0+1];ib++){
                in=NODGRP_ITEM[ib];
                IWKX[in-1][0]=1;
        }

        for(in=0;in<NP;in++){
                if( IWKX[in][0] == 1 ){
                        B[3*in  ]= B[3*in  ] - D[9*in+2]*1.e0;
                        B[3*in+1]= B[3*in+1] - D[9*in+5]*1.e0;
                        D[9*in+2]= 0.e0;
                        D[9*in+5]= 0.e0;
                        D[9*in+6]= 0.e0;
                        D[9*in+7]= 0.e0;
                        D[9*in+8]= 1.e0;
                        B[3*in+2]= 1.e0;
                        iS= indexL[in];
                        iE= indexL[in+1];
                        for(k=iS;k<iE;k++){
                                AL[9*k+6]= 0.e0;
                                AL[9*k+7]= 0.e0;
                                AL[9*k+8]= 0.e0;
                        }
                        iS= indexU[in];
                        iE= indexU[in+1];
                        for(k=iS;k<iE;k++){
                                AU[9*k+6]= 0.e0;
                                AU[9*k+7]= 0.e0;
                                AU[9*k+8]= 0.e0;
                        }
                } }
```



境界条件：MAT ASS BC（2/9）

```
/**
        Z=Zmax
**/
        for(in=0;in<NP;in++) IWKX[in][0]=0;

        ib0=-1;

        for( ib0=0;ib0<NODGRPtot;ib0++){
                if( strcmp(NODGRP_NAME[ib0].name,"Zmax") == 0 ) break;
        }

        for( ib=NODGRP_INDEX[ib0];ib<NODGRP_INDEX[ib0+1];ib++){
                in=NODGRP_ITEM[ib];
                IWKX[in-1][0]=1;
        }

        for(in=0;in<NP;in++){
                if( IWKX[in][0] == 1 ){
                        B[3*in  ]= B[3*in  ] - D[9*in+2]*
                        B[3*in+1]= B[3*in+1] - D[9*in+5]*
                        D[9*in+2]= 0.e0;
                        D[9*in+5]= 0.e0;
                        D[9*in+6]= 0.e0;
                        D[9*in+7]= 0.e0;
                        D[9*in+8]= 1.e0;
                        B[3*in+2]= 1.e0;
                        iS= indexL[in];
                        iE= indexL[in+1];
                        for(k=iS;k<iE;k++){
                                AL[9*k+6]= 0.e0;
                                AL[9*k+7]= 0.e0;
                                AL[9*k+8]= 0.e0;
                        }
                        iS= indexU[in];
                        iE= indexU[in+1];
                        for(k=iS;k<iE;k++){
                                AU[9*k+6]= 0.e0;
                                AU[9*k+7]= 0.e0;
                                AU[9*k+8]= 0.e0;
                        }
                }  }
```

節点グループ名が「Zmax」である節点inにおいて：

IWKX[in-1][0]= 1

とする

境界条件：MAT ASS BC（2/9）

```
/**
        Z=Zmax
**/
        for(in=0;in<NP;in++) IWKX[in][0]=0;

        ib0=-1;

        for( ib0=0;ib0<NODGRPtot;ib0++){
                if( strcmp(NODGRP_NAME[ib0].name,"Zmax") == 0 ) break;
        }

        for( ib=NODGRP_INDEX[ib0];ib<NODGRP_INDEX[ib0+1];ib++){
                in=NODGRP_ITEM[ib];
                IWKX[in-1][0]=1;
        }

        for(in=0;in<NP;in++){
                if( IWKX[in][0] == 1 ){
                        B[3*in  ]= B[3*in  ] - D[9*in+2]*1.e0;
                        B[3*in+1]= B[3*in+1] - D[9*in+5]*1.e0;
                        D[9*in+2]= 0.e0;
                        D[9*in+5]= 0.e0;
                        D[9*in+6]= 0.e0;
                        D[9*in+7]= 0.e0;
                        D[9*in+8]= 1.e0;
                        B[3*in+2]= 1.e0;
                        iS= indexL[in];
                        iE= indexL[in+1];
                        for(k=iS;k<iE;k++){
                                AL[9*k+6]= 0.e0;
                                AL[9*k+7]= 0.e0;
                                AL[9*k+8]= 0.e0;
                        }
                        iS= indexU[in];
                        iE= indexU[in+1];
                        for(k=iS;k<iE;k++){
                                AU[9*k+6]= 0.e0;
                                AU[9*k+7]= 0.e0;
                                AU[9*k+8]= 0.e0;
                        }
                } }
```

節点グループ名が「Zmax」である節点では，Z方向変位成分=1にすなわち：

B[3*in+2]= 1.0 (in= 0~N-1)

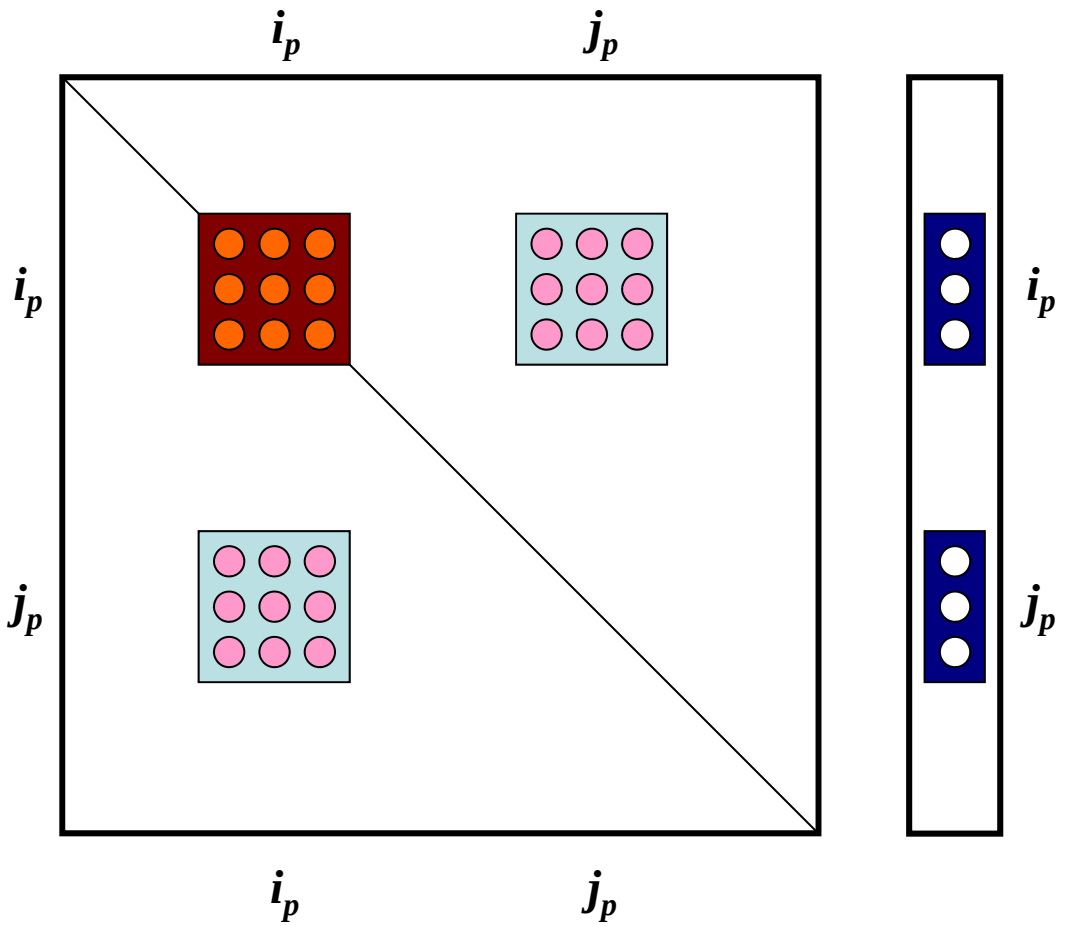
FORTRANでは

B[3*in-2]= 1.0 (in= 1, N)

境界条件：MAT ASS BC（3/9）

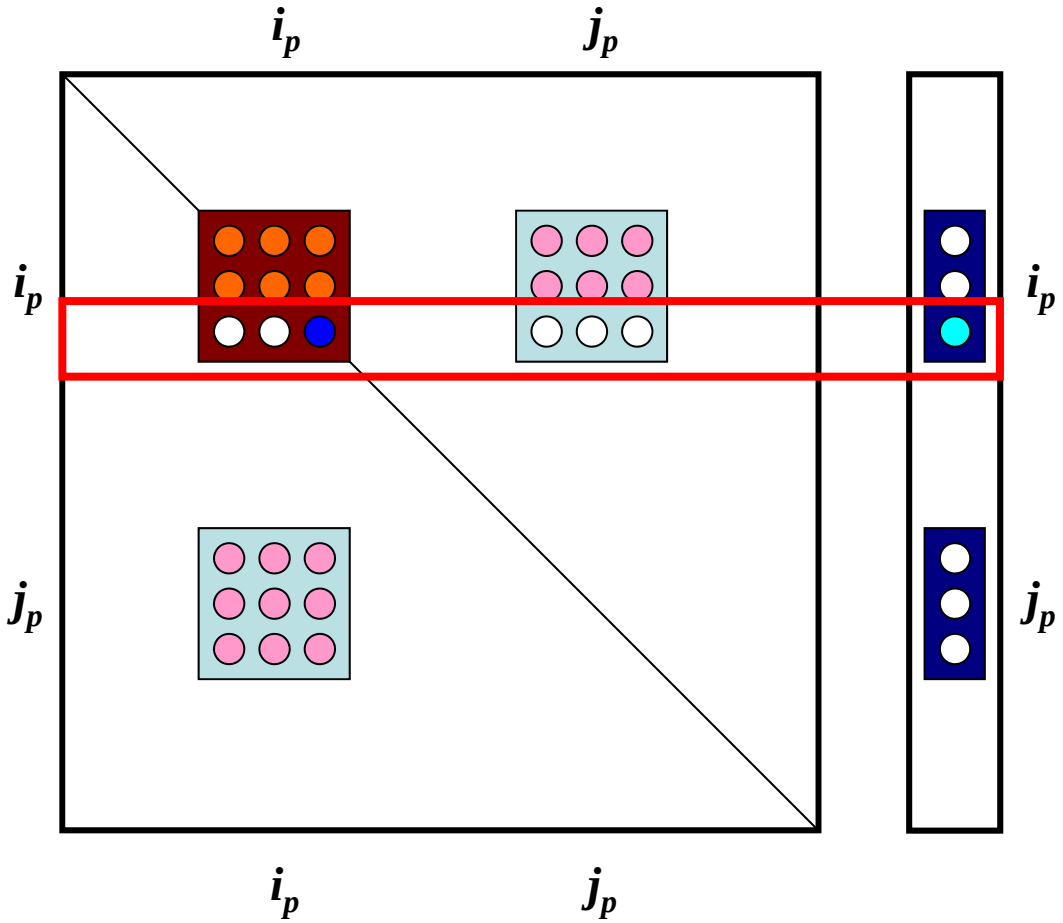
```
for(in=0;in<NP;in++){
                iS= indexL[in];
                iE= indexL[in+1];
                for(k=iS;k<=iE;k++){
                        if (IWKX[itemL[k]][0] == 1 ) {
                                B[3*in  ]= B[3*in  ] - AL[9*k+2]*1.e0;
                                B[3*in+1]= B[3*in+1] - AL[9*k+5]*1.e0;
                                B[3*in+2]= B[3*in+2] - AL[9*k+8]*1.e0;
                                AL[9*k+2]= 0.e0;
                                AL[9*k+5]= 0.e0;
                                AL[9*k+8]= 0.e0;
                        }
                }
                iS= indexU[in];
                iE= indexU[in+1];

                for(k=iS;k<=iE;k++){
                        if (IWKX[itemU[k]][0] == 1 ) {
                                B[3*in  ]= B[3*in  ] - AU[9*k+2]*1.e0;
                                B[3*in+1]= B[3*in+1] - AU[9*k+5]*1.e0;
                                B[3*in+2]= B[3*in+2] - AU[9*k+8]*1.e0;
                                AU[9*k+2]= 0.e0;
                                AU[9*k+5]= 0.e0;
                                AU[9*k+8]= 0.e0;
                        }
                }
        }
```

全体マトリクス



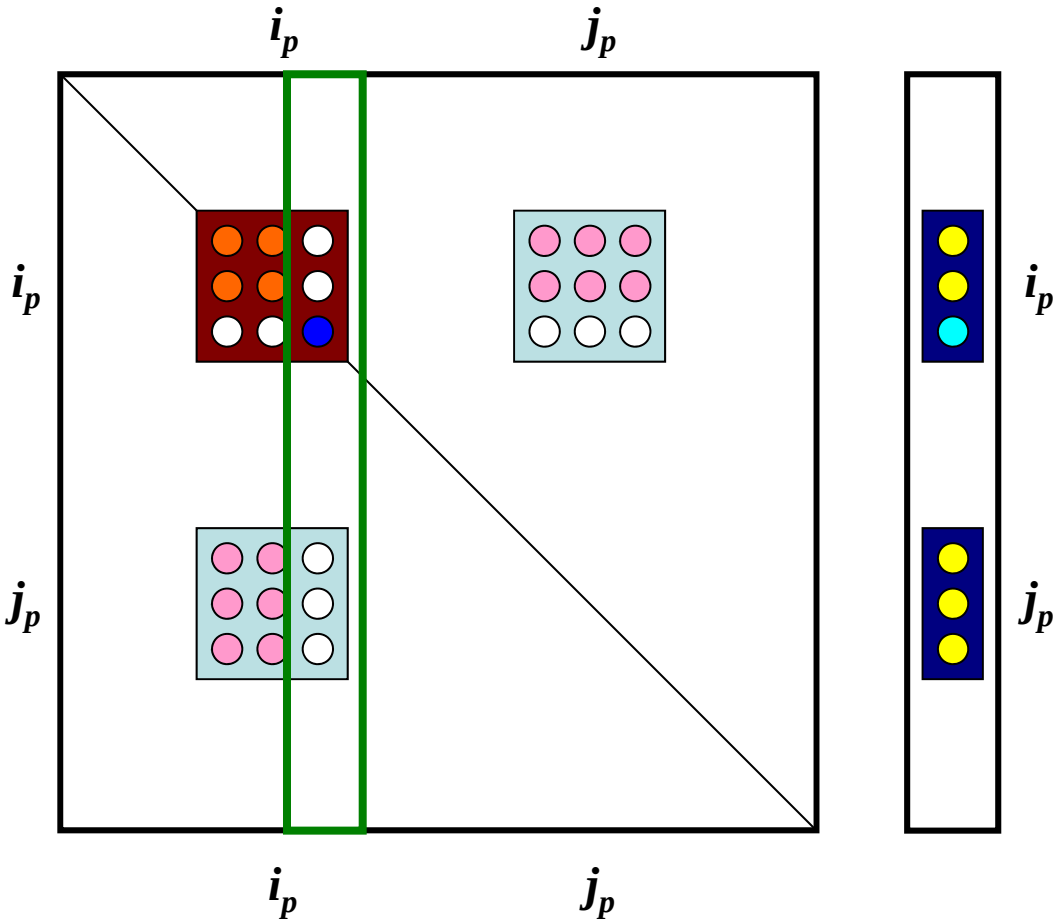
一次元と同じことをやればよい

自分の「行」：対角成分=1，それ以外=0



一次元と同じことをやればよい

自分の「列」：右辺へ移項，非対角成分=0



境界条件：MAT ASS BC（2/9）

```
/**
        Z=Zmax
**/
        for(in=0;in<NP;in++) IWKX[in][0]=0;

        ib0=-1;

        for( ib0=0;ib0<NODGRPtot;ib0++){
                if( strcmp(NODGRP_NAME[ib0].name,"Zmax") == 0 ) break;
        }

        for( ib=NODGRP_INDEX[ib0];ib<NODGRP_INDEX[ib0+1];ib++){
                in=NODGRP_ITEM[ib];
                IWKX[in-1][0]=1;
        }

        for(in=0;in<NP;in++){
                if( IWKX[in][0] == 1 ){
                        B[3*in  ]= B[3*in  ] - D[9*in+2]*
                        B[3*in+1]= B[3*in+1] - D[9*in+5]*
                        D[9*in+2]= 0.e0;
                        D[9*in+5]= 0.e0;
                        D[9*in+6]= 0.e0;
                        D[9*in+7]= 0.e0;
                        D[9*in+8]= 1.e0;
                        B[3*in+2]= 1.e0;
                        iS= indexL[in];
                        iE= indexL[in+1];
                        for(k=iS;k<iE;k++){
                                AL[9*k+6]= 0.e0;
                                AL[9*k+7]= 0.e0;
                                AL[9*k+8]= 0.e0;
                        }
                        iS= indexU[in];
                        iE= indexU[in+1];
                        for(k=iS;k<iE;k++){
                                AU[9*k+6]= 0.e0;
                                AU[9*k+7]= 0.e0;
                                AU[9*k+8]= 0.e0;
                        }
                }  }
```

節点グループ名が「Zmax」である節点inにおいて：

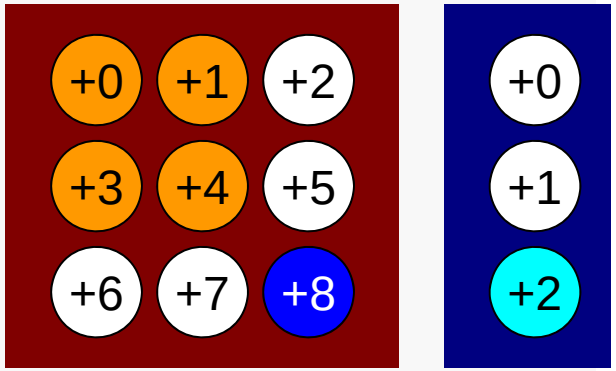
IWKX[in-1][0]= 1

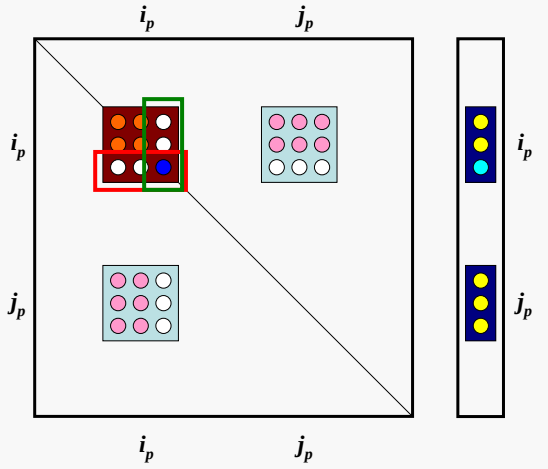
とする

境界条件：MAT_ASS_BC（2/9）

```
/**
        Z=Zmax
**/
        for(in=0;in<NP;in++) IWKX[in][0]=0;

        ib0=-1;

        for( ib0=0;ib0<NODGRPtot;ib0++){
                if( strcmp(NODGRP_NAME[ib0].name,"Zmax") == 0 ) break;
        }

        for( ib=NODGRP_INDEX[ib0];ib<NODGRP_INDEX[ib0+1];ib++){
                in=NODGRP_ITEM[ib];
                IWKX[in-1][0]=1;
        }

        for(in=0;in<NP;in++){
                if( IWKX[in][0] == 1 ){
                        B[3*in  ]= B[3*in  ] - D[9*in+2]*1.e0;
                        B[3*in+1]= B[3*in+1] - D[9*in+5]*1.e0;
                        D[9*in+2]= 0.e0;
                        D[9*in+5]= 0.e0;
                        D[9*in+6]= 0.e0;
                        D[9*in+7]= 0.e0;
                        D[9*in+8]= 1.e0;
                        B[3*in+2]= 1.e0;
                        iS= indexL[in];
                        iE= indexL[in+1];
                        for(k=iS;k<iE;k++){
                                AL[9*k+6]= 0.e0;
                                AL[9*k+7]= 0.e0;
                                AL[9*k+8]= 0.e0;
                        }
                        iS= indexU[in];
                        iE= indexU[in+1];
                        for(k=iS;k<iE;k++){
                                AU[9*k+6]= 0.e0;
                                AU[9*k+7]= 0.e0;
                                AU[9*k+8]= 0.e0;
                        }
                }  }
```

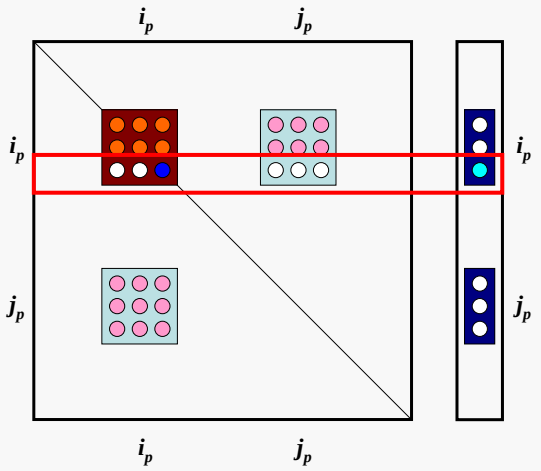
B[3*in], B[3*in+1]を変更してからD[9*in+6], D[9*in+7]をゼロクリア

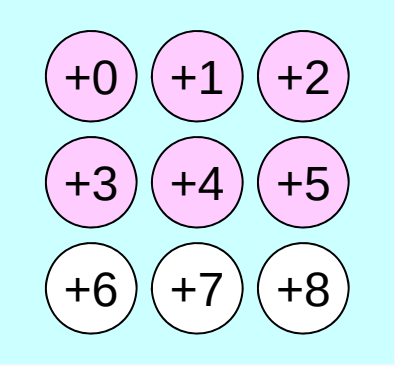




境界条件：MAT_ASS_BC（2/9）

```
/**
        Z=Zmax
**/
        for(in=0;in<NP;in++) IWKX[in][0]=0;

        ib0=-1;

        for( ib0=0;ib0<NODGRPtot;ib0++){
                if( strcmp(NODGRP_NAME[ib0].name,"Zmax") == 0 ) break;
        }

        for( ib=NODGRP_INDEX[ib0];ib<NODGRP_INDEX[ib0+1];ib++){
                in=NODGRP_ITEM[ib];
                IWKX[in-1][0]=1;
        }

        for(in=0;in<NP;in++){
                if( IWKX[in][0] == 1 ){
                        B[3*in  ]= B[3*in  ] - D[9*in+2]*1.e0;
                        B[3*in+1]= B[3*in+1] - D[9*in+5]*1.e0;
                        D[9*in+2]= 0.e0;
                        D[9*in+5]= 0.e0;
                        D[9*in+6]= 0.e0;
                        D[9*in+7]= 0.e0;
                        D[9*in+8]= 1.e0;
                        B[3*in+2]= 1.e0;
                        iS= indexL[in];
                        iE= indexL[in+1];
                        for(k=iS;k<iE;k++){
                                AL[9*k+6]= 0.e0;
                                AL[9*k+7]= 0.e0;
                                AL[9*k+8]= 0.e0;
                        }
                        iS= indexU[in];
                        iE= indexU[in+1];
                        for(k=iS;k<iE;k++){
                                AU[9*k+6]= 0.e0;
                                AU[9*k+7]= 0.e0;
                                AU[9*k+8]= 0.e0;
                        }
                } }
```

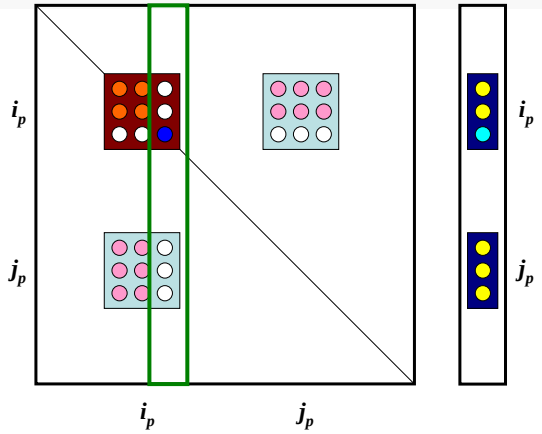


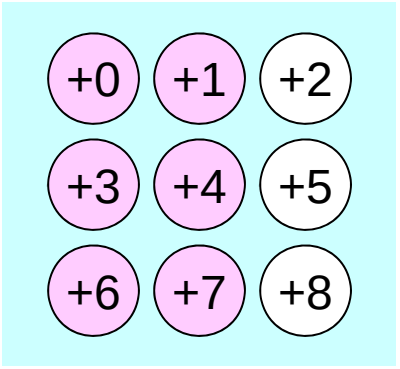


境界条件：MAT ASS BC（3/9）

```
for(in=0;in<NP;in++){
                    iS= indexL[in];
                    iE= indexL[in+1];
                    for(k=iS;k<iE;k++){
                                if (IWKX[itemL[k]][0] == 1 ) {
                                            B[3*in  ]= B[3*in  ] - AL[9*k+2]*1.e0;
                                            B[3*in+1]= B[3*in+1] - AL[9*k+5]*1.e0;
                                            B[3*in+2]= B[3*in+2] - AL[9*k+8]*1.e0;
                                            AL[9*k+2]= 0.e0;
                                            AL[9*k+5]= 0.e0;
                                            AL[9*k+8]= 0.e0;
                                }
                    }
                    iS= indexU[in];
                    iE= indexU[in+1];

                    for(k=iS;k<iE;k++){
                                if (IWKX[itemU[k]][0] == 1 ) {
                                            B[3*in  ]= B[3*in  ] - AU[9*k+2]*1.e0;
                                            B[3*in+1]= B[3*in+1] - AU[9*k+5]*1.e0;
                                            B[3*in+2]= B[3*in+2] - AU[9*k+8]*1.e0;
                                            AU[9*k+2]= 0.e0;
                                            AU[9*k+5]= 0.e0;
                                            AU[9*k+8]= 0.e0;
                                }
                    }
        }
```

右辺を変更してから
ゼロクリア

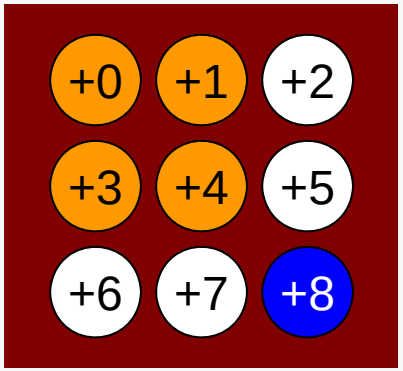


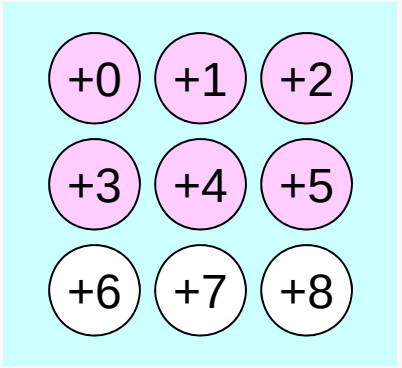


境界条件：MAT ASS BC（4/9）

w=0@Zmin

```
/**
        Z=Zmin
**/
        for(in=0;in<NP;in++) IWKX[in][0]=0;

        ib0=-1;

        for( ib0=0;ib0<NODGRPtot;ib0++){
                if( strcmp(NODGRP_NAME[ib0].name,"Zmin") == 0 ) break;
        }

        for( ib=NODGRP_INDEX[ib0];ib<NODGRP_INDEX[ib0+1];ib++){
                in=NODGRP_ITEM[ib];
                IWKX[in-1][0]=1;
        }

        for(in=0;in<NP;in++){
                if( IWKX[in][0] == 1 ){
                        D[9*in+2]= 0.e0;
                        D[9*in+5]= 0.e0;
                        D[9*in+6]= 0.e0;
                        D[9*in+7]= 0.e0;
                        D[9*in+8]= 1.e0;
                        B[3*in+2]= 0.e0;
                        iS= indexL[in];
                        iE= indexL[in+1];
                        for(k=iS;k<iE;k++){
                                AL[9*k+6]= 0.e0;
                                AL[9*k+7]= 0.e0;
                                AL[9*k+8]= 0.e0;
                        }
                        iS= indexU[in];
                        iE= indexU[in+1];
                        for(k=iS;k<iE;k++){
                                AU[9*k+6]= 0.e0;
                                AU[9*k+7]= 0.e0;
                                AU[9*k+8]= 0.e0;
                        }
                }
        }
```

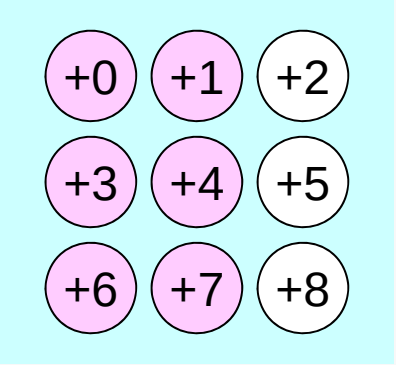




境界条件：MAT ASS BC（5/9）

```
for(in=0;in<NP;in++){
                iS= indexL[in];
                iE= indexL[in+1];
                for(k=iS;k<iE;k++){
                        if (IWKX[itemL[k]][0] == 1 ) {
                                AL[9*k+2]= 0.e0;
                                AL[9*k+5]= 0.e0;
                                AL[9*k+8]= 0.e0;
                        }
                }
                iS= indexU[in];
                iE= indexU[in+1];

                for(k=iS;k<iE;k++){
                        if (IWKX[itemU[k]][0] == 1 ) {
                                AU[9*k+2]= 0.e0;
                                AU[9*k+5]= 0.e0;
                                AU[9*k+8]= 0.e0;
                        }
                }
}
```

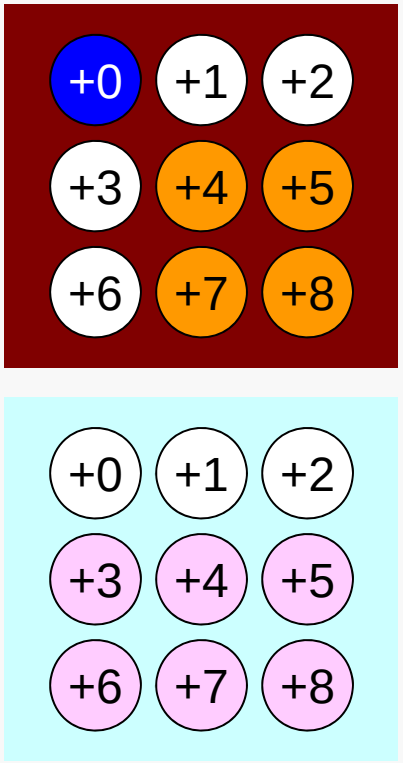
w=0@Zmin



境界条件：MAT ASS BC（6/9）

u=0@Xmin

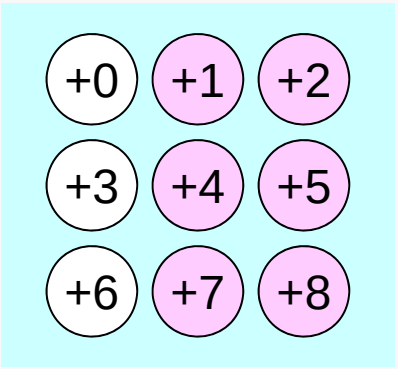
```
/**
        X=Xmin
**/
        for(in=0;in<NP;in++) IWKX[in][0]=0;

        ib0=-1;

        for( ib0=0;ib0<NODGRPtot;ib0++){
                if( strcmp(NODGRP_NAME[ib0].name,"Xmin") == 0 ) break;
        }

        for( ib=NODGRP_INDEX[ib0];ib<NODGRP_INDEX[ib0+1];ib++){
                in=NODGRP_ITEM[ib];
                IWKX[in-1][0]=1;
        }

        for(in=0;in<NP;in++){
                if( IWKX[in][0] == 1 ){
                        D[9*in  ]= 1.e0;
                        D[9*in+1]= 0.e0;
                        D[9*in+2]= 0.e0;
                        D[9*in+3]= 0.e0;
                        D[9*in+6]= 0.e0;
                        B[3*in  ]= 0.e0;

                        iS= indexL[in];
                        iE= indexL[in+1];
                        for(k=iS;k<iE;k++){
                                AL[9*k  ]= 0.e0;
                                AL[9*k+1]= 0.e0;
                                AL[9*k+2]= 0.e0;
                        }

                        iS= indexU[in];
                        iE= indexU[in+1];
                        for(k=iS;k<iE;k++){
                                AU[9*k  ]= 0.e0;
                                AU[9*k+1]= 0.e0;
                                AU[9*k+2]= 0.e0;
                        }
                } }
```



境界条件：MAT ASS BC（7/9）

```
for(in=0;in<NP;in++){
            iS= indexL[in];
            iE= indexL[in+1];
            for(k=iS;k<iE;k++){
                        if (IWKX[itemL[k]][0] == 1 ) {
                                    AL[9*k  ]= 0.e0;
                                    AL[9*k+3]= 0.e0;
                                    AL[9*k+6]= 0.e0;
                        }
            }

            iS= indexU[in];
            iE= indexU[in+1];
            for(k=iS;k<iE;k++){
                        if (IWKX[itemU[k]][0] == 1 ) {
                                    AU[9*k  ]= 0.e0;
                                    AU[9*k+3]= 0.e0;
                                    AU[9*k+6]= 0.e0;
                        }
            }
}
```

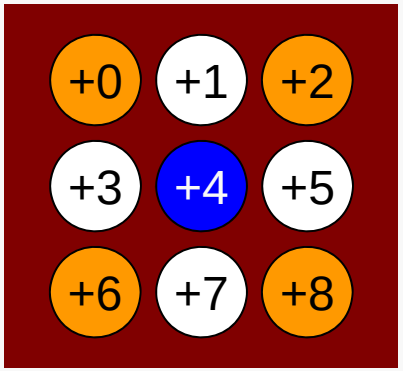
u=0@Xmin

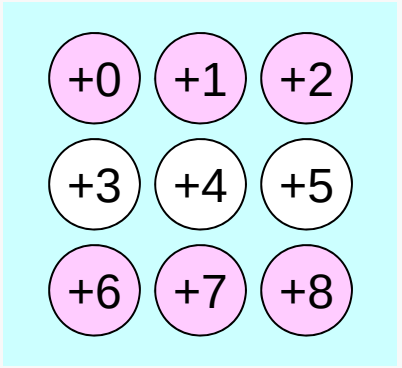


境界条件：MAT ASS BC（8/9）

v=0@Ymin

```
/**
        Y=Ymin
**/
        for(in=0;in<NP;in++) IWKX[in][0]=0;

        ib0=-1;

        for( ib0=0;ib0<NODGRPtot;ib0++){
                if( strcmp(NODGRP_NAME[ib0].name,"Ymin") == 0 ) break;
        }


        for( ib=NODGRP_INDEX[ib0];ib<NODGRP_INDEX[ib0+1];ib++){
                in=NODGRP_ITEM[ib];
                IWKX[in-1][0]=1;
        }

        for(in=0;in<NP;in++){
                if( IWKX[in][0] == 1 ){
                        D[9*in+1]= 0.e0;
                        D[9*in+4]= 1.e0;
                        D[9*in+7]= 0.e0;
                        D[9*in+3]= 0.e0;
                        D[9*in+5]= 0.e0;
                        B[3*in+1]= 0.e0;

                        iS= indexL[in];
                        iE= indexL[in+1];
                        for(k=iS;k<iE;k++){
                                AL[9*k+3]= 0.e0;
                                AL[9*k+4]= 0.e0;
                                AL[9*k+5]= 0.e0;
                        }

                        iS= indexU[in];
                        iE= indexU[in+1];
                        for(k=iS;k<iE;k++){
                                AU[9*k+3]= 0.e0;
                                AU[9*k+4]= 0.e0;
                                AU[9*k+5]= 0.e0;
                        } } }
```

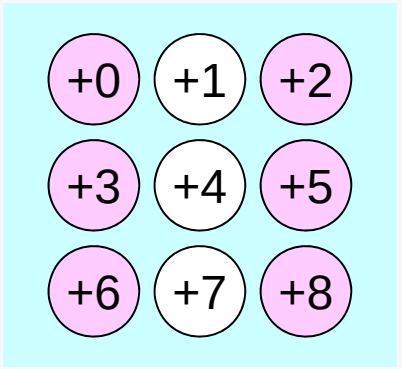




境界条件：MAT_ASS_BC（9/9）

```
        for(in=0;in<NP;in++){
                iS= indexL[in];
                iE= indexL[in+1];
                for(k=iS;k<iE;k++){
                        if (IWKX[itemL[k]][0] == 1 ) {
                                AL[9*k+1]= 0.e0;
                                AL[9*k+4]= 0.e0;
                                AL[9*k+7]= 0.e0;
                        }
                }

                iS= indexU[in];
                iE= indexU[in+1];
                for(k=iS;k<iE;k++){
                        if (IWKX[itemU[k]][0] == 1 ) {
                                AU[9*k+1]= 0.e0;
                                AU[9*k+4]= 0.e0;
                                AU[9*k+7]= 0.e0;
                        }
                }
        }
}
```

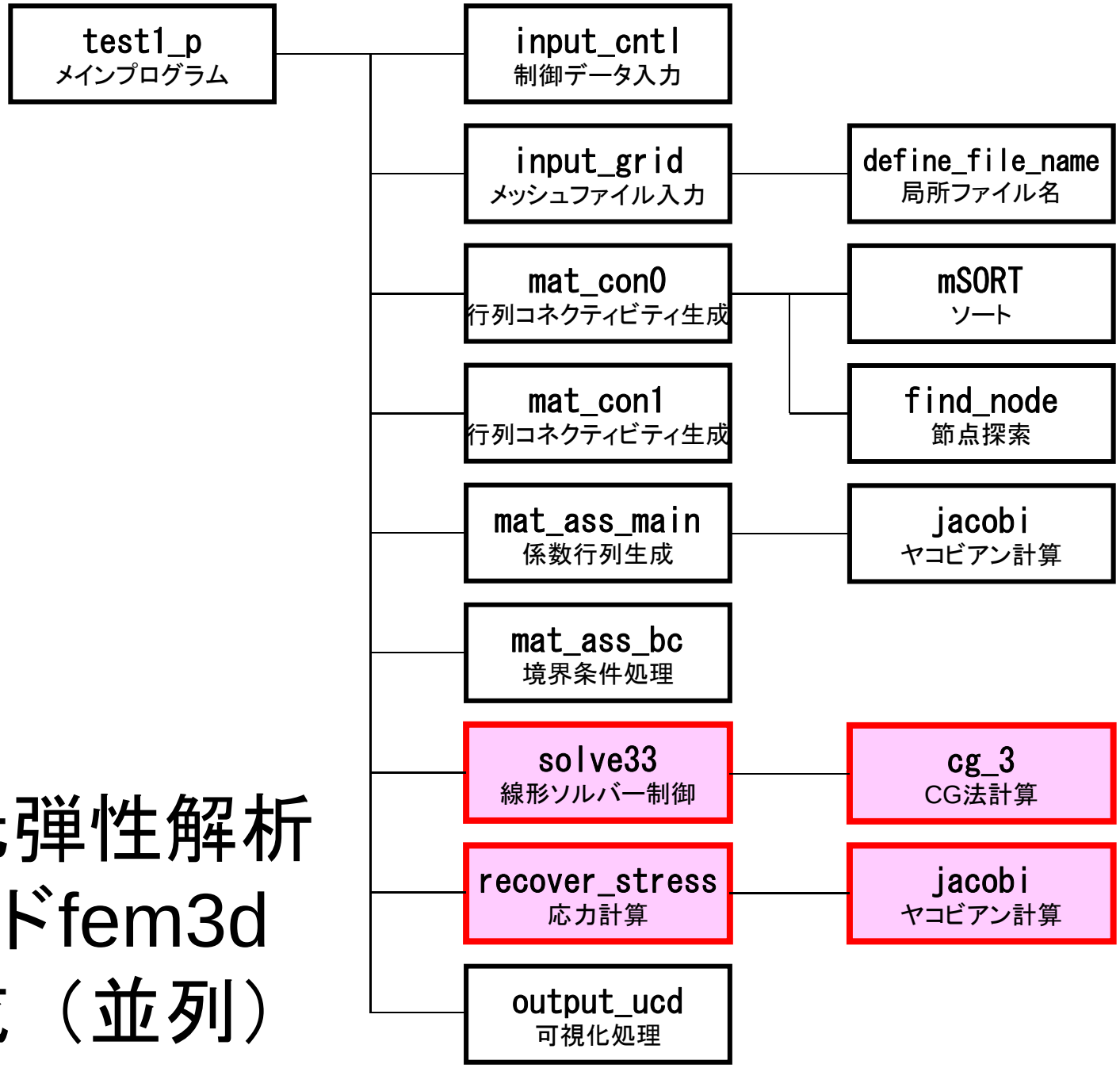
v=0@Ymin



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）**
- **応力計算**

三次元弾性解析コードfem3dの構成（並列）

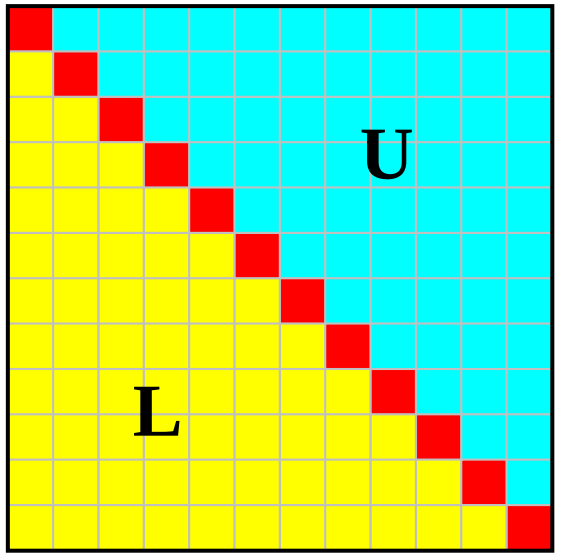


全体処理

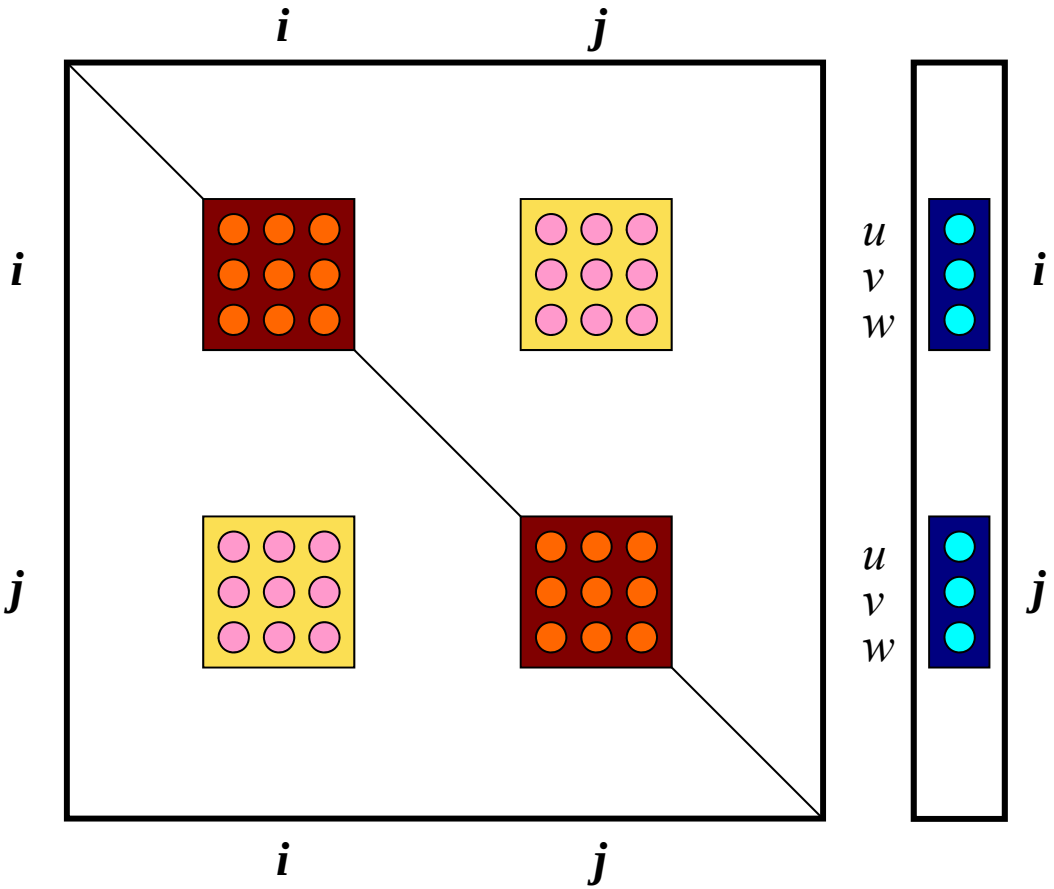
```
#include <stdio.h>
#include <stdlib.h>
FILE* fp_log;
#define GLOBAL_VALUE_DEFINE
#include "pfem_util.h"
//#include "solver33.h"
extern void PFEM_INIT(int,char**);
extern void INPUT_CNTL();
extern void INPUT_GRID();
extern void MAT_CON0();
extern void MAT_CON1();
extern void MAT_ASS_MAIN();
extern void MAT_ASS_BC();
extern void SOLVE33();
extern void RECOVER_STRESS();
extern void OUTPUT_UCD();
extern void PFEM_FINALIZE();
int main(int argc,char* argv[])
{
        double START_TIME,END_TIME;

        PFEM_INIT(argc,argv);

        INPUT_CNTL();
        INPUT_GRID();

        MAT_CON0();
        MAT_CON1();

        MAT_ASS_MAIN();
        MAT_ASS_BC()  ;

        SOLVE33();

        RECOVER_STRESS();
        OUTPUT_UCD()    ;
        PFEM_FINALIZE() ;
```

fem3d：いくつかの特徴

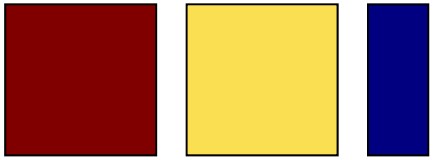
- 非対角成分
  - 上三角，下三角を分けて記憶
    - indexL，itemL，AL
    - indexU，itemU，AU
- ブロックとして記憶
  - ベクトル：1節点3成分
  - 行列：各ブロック9成分
  - 行列の各成分ではなく，節点上の3変数に基づくブロックとして処理する



用語の定義



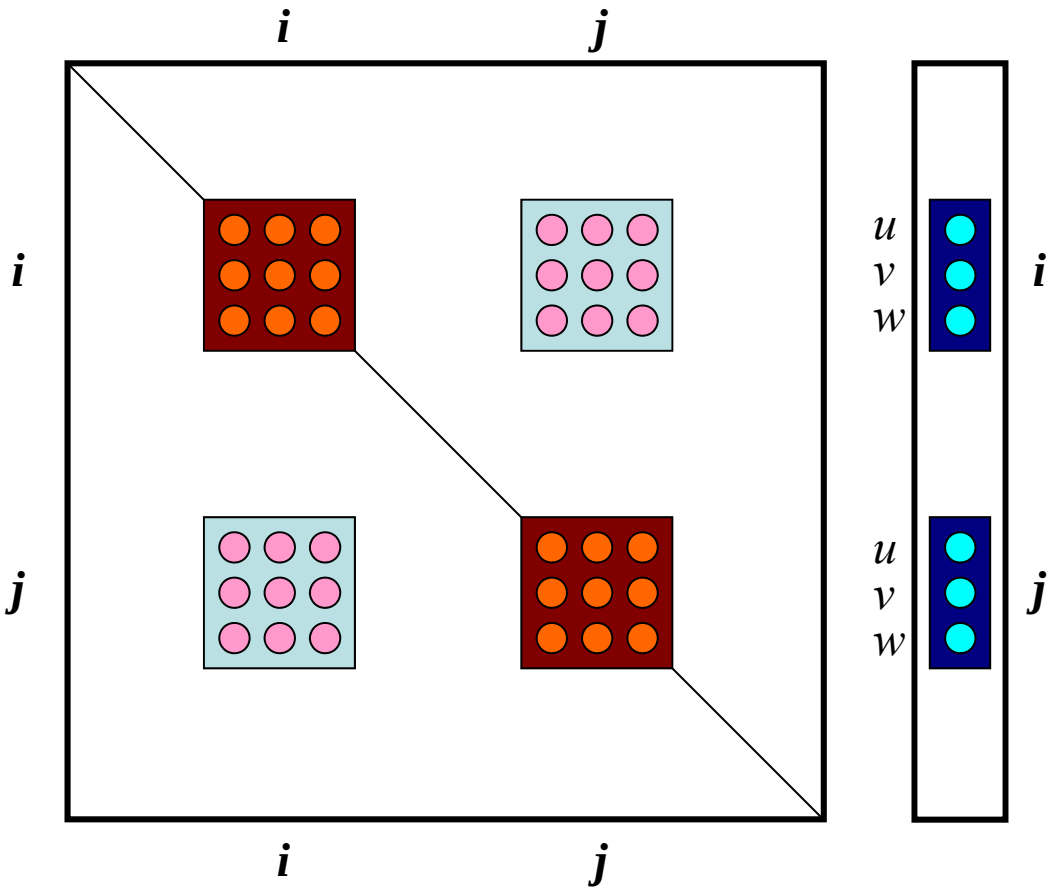
ブロック（節点）:



成分（自由度）:

ブロックとして記憶（1/3）

- 記憶容量が減る
  - index, itemに関する記憶容量を数十分の1に削減できる



ブロックとして記憶（2/3）

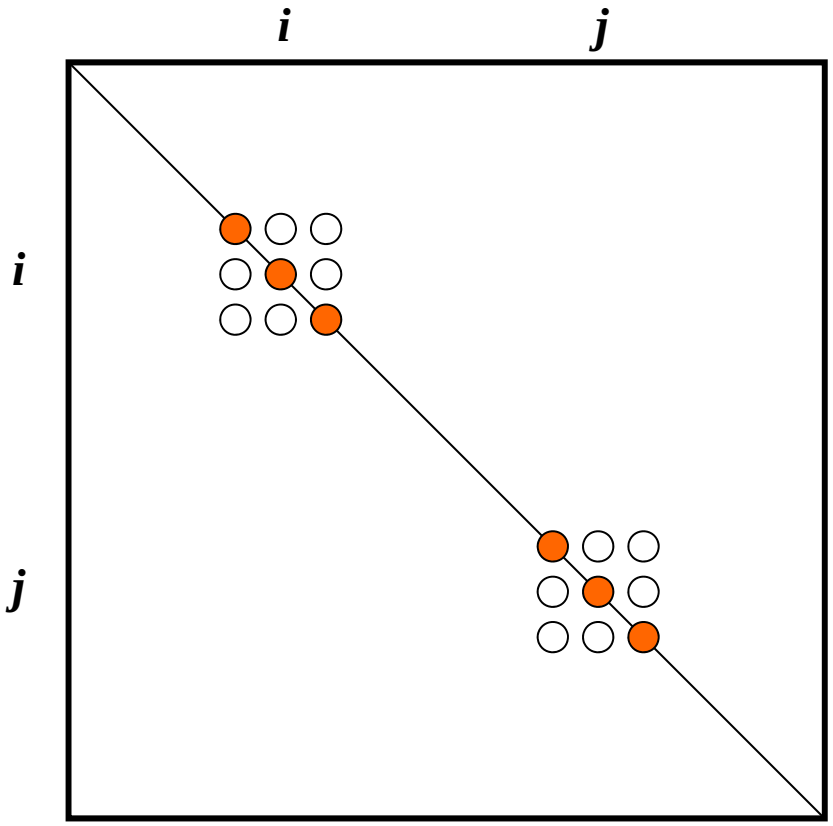
- 計算効率
  - 間接参照（メモリに負担）と計算の比が小さくなる
  - ベクトル，スカラー共に効く：2倍以上の性能

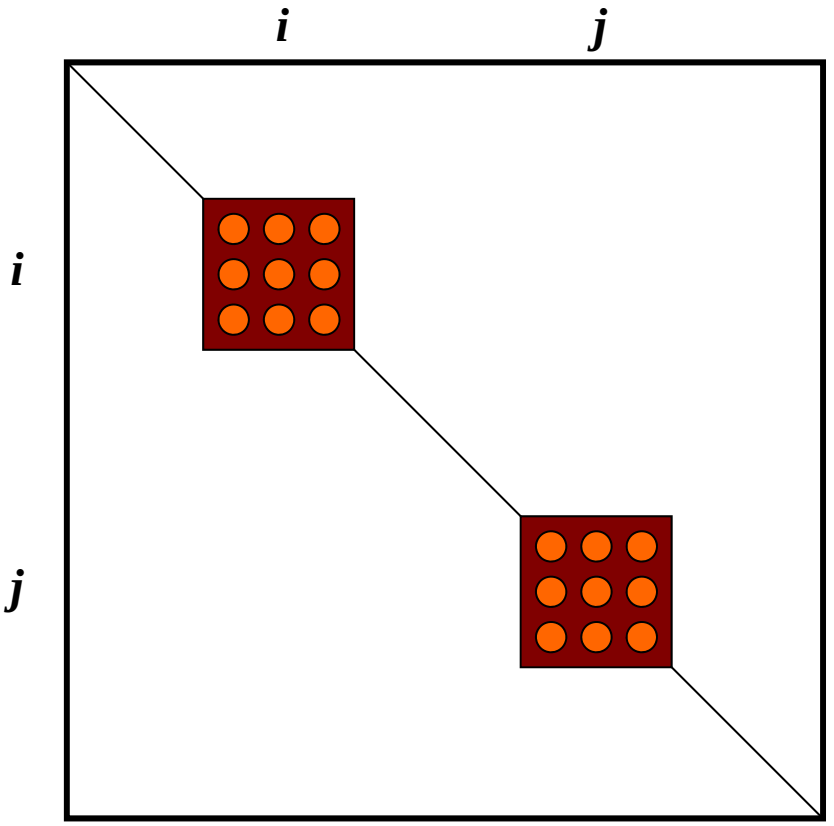
```
do i= 1, 3*N
  Y(i)= D(i)*X(i)
  do k= index(i-1)+1, index(i)
    kk= item(k)
    Y(i)= Y(i) + AMAT(k)*X(k)
  enddo
enddo
```

```
do i= 1, N
  X1= X(3*i-2)
  X2= X(3*i-1)
  X3= X(3*i)
  Y(3*i-2)= D(9*i-8)*X1+D(9*i-7)*X2+D(9*i-6)*X3
  Y(3*i-1)= D(9*i-5)*X1+D(9*i-4)*X2+D(9*i-3)*X3
  Y(3*I  )= D(9*i-2)*X1+D(9*i-1)*X2+D(9*I  )*X3
  do k= index(i-1)+1, index(i)
    kk= item(k)
    X1= X(3*kk-2)     u
    X2= X(3*kk-1)     v
    X3= X(3*kk)       w
    Y(3*i-2)= Y(3*i-2)+AMAT(9*k-8)*X1+AMAT(9*k-7)*X2 &
                                     +AMAT(9*k-6)*X3
    Y(3*i-1)= Y(3*i-1)+AMAT(9*k-5)*X1+AMAT(9*k-4)*X2 &
                                     +AMAT(9*k-3)*X3
    Y(3*I  )= Y(3*I  )+AMAT(9*k-2)*X1+AMAT(9*k-1)*X2 &
                                     +AMAT(9*k  )*X3
  enddo
enddo
```

ブロックとして記憶（3/3）

- 計算の安定化
  - 対角成分で割るのではなく，対角ブロックの完全LU分解を求めて解く
  - 特に悪条件問題で有効





Global変数表：pfem_util.h/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	[NODGRPtot+1]	I	各節点グループに含まれる節点数（累積）
NODGRP_ITEM	I	[NODGRP_INDEX[NODGRPTOT+1]]	I	節点グループに含まれる節点
NODGRP_NAME	C80	[NODGRP_INDEX[NODGRPTOT+1]]	I	節点グループ名
NL，NU	I		O	各節点非対角成分数（上三角・下三角）
NPL，NPU	I		O	非対角成分総数（上三角・下三角）
D	R	[9*NP]	O	全体行列：対角ブロック
B,X	R	[3*NP]	O	右辺ベクトル，未知数ベクトル

Global変数表：pfem_util.h/f（2/4）

変数名	種別	サイズ	I/O	内　容
ALUG	R	[9*NP]	O	全体行列：対角ブロックの完全LU分解
AL,AU	R	[9*NPL],[9*NPU]	O	全体行列：上・下三角成分
indexL, indexU	I	[NP+1]	O	全体行列：非零非対角ブロック数
itemL, itemU	I	[NPL], [NPU]	O	全体行列：上・下三角ブロック（列番号）
INL, INU	I	[NP]	O	各節点の上・下三角ブロック数
IAL, IAU	I	[NP][NL],[NP][NU]	O	各節点の上・下三角ブロック（列番号）
IWKX	I	[NP][2]	O	ワーク用配列
METHOD	I		I	反復解法（=1に固定）
PRECOND	I		I	前処理手法（=0：ブロックSSOR, =1：ブロック対角スケーリング）
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）
SIGMA_DIAG	R		I	LU分解時の対角成分係数（=1.0に設定）
pfemIarray	I	[100]	O	諸定数（整数）
pfemRarray	R	[100]	O	諸定数（実数）

Global変数表：pfem_util.h/f（3/4）

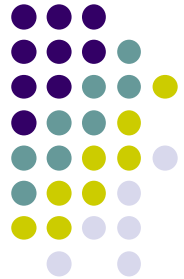
変数名	種別	サイズ	I/O	内　容
O8th	R		I	=0.125
PNQ, PNE, PNT	R	[2][2][8]	O	各ガウス積分点における $\frac{\partial N_i}{\partial \xi}, \frac{\partial N_i}{\partial \eta}, \frac{\partial N_i}{\partial \zeta} (i=1\sim 8)$
POS, WEI	R	[2]	O	各ガウス積分点の座標，重み係数
NCOL1, NCOL2	I	[100]	O	ソート用ワーク配列
SHAPE	R	[2][2][2][8]	O	各ガウス積分点における形状関数 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
ELAST, POISSON	R		I	ヤング率，ポアソン比
SIGMA_N, TAU_N	R	[N][3]	O	節点における垂直，せん断応力成分

Global変数表：pfem_util.h/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	[NEIBPETOT+1]	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	受信テーブル（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
iterPREmax	I		I	ASDDの繰返数

Preconditioned Iterative Solver

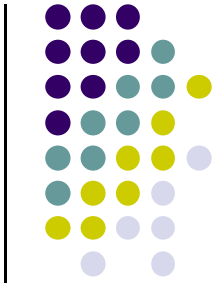
e.g. CG method (Conjugate Gradient)



```
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
```

- Dot products
- Matrix-vector multiplication
- Preconditioners
- DAXPY

局所SGS/SSOR前処理



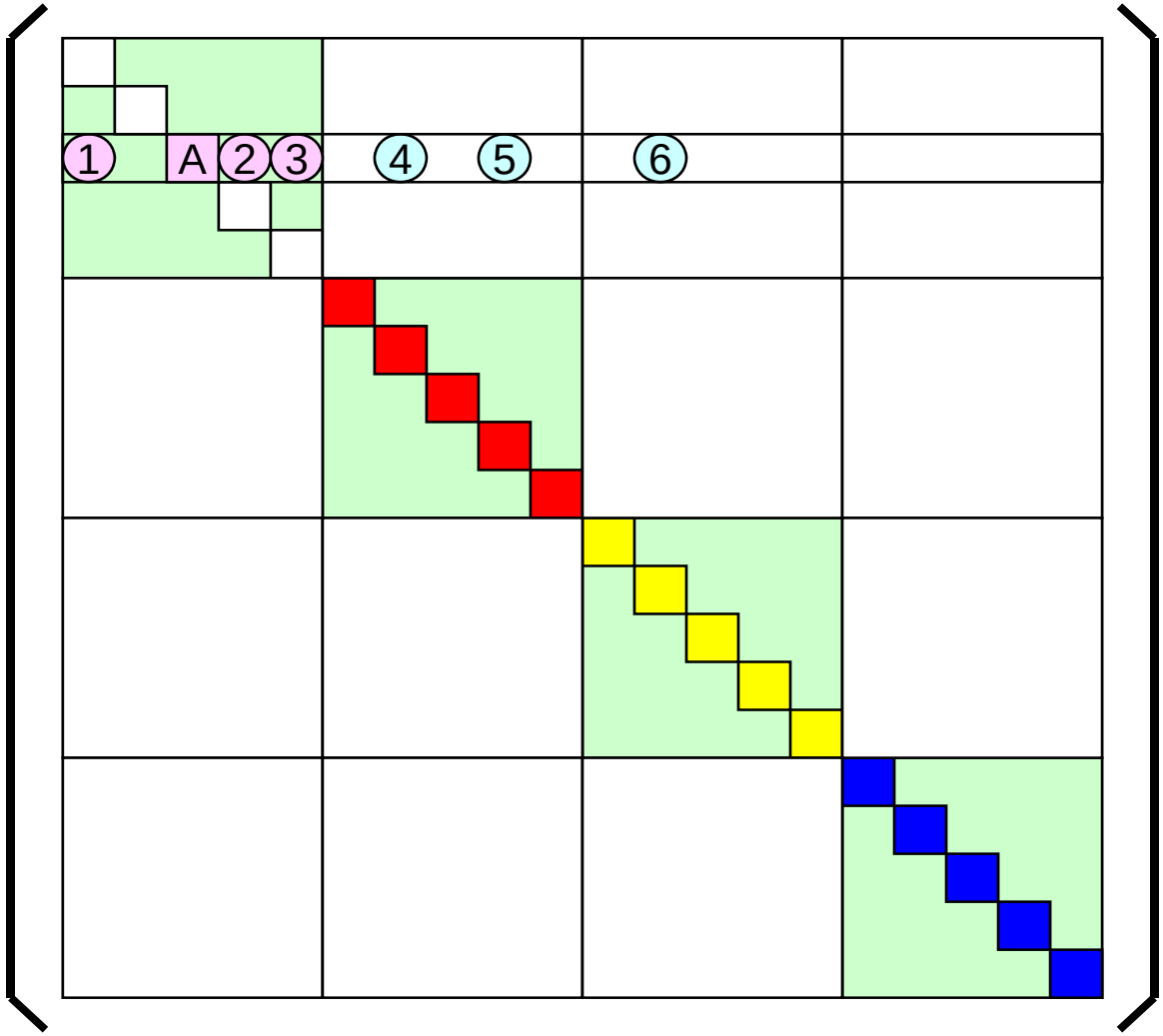
- SGS/SSOR：大域的な処理（前進代入，後退代入）
  - 並列処理に不向き
- 前処理時に領域外（外点）の影響を無視
  - ブロックJacobi型局所前処理
- 本来のSGS/SSORに比べて弱い
  - 領域数が増えると反復回数増加

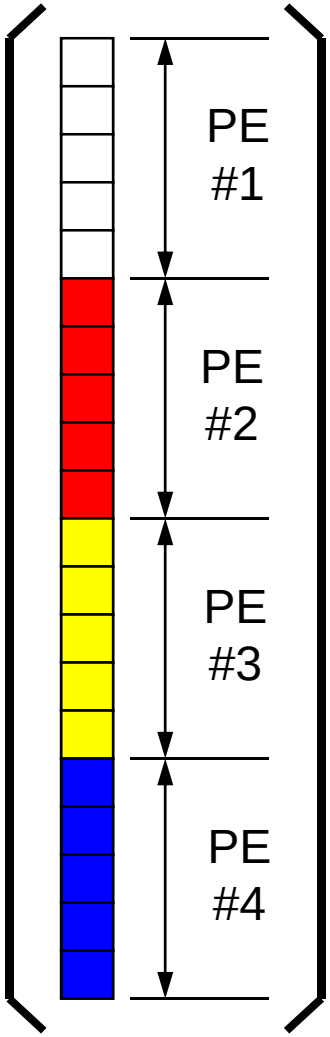
$$(L)\{z\}=\{r\}$$

$$(U)\{z\}=\{z\}$$

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

      do i= 1, N
        WVAL= W(i,Z)
        do k= indexL(i-1)+1, indexL(i)
          WVAL= WVAL -  AL(k) * W(itemL(k),Z)
        enddo
        W(i,Z)= WVAL / D(i)
      enddo

      do i= N, 1, -1
        SW  = 0.0d0
        do k= indexU(i), indexU(i-1)+1, -1
          SW= SW + AU(k) * W(itemU(k),Z)
        enddo
        W(i,Z)= W(i,Z) - SW / D(i)
      enddo
!C===
```

局所SGS/SSOR前処理





Considered :

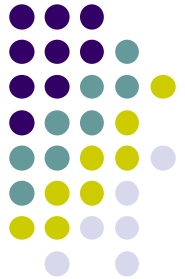
① ② ③

Ignored :

④ ⑤ ⑥

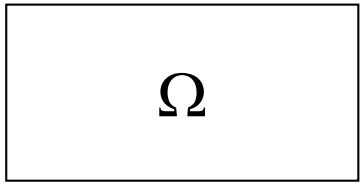
Overlapped Additive Schwartz Domain Decomposition Method

局所前処理手法の安定化：ASDD



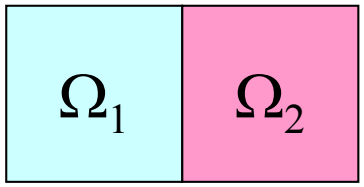
Global Operation

$$Mz = r$$



Local Operation

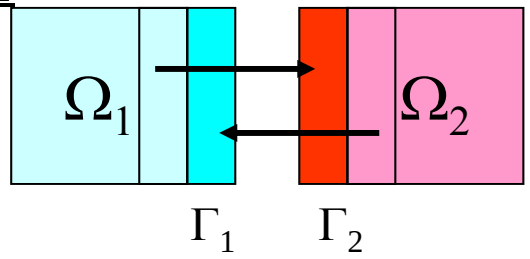
$$z_{\Omega_1} = M_{\Omega_1}^{-1} r_{\Omega_1}, \quad z_{\Omega_2} = M_{\Omega_2}^{-1} r_{\Omega_2}$$



Global Nesting Correction：何回も反復⇒安定

$$z_{\Omega_1}^{n} = z_{\Omega_1}^{n-1} + M_{\Omega_1}^{-1}(r_{\Omega_1} - M_{\Omega_1} z_{\Omega_1}^{n-1} - M_{\Gamma_1} z_{\Gamma_1}^{n-1})$$

$$z_{\Omega_2}^{n} = z_{\Omega_2}^{n-1} + M_{\Omega_2}^{-1}(r_{\Omega_2} - M_{\Omega_2} z_{\Omega_2}^{n-1} - M_{\Gamma_2} z_{\Gamma_2}^{n-1})$$



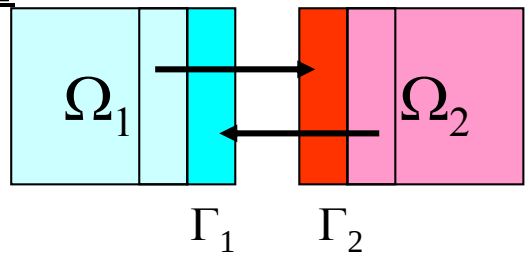
Overlapped Additive Schwartz Domain Decomposition Method

局所前処理手法の安定化：ASDD

Global Nesting Correction：何回も反復⇒安定

$$z_{\Omega_1}^{n} = z_{\Omega_1}^{n-1} + M_{\Omega_1}^{-1}(r_{\Omega_1} - M_{\Omega_1} z_{\Omega_1}^{n-1} - M_{\Gamma_1} z_{\Gamma_1}^{n-1})$$

$$z_{\Omega_2}^{n} = z_{\Omega_2}^{n-1} + M_{\Omega_2}^{-1}(r_{\Omega_2} - M_{\Omega_2} z_{\Omega_2}^{n-1} - M_{\Gamma_2} z_{\Gamma_2}^{n-1})$$



$$\Delta r_{\Omega_1} = r_{\Omega_1} - M_{\Omega_1} z_{\Omega_1}^{n-1} - M_{\Gamma_1} z_{\Gamma_1}^{n-1}$$

$$\Delta z_{\Omega_1} = M_{\Omega_1}^{-1} \Delta r_{\Omega_1} \quad where \quad \Delta z_{\Omega_1} = z_{\Omega_1}^{n} - z_{\Omega_1}^{n-1}$$

$$z_{\Omega_1}^{n} = z_{\Omega_1}^{n-1} + \Delta z_{\Omega_1} = z_{\Omega_1}^{n-1} + M_{\Omega_1}^{-1} \Delta r_{\Omega_1} = z_{\Omega_1}^{n-1} + M_{\Omega_1}^{-1}(r_{\Omega_1} - M_{\Omega_1} z_{\Omega_1}^{n-1} - M_{\Gamma_1} z_{\Gamma_1}^{n-1})$$

Overlapped Additive Schwartz Domain Decomposition Method

Effect of additive Schwartz domain decomposition for solid mechanics example example with $3x44^3$ DOF on Hitachi SR2201, Number of ASDD cycle/iteration= 1, $\varepsilon = 10^{-8}$

	NO Additive Schwartz			WITH Additive Schwartz		
PE #	Iter. #	Sec.	Speed Up	Iter.#	Sec.	Speed Up
1	204	233.7	-	144	325.6	-
2	253	143.6	1.63	144	163.1	1.99
4	259	74.3	3.15	145	82.4	3.95
8	264	36.8	6.36	146	39.7	8.21
16	262	17.4	13.52	144	18.7	17.33
32	268	9.6	24.24	147	10.2	31.80
64	274	6.6	35.68	150	6.5	50.07

SOLVE33（1/4）：INPUT_CNTL

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "pfem_util.h"
#include "allocate.h"
extern FILE *fp_log;
extern void CG_3();
void SOLVE33()
{
        int i,j,k,ii,L;
        KREAL ALU[3][3];
        KREAL PW[3];
        double ALO;

        int  ERROR, ICFLAG=0;
        CHAR_LENGTH BUF;

/**
        +------------+
        | PARAMETERs |
        +------------+
**/
        ITER      = pfemIarray[0];
        METHOD    = pfemIarray[1];
        PRECOND   = pfemIarray[2];
        NSET      = pfemIarray[3];  0
        iterPREmax= pfemIarray[4];
        NREST     = pfemIarray[5];  使用せず

        RESID     = pfemRarray[0];
        SIGMA_DIAG= pfemRarray[1];  1.0

        if( iterPREmax  < 1   ) iterPREmax= 1;
        if (iterPREmax  > 4   ) iterPREmax= 4;
```

Overlapped Additive Schwartz Domain Decomposition Method

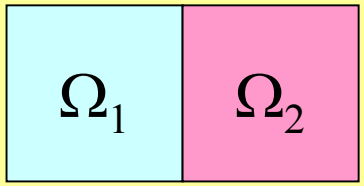
局所前処理手法の安定化：ASDD

do iterPRE= 1, iterPREmax

Local Operation (前進後退代入)

$$calc. \quad M_{\Omega_1}^{-1}(r_{\Omega_1} - M_{\Omega_1} z_{\Omega_1}^{n-1} - M_{\Gamma_1} z_{\Gamma_1}^{n-1})$$

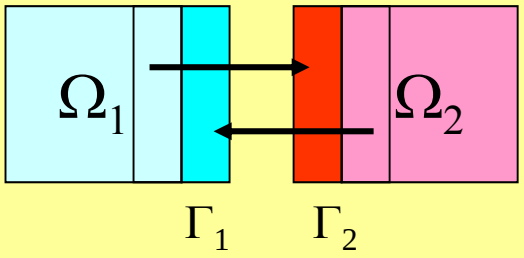
$$calc. \quad M_{\Omega_2}^{-1}(r_{\Omega_2} - M_{\Omega_2} z_{\Omega_2}^{n-1} - M_{\Gamma_2} z_{\Gamma_2}^{n-1})$$



Global Nesting Correction：何回も反復⇒安定

$$z_{\Omega_1}^{n} = z_{\Omega_1}^{n-1} + M_{\Omega_1}^{-1}(r_{\Omega_1} - M_{\Omega_1} z_{\Omega_1}^{n-1} - M_{\Gamma_1} z_{\Gamma_1}^{n-1})$$

$$z_{\Omega_2}^{n} = z_{\Omega_2}^{n-1} + M_{\Omega_2}^{-1}(r_{\Omega_2} - M_{\Omega_2} z_{\Omega_2}^{n-1} - M_{\Gamma_2} z_{\Gamma_2}^{n-1})$$



enddo

SOLVE33 (2/4)

```
/**
        +-----------+
        | BLOCK LUs |
        +-----------+
**/
        if( ICFLAG == 0 ){
                ALUG  =(KREAL*)allocate_vector(sizeof(KREAL),9*N);
                ICFLAG= 1;
                strcpy(BUF.name,"### LINEAR SOLVER:  3x3 Block" );

                if (METHOD == 1) strcat(BUF.name,"ssCG");
                if (METHOD == 2) strcat(BUF.name,"ssBiCGSTAB");

                if (PRECOND == 0) {
                    strcat(BUF.name,"BILU(0)-no ASDD");
                }

                if (PRECOND != 0) strcat(BUF.name,"Block Scaling");

                        fprintf(stdout,"%s¥n",BUF.name);
                        fprintf(fp_log,"%s¥n",BUF.name);
        }
```

SOLVE33（3/4）

```
if( NSET == 0 ){
     for(i=0;i<9*N;i++) ALUG[i]=0.0;
          for( ii=0;ii<N;ii++){
               ALU[0][0]= D[9*ii  ]*SIGMA_DIAG;
               ALU[0][1]= D[9*ii+1];
               ALU[0][2]= D[9*ii+2];
               ALU[1][0]= D[9*ii+3];
               ALU[1][1]= D[9*ii+4]*SIGMA_DIAG;
               ALU[1][2]= D[9*ii+5];
               ALU[2][0]= D[9*ii+6];
               ALU[2][1]= D[9*ii+7];
               ALU[2][2]= D[9*ii+8]*SIGMA_DIAG;

               for(k=0;k<3;k++){
                    L=k;
                    ALO=fabs(ALU[L][k]);
                    for( i=k+1;i<3;i++){
                         if( fabs(ALU[i][k]) > ALO ){
                              L=i;
                              ALO=fabs(ALU[L][k]);}
                    }
                    ALU[k][k]= 1.e0/ALU[k][k];
                    for(i=k+1;i<3;i++){
                         ALU[i][k]*=ALU[k][k];
                              for(j=k+1;j<3;j++){
                                   PW[j]=ALU[i][j] - ALU[i][k]*ALU[k][j];}
                              for(j=k+1;j<3;j++){
                                   ALU[i][j]=PW[j];}
                    }
               }
               ALUG[9*ii  ]=ALU[0][0];
               ALUG[9*ii+1]=ALU[0][1];
               ALUG[9*ii+2]=ALU[0][2];
               ALUG[9*ii+3]=ALU[1][0];
               ALUG[9*ii+4]=ALU[1][1];
               ALUG[9*ii+5]=ALU[1][2];
               ALUG[9*ii+6]=ALU[2][0];
               ALUG[9*ii+7]=ALU[2][1];
               ALUG[9*ii+8]=ALU[2][2];
     }
}
```

ALUG：Dの完全LU分解
SIGMA_DIAG= 1.0
（INPUT_CNTL）

SOLVE33（3/4）

```
      if( NSET == 0 ){
           for(i=0;i<9*N;i++) ALUG[i]=0.0;
                for( ii=0;ii<N;ii++){
                     ALU[0][0]= D[9*ii  ]*SIGMA_DIAG;
                     ALU[0][1]= D[9*ii+1];
                     ALU[0][2]= D[9*ii+2];
                     ALU[1][0]= D[9*ii+3];
                     ALU[1][1]= D[9*ii+4]*SIGMA_DIAG;
                     ALU[1][2]= D[9*ii+5];
                     ALU[2][0]= D[9*ii+6];
                     ALU[2][1]= D[9*ii+7];
                     ALU[2][2]= D[9*ii+8]*SIGMA_DIAG;

                     for(k=0;k<3;k++){
                          L=k;
                          ALO=fabs(ALU[L][k]);
                          for( i=k+1;i<3;i++){
                               if( fabs(ALU[i][k]) > ALO ){
                                    L=i;
                                    ALO=fabs(ALU[L][k]);}
                          }
                          ALU[k][k]= 1.e0/ALU[k][k];
                          for(i=k+1;i<3;i++){
                               ALU[i][k]*=ALU[k][k];
                                    for(j=k+1;j<3;j++){
                                         PW[j]=ALU[i][j] - ALU[i][k]*ALU[k][j];}
                                    for(j=k+1;j<3;j++){
                                         ALU[i][j]=PW[j];}
                          }
                     }
                     ALUG[9*ii  ]=ALU[0][0];
                     ALUG[9*ii+1]=ALU[0][1];
                     ALUG[9*ii+2]=ALU[0][2];
                     ALUG[9*ii+3]=ALU[1][0];
                     ALUG[9*ii+4]=ALU[1][1];
                     ALUG[9*ii+5]=ALU[1][2];
                     ALUG[9*ii+6]=ALU[2][0];
                     ALUG[9*ii+7]=ALU[2][1];
                     ALUG[9*ii+8]=ALU[2][2];
           }
      }
```

ALUG：Dの完全LU分解
SIGMA_DIAG= 1.0
（INPUT_CNTL）

SOLVE33（3/4）

```
      if( NSET == 0 ){
           for(i=0;i<9*N;i++) ALUG[i]=0.0;
                for( ii=0;ii<N;ii++){
                     ALU[0][0]= D[9*ii  ]*SIGMA_DIAG;
                     ALU[0][1]= D[9*ii+1];
                     ALU[0][2]= D[9*ii+2];
                     ALU[1][0]= D[9*ii+3];
                     ALU[1][1]= D[9*ii+4]*SIGMA_DIAG;
                     ALU[1][2]= D[9*ii+5];
                     ALU[2][0]= D[9*ii+6];
                     ALU[2][1]= D[9*ii+7];
                     ALU[2][2]= D[9*ii+8]*SIGMA_DIAG;

                     for(k=0;k<3;k++){
                          L=k;
                          ALO=fabs(ALU[L][k]);
                          for( i=k+1;i<3;i++){
                               if( fabs(ALU[i][k]) > ALO ){
                                    L=i;
                                    ALO=fabs(ALU[L][k]);}
                          }
                          ALU[k][k]= 1.e0/ALU[k][k];
                          for(i=k+1;i<3;i++){
                               ALU[i][k]*=ALU[k][k];
                                    for(j=k+1;j<3;j++){
                                         PW[j]=ALU[i][j] - ALU[i][k]*ALU[k][j];}
                                    for(j=k+1;j<3;j++){
                                         ALU[i][j]=PW[j];}
                          }
                     }
                     ALUG[9*ii  ]=ALU[0][0];
                     ALUG[9*ii+1]=ALU[0][1];
                     ALUG[9*ii+2]=ALU[0][2];
                     ALUG[9*ii+3]=ALU[1][0];
                     ALUG[9*ii+4]=ALU[1][1];
                     ALUG[9*ii+5]=ALU[1][2];
                     ALUG[9*ii+6]=ALU[2][0];
                     ALUG[9*ii+7]=ALU[2][1];
                     ALUG[9*ii+8]=ALU[2][2];
           }
      }
```

LU分解・完全pivoting付き
絶対値の大きい成分が分母となるようにする

ALUG：Dの完全LU分解

SOLVE33 (4/4)

```
/***
        +------------------+
        | ITERATIVE solver |
        +------------------+
***/
        if (METHOD == 1 ) {
                CG_3( N, NP, NPL, NPU, D, AL, indexL, itemL, AU, indexU, itemU,
                        B, X,  ALUG, RESID, ITER, &ERROR, my_rank,
                        NEIBPETOT,NEIBPE,IMPORT_INDEX,IMPORT_ITEM,
                        EXPORT_INDEX,EXPORT_ITEM,
                        PRECOND, iterPREmax);
        }

        ITERactual= ITER;
}
```

CG（初期化）（1/3）

```
#include <stdio.h>
#include <math.h>
#include "mpi.h"
#include "precision.h"
#include "allocate.h"
extern FILE *fp_log;
extern void SOLVER_SEND_RECV_3 ();

void  CG_3(
                KINT N,KINT NP,KINT NPL, KINT NPU,KREAL D[],
                KREAL AL[],KINT INL[], KINT IAL[],
                KREAL AU[],KINT INU[], KINT IAU[],
                KREAL B[],KREAL X[],KREAL ALU[],
                KREAL RESID,KINT ITER, KINT *ERROR,int my_rank,
                int NEIBPETOT,int NEIBPE[],
                int IMPORT_INDEX[], int IMPORT_ITEM[],
                int EXPORT_INDEX[], int EXPORT_ITEM[],
                KINT  PRECOND, KINT iterPREmax)
{
        int i,j,k;
        int ieL,isL,ieU,isU;
        double X1,X2,X3;
        double WVAL1,WVAL2,WVAL3;
        double SW1,SW2,SW3; WV1,WV2,WV3;
        double BNRM20,BNRM2,DNRM20,DNRM2;
        double S1_TIME,E1_TIME;
        double ALPHA,BETA;
        double C1,C10,RHO,RHO0,RHO1;
        int    iterPRE;
        int    indexA,indexB;

        KREAL *WS,*WR;
        KREAL **WW;

        KINT R=0,Z=1,Q=1,P=2,ZP=3;
        KINT MAXIT;
        KREAL TOL,W,SS;

        double COMPtime,COMMtime,R1;
        double START_TIME,END_TIME;
```

Variables/Arrays

Global	I/R	Size	CG_3
N, NP, NPL, NPU	I		N, NP, NPL, NPU
D, B, X	R	[3*NP]	D, B, X
AL, AU	R	[9*NPL], [9*NPU]	AL, AU
indexL, indexU	I	[NP+1]	INL, INU
itemL, itemU	I	[NPL], [NPU]	IAL, IAU
ALUG	R	[9*NP]	ALU
RESID	R		RESID
ITER	I		ITER
PRECOND	I		PRECOND
iterPREmax	I		iterPREmax
	R	[4][3*NP]	WW

SOLVER_SEND_RECV_3 (1/2)

```
#include <stdio.h>
#include <math.h>
#include "mpi.h"
#include "precision.h"
#include "allocate.h"
static MPI_Status  *sta1,*sta2;
static MPI_Request *req1,*req2;
static KINT NFLAG=0;
extern FILE *fp_log;
void SOLVER_SEND_RECV_3( int N, int NEIBPETOT,
        int NEIBPE[], int IMPORT_INDEX[], int IMPORT_ITEM[],
        int EXPORT_INDEX[], int EXPORT_ITEM[],
        KREAL WS[], KREAL WR[], KREAL X[], int my_rank)
{
        int  ii,k,neib,istart,inum;
/***
        INIT.
***/
        if( NFLAG == 0 ){
                sta1=(MPI_Status*)allocate_vector(sizeof(MPI_Status),NEIBPETOT);
                sta2=(MPI_Status*)allocate_vector(sizeof(MPI_Status),NEIBPETOT);
                req1=(MPI_Request*)allocate_vector(sizeof(MPI_Request),NEIBPETOT);
                req2=(MPI_Request*)allocate_vector(sizeof(MPI_Request),NEIBPETOT);
                NFLAG=1;}
/***
        SEND
***/
        for( neib=1;neib<=NEIBPETOT;neib++){
                istart=EXPORT_INDEX[neib-1];
                inum  =EXPORT_INDEX[neib]-istart;
                for( k=istart+1;k<=istart+inum;k++){
                        ii=3*EXPORT_ITEM[k-1];
                        WS[3*k-3]=X[ii-3];
                        WS[3*k-2]=X[ii-2];
                        WS[3*k-1]=X[ii-1];
                }

                MPI_Isend(&WS[3*istart],3*inum,MPI_DOUBLE,
                                NEIBPE[neib-1],0,MPI_COMM_WORLD,&req1[neib-1]);
        }
```

SOLVER_SEND_RECV_3 (2/2)

```
/***
        RECEIVE
***/
        for( neib=1;neib<=NEIBPETOT;neib++){
                istart=IMPORT_INDEX[neib-1];
                inum  =IMPORT_INDEX[neib]-istart;
                MPI_Irecv(&WR[3*istart],3*inum,MPI_DOUBLE,
                                        NEIBPE[neib-1],0,MPI_COMM_WORLD,&req2[neib-1]);
        }

        MPI_Waitall (NEIBPETOT, req2, sta2);

        for( neib=1;neib<=NEIBPETOT;neib++){
                istart=IMPORT_INDEX[neib-1];
                inum  =IMPORT_INDEX[neib]-istart;
                for( k=istart+1;k<=istart+inum;k++){
                        ii   = 3*IMPORT_ITEM[k-1];
                        X[ii-3]=WR[3*k-3];
                        X[ii-2]=WR[3*k-2];
                        X[ii-1]=WR[3*k-1];
                }
        }

        MPI_Waitall (NEIBPETOT, req1, sta1);

}
```

CG（初期化）（2/3）

```
/***
        +-------+
        | INIT. |
        +-------+
***/
        ERROR= 0;

        COMPtime=0.0;
        COMMtime=0.0;

        WW=(KREAL**) allocate_matrix(sizeof(KREAL),4,3*NP);
        WS=(KREAL* ) allocate_vector(sizeof(KREAL),3*NP);
        WR=(KREAL* ) allocate_vector(sizeof(KREAL),3*NP);

        MAXIT  = ITER;
        TOL    = RESID;

        for(i=0;i<NP;i++){
                X[i]=0.0;
        }
        for(i=0;i<4;i++) for(j=0;j<3*NP;j++) WW[i][j]=0.0;
        for(i=0;i<3*NP;i++) WS[i]=0.0;
        for(i=0;i<3*NP;i++) WR[i]=0.0;
```

CG（初期化）（3/3）

```
/***
        +-------+
        | INIT. |
        +-------+
***/
        ERROR= 0;

        COMPtime=0.0;
        COMMtime=0.0;

        WW=(KREAL**) allocate_matrix(sizeof(KREAL),4,3*NP);
        WS=(KREAL* ) allocate_vector(sizeof(KREAL),3*NP);   送信バッファ
        WR=(KREAL* ) allocate_vector(sizeof(KREAL),3*NP);   受信バッファ

        MAXIT  = ITER;
        TOL    = RESID;

        for(i=0;i<NP;i++){
                X[i]=0.0;
        }
        for(i=0;i<4;i++) for(j=0;j<3*NP;j++) WW[i][j]=0.0;
        for(i=0;i<3*NP;i++) WS[i]=0.0;
        for(i=0;i<3*NP;i++) WR[i]=0.0;
/**
```

前処理：SGS/SSOR（1/5）

```
/**
        +----------------+
        | {z}= [Minv]{r} |
        +----------------+
**/
     if( PRECOND == 0 ){
/**
     Block SSOR
**/
 for( i=0;i<N;i++){
          WW[ZP][3*i  ]=WW[R][3*i  ];
          WW[ZP][3*i+1]=WW[R][3*i+1];
          WW[ZP][3*i+2]=WW[R][3*i+2];
        }

        for( i=0;i<NP;i++){
          WW[Z][3*i  ]=0.e0;
          WW[Z][3*i+1]=0.e0;
          WW[Z][3*i+2]=0.e0;
        }


        for(iterPRE=0;iterPRE<iterPREmax;iterPRE++){        loops for ASDD
          for( i=N;i<NP;i++){
                WW[ZP][3*i  ]=0.0;
                WW[ZP][3*i+1]=0.0;
                WW[ZP][3*i+2]=0.0;
          }
```

前処理：SGS/SSOR（2/5）

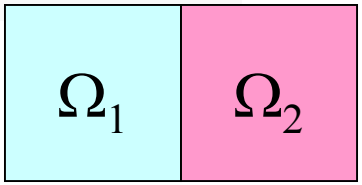
$$(L)\{z\}=\{r\}$$

```
/**
     FORWARD
**/

 for( i=0;i<N;i++){
          SW1= WW[ZP][3*i  ];
          SW2= WW[ZP][3*i+1];
          SW3= WW[ZP][3*i+2];

          isL=INL[i];
          ieL=INL[i+1];
          for(j=isL;j<ieL;j++){
            k=IAL[j];
            X1= WW[ZP][3*k  ];
            X2= WW[ZP][3*k+1];
            X3= WW[ZP][3*k+2];
            SW1+=  - AL[9*j  ]*X1 - AL[9*j+1]*X2 - AL[9*j+2]*X3;
            SW2+=  - AL[9*j+3]*X1 - AL[9*j+4]*X2 - AL[9*j+5]*X3;
            SW3+=  - AL[9*j+6]*X1 - AL[9*j+7]*X2 - AL[9*j+8]*X3;
          }

          X1= SW1;
          X2= SW2;
          X3= SW3;
          X2= X2 - ALU[9*i+3]*X1;
          X3= X3 - ALU[9*i+6]*X1 - ALU[9*i+7]*X2;
          X3= ALU[9*i+8]*  X3;
          X2= ALU[9*i+4]*( X2 - ALU[9*i+5]*X3 );
          X1= ALU[9*i  ]*( X1 - ALU[9*i+2]*X3 - ALU[9*i+1]*X2);

          WW[ZP][3*i  ]= X1;
          WW[ZP][3*i+1]= X2;
          WW[ZP][3*i+2]= X3;
        }
```

indexL ⇒ INL
itemL ⇒ IAL

$$calc.\quad M_{\Omega_1}^{-1}(r_{\Omega_1}-M_{\Omega_1}z_{\Omega_1}^{n-1}-M_{\Gamma_1}z_{\Gamma_1}^{n-1})$$

$$calc.\quad M_{\Omega_2}^{-1}(r_{\Omega_2}-M_{\Omega_2}z_{\Omega_2}^{n-1}-M_{\Gamma_2}z_{\Gamma_2}^{n-1})$$



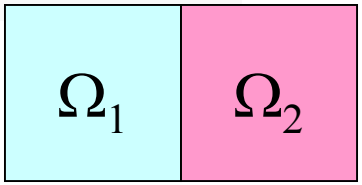
前処理：SGS/SSOR（2/5）：初期

$$(L)\{z\}=\{r\}$$

```
/**
     FORWARD
**/

 for( i=0;i<N;i++){
          SW1= WW[ZP][3*i  ];
          SW2= WW[ZP][3*i+1];
          SW3= WW[ZP][3*i+2];

          isL=INL[i];
          ieL=INL[i+1];
          for(j=isL;j<ieL;j++){
            k=IAL[j];
            X1= WW[ZP][3*k  ];
            X2= WW[ZP][3*k+1];
            X3= WW[ZP][3*k+2];
            SW1+=  - AL[9*j  ]*X1 - AL[9*j+1]*X2 - AL[9*j+2]*X3;
            SW2+=  - AL[9*j+3]*X1 - AL[9*j+4]*X2 - AL[9*j+5]*X3;
            SW3+=  - AL[9*j+6]*X1 - AL[9*j+7]*X2 - AL[9*j+8]*X3;
          }

          X1= SW1;
          X2= SW2;
          X3= SW3;
          X2= X2 - ALU[9*i+3]*X1;
          X3= X3 - ALU[9*i+6]*X1 - ALU[9*i+7]*X2;
          X3= ALU[9*i+8]*  X3;
          X2= ALU[9*i+4]*( X2 - ALU[9*i+5]*X3 );
          X1= ALU[9*i  ]*( X1 - ALU[9*i+2]*X3 - ALU[9*i+1]*X2);

          WW[ZP][3*i  ]= X1;
          WW[ZP][3*i+1]= X2;
          WW[ZP][3*i+2]= X3;
        }
```

indexL ⇒ INL
itemL ⇒ IAL

$$calc.\quad M_{\Omega_1}^{-1} r_{\Omega_1}$$
$$calc.\quad M_{\Omega_2}^{-1} r_{\Omega_2}$$



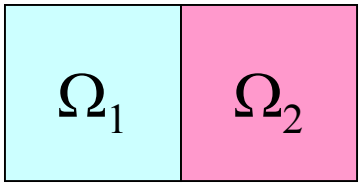
前処理：SGS/SSOR（3/5）

$$(U)\{z\}=\{z\}$$

```
/**
      BACKWARD
**/
      for(i=N-1;i>=0;i--){
            isU= INU[i];
            ieU= INU[i+1];
            SW1= 0.e0;
            SW2= 0.e0;
            SW3= 0.e0;

            for(j=isU;j<ieU;j++){
              k=IAU[j];

              X1=WW[ZP][3*k  ];
              X2=WW[ZP][3*k+1];
              X3=WW[ZP][3*k+2];
              SW1+=  + AU[9*j  ]*X1 + AU[9*j+1]*X2 + AU[9*j+2]*X3;
              SW2+=  + AU[9*j+3]*X1 + AU[9*j+4]*X2 + AU[9*j+5]*X3;
              SW3+=  + AU[9*j+6]*X1 + AU[9*j+7]*X2 + AU[9*j+8]*X3;
            }

            X1= SW1;
            X2= SW2;
            X3= SW3;
            X2= X2 - ALU[9*i+3]*X1;
            X3= X3 - ALU[9*i+6]*X1 - ALU[9*i+7]*X2;
            X3= ALU[9*i+8]*  X3;
            X2= ALU[9*i+4]*( X2 - ALU[9*i+5]*X3 );
            X1= ALU[9*i  ]*( X1 - ALU[9*i+2]*X3 - ALU[9*i+1]*X2);
            WW[ZP][3*i  ]+= -X1;
            WW[ZP][3*i+1]+= -X2;
            WW[ZP][3*i+2]+= -X3;
      }
```

indexU ⇒ INU
itemU ⇒ IAU

$$calc.\quad M_{\Omega_1}^{-1}(r_{\Omega_1}-M_{\Omega_1}z_{\Omega_1}^{n-1}-M_{\Gamma_1}z_{\Gamma_1}^{n-1})$$
$$calc.\quad M_{\Omega_2}^{-1}(r_{\Omega_2}-M_{\Omega_2}z_{\Omega_2}^{n-1}-M_{\Gamma_2}z_{\Gamma_2}^{n-1})$$

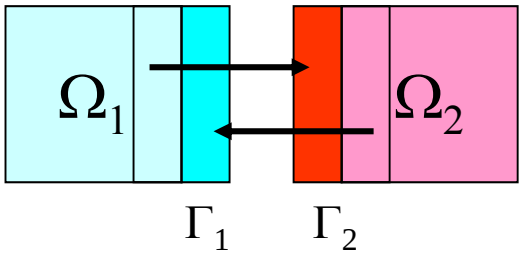


前処理：SGS/SSOR（4/5）

```
/**
        additive Schwartz
**/
         SOLVER_SEND_RECV_3
         ( NP, NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_ITEM,
           EXPORT_INDEX, EXPORT_ITEM, WS, WR, WW[ZP], my_rank);

         indexA= ZP;
         indexB= Z;

         for( i=0;i<NP;i++){
            WW[indexB][3*i  ]+=WW[indexA][3*i  ];
            WW[indexB][3*i+1]+=WW[indexA][3*i+1];
            WW[indexB][3*i+2]+=WW[indexA][3*i+2];
         }

         if (iterPRE == iterPREmax ) goto LABEL750;
```

$$z_{\Omega_1}^{n} = z_{\Omega_1}^{n-1} + M_{\Omega_1}^{-1}(r_{\Omega_1} - M_{\Omega_1} z_{\Omega_1}^{n-1} - M_{\Gamma_1} z_{\Gamma_1}^{n-1})$$

$$z_{\Omega_2}^{n} = z_{\Omega_2}^{n-1} + M_{\Omega_2}^{-1}(r_{\Omega_2} - M_{\Omega_2} z_{\Omega_2}^{n-1} - M_{\Gamma_2} z_{\Gamma_2}^{n-1})$$

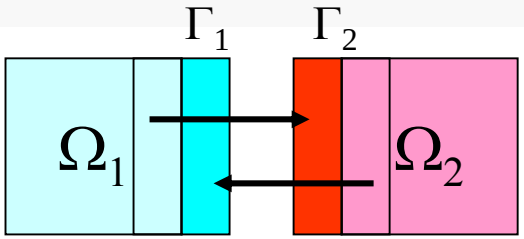


前処理：SGS/SSOR（5/5）

```
          for( j=0;j<N;j++){
             X1= WW[indexB][3*j  ];
             X2= WW[indexB][3*j+1];
             X3= WW[indexB][3*j+2];
             WV1= WW[R][3*j  ] - D[9*j  ]*X1 - D[9*j+1]*X2 - D[9*j+2]*X3;
             WV2= WW[R][3*j+1] - D[9*j+3]*X1 - D[9*j+4]*X2 - D[9*j+5]*X3;
             WV3= WW[R][3*j+2] - D[9*j+6]*X1 - D[9*j+7]*X2 - D[9*j+8]*X3;
             for(k=INL[j];k<INL[j+1];k++){
                 i=IAL[k];
                 X1= WW[indexB][3*i  ];
                 X2= WW[indexB][3*i+1];
                 X3= WW[indexB][3*i+2];
                 WV1+=  - AL[9*k  ]*X1 - AL[9*k+1]*X2 - AL[9*k+2]*X3;
                 WV2+=  - AL[9*k+3]*X1 - AL[9*k+4]*X2 - AL[9*k+5]*X3;
                 WV3+=  - AL[9*k+6]*X1 - AL[9*k+7]*X2 - AL[9*k+8]*X3;
             }
             for(k=INU[j];k<INU[j+1];k++){
                 i=IAU[k];
                 X1= WW[indexB][3*i  ];
                 X2= WW[indexB][3*i+1];
                 X3= WW[indexB][3*i+2];
                 WV1+=  - AU[9*k  ]*X1 - AU[9*k+1]*X2 - AU[9*k+2]*X3;
                 WV2+=  - AU[9*k+3]*X1 - AU[9*k+4]*X2 - AU[9*k+5]*X3;
                 WV3+=  - AU[9*k+6]*X1 - AU[9*k+7]*X2 - AU[9*k+8]*X3;
             }
             WW[indexA][3*j  ]=WV1;
             WW[indexA][3*j+1]=WV2;
             WW[indexA][3*j+2]=WV3;
          }
           LABEL750:
           continue;}
}
```

$$z_{\Omega_1}^{n} = z_{\Omega_1}^{n-1} + M_{\Omega_1}^{-1}\left(r_{\Omega_1} - M_{\Omega_1} z_{\Omega_1}^{n-1} - M_{\Gamma_1} z_{\Gamma_1}^{n-1}\right)$$

$$z_{\Omega_2}^{n} = z_{\Omega_2}^{n-1} + M_{\Omega_2}^{-1}\left(r_{\Omega_2} - M_{\Omega_2} z_{\Omega_2}^{n-1} - M_{\Gamma_2} z_{\Gamma_2}^{n-1}\right)$$



前処理：Block Scaling

```
      if (PRECOND != 0 ) {
/**
      Block SCALING
**/
        for(i=0;i<N;i++){
             WW[3*i  ][Z]=WW[3*i  ][R];
             WW[3*i+1][Z]=WW[3*i+1][R];
             WW[3*i+2][Z]=WW[3*i+2][R];
        }

        for(i=0;i<N;i++){
             X1=WW[3*i  ][Z];
             X2=WW[3*i+1][Z];
             X3=WW[3*i+2][Z];
             X2= X2 - ALU[9*i+3]*X1;
             X3= X3 - ALU[9*i+6]*X1 - ALU[9*i+7]*X2;
             X3= ALU[9*i+8]*  X3;
             X2= ALU[9*i+4]*( X2 - ALU[9*i+5]*X3 );
             X1= ALU[9*i  ]*( X1 - ALU[9*i+2]*X3 - ALU[9*i+1]*X2);
             WW[3*i  ][Z]= X1;
             WW[3*i+1][Z]= X2;
             WW[3*i+2][Z]= X3;
        }
      }
```

対角スケーリングの代わりに対角ブロックのLU分解による前進後退代入

行列ベクトル積

```
/***
        +-------------+
        | {q}= [A]{p} |
        +-------------+
***/
     SOLVER_SEND_RECV_3
     ( NP, NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_ITEM,
       EXPORT_INDEX, EXPORT_ITEM, WS, WR, WW[P], my_rank);

     for( j=0;j<N;j++){
          X1=WW[P][3*j  ];
          X2=WW[P][3*j+1];
          X3=WW[P][3*j+2];
          WVAL1= D[9*j  ]*X1 + D[9*j+1]*X2 + D[9*j+2]*X3;
          WVAL2= D[9*j+3]*X1 + D[9*j+4]*X2 + D[9*j+5]*X3;
          WVAL3= D[9*j+6]*X1 + D[9*j+7]*X2 + D[9*j+8]*X3;
          for(k=INL[j];k<INL[j+1];k++){
            i=IAL[k];
            X1=WW[P][3*i  ];
            X2=WW[P][3*i+1];
            X3=WW[P][3*i+2];
            WVAL1+=   AL[9*k  ]*X1 + AL[9*k+1]*X2 + AL[9*k+2]*X3;
            WVAL2+=   AL[9*k+3]*X1 + AL[9*k+4]*X2 + AL[9*k+5]*X3;
            WVAL3+=   AL[9*k+6]*X1 + AL[9*k+7]*X2 + AL[9*k+8]*X3;
          }
          for(k=INU[j];k<INU[j+1];k++){
            i=IAU[k];
            X1=WW[P][3*i  ];
            X2=WW[P][3*i+1];
            X3=WW[P][3*i+2];
            WVAL1+=   AU[9*k  ]*X1 + AU[9*k+1]*X2 + AU[9*k+2]*X3;
            WVAL2+=   AU[9*k+3]*X1 + AU[9*k+4]*X2 + AU[9*k+5]*X3;
            WVAL3+=   AU[9*k+6]*X1 + AU[9*k+7]*X2 + AU[9*k+8]*X3;
          }
          WW[Q][3*j  ]=WVAL1;
          WW[Q][3*j+1]=WVAL2;
          WW[Q][3*j+2]=WVAL3;
        }
```

DAXPY，内積

```
/***
        +---------------------+
        | {x}= {x} + ALPHA*{p} |
        | {r}= {r} - ALPHA*{q} |
        +---------------------+
***/
     for(i=0;i<N;i++){
          X[3*i  ]+=ALPHA *WW[3*i  ][P];
          X[3*i+1]+=ALPHA *WW[3*i+1][P];
          X[3*i+2]+=ALPHA *WW[3*i+2][P];
          WW[3*i  ][R]+= -ALPHA *WW[3*i  ][Q];
          WW[3*i+1][R]+= -ALPHA *WW[3*i+1][Q];
          WW[3*i+2][R]+= -ALPHA *WW[3*i+2][Q];
     }

     DNRM20= 0.e0;
     for(i=0;i<N;i++){
          DNRM20+= WW[3*i  ][R]*WW[3*i  ][R] +
                   WW[3*i+1][R]*WW[3*i+1][R] +
                   WW[3*i+2][R]*WW[3*i+2][R];
     }

     MPI_Allreduce (&DNRM20,&DNRM2, 1, MPI_DOUBLE,MPI_SUM, MPI_COMM_WORLD);

     RESID= sqrt(DNRM2/BNRM2);
```

応 力

- ここまでで求まるのは,
  各節点における「変位」
- 工学的に興味のあるのは
  「応力」
  - 「変位」の微分である
    「ひずみ」から計算

ひずみ⇒応力関係

$$\begin{Bmatrix}\sigma_x\\ \sigma_y\\ \sigma_z\\ \tau_{xy}\\ \tau_{yz}\\ \tau_{zx}\end{Bmatrix} = \frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0\\ \nu & 1-\nu & \nu & 0 & 0 & 0\\ \nu & \nu & 1-\nu & 0 & 0 & 0\\ 0 & 0 & 0 & \frac{1}{2}(1-2\nu) & 0 & 0\\ 0 & 0 & 0 & 0 & \frac{1}{2}(1-2\nu) & 0\\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2}(1-2\nu)\end{bmatrix}\begin{Bmatrix}\varepsilon_x\\ \varepsilon_y\\ \varepsilon_z\\ \gamma_{xy}\\ \gamma_{yz}\\ \gamma_{zx}\end{Bmatrix}$$

$$[D]$$

$$\{\sigma\} = [D]\{\varepsilon\}$$

ひずみ⇒応力関係

$$\begin{Bmatrix}\sigma_x\\ \sigma_y\\ \sigma_z\\ \tau_{xy}\\ \tau_{yz}\\ \tau_{zx}\end{Bmatrix} = \begin{bmatrix} \mathrm{valX} & \mathrm{valA} & \mathrm{valA} & 0 & 0 & 0 \\ \mathrm{valA} & \mathrm{valX} & \mathrm{valA} & 0 & 0 & 0 \\ \mathrm{valA} & \mathrm{valA} & \mathrm{valX} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{valB} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathrm{valB} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathrm{valB} \end{bmatrix} \begin{Bmatrix}\varepsilon_x\\ \varepsilon_y\\ \varepsilon_z\\ \gamma_{xy}\\ \gamma_{yz}\\ \gamma_{zx}\end{Bmatrix}$$

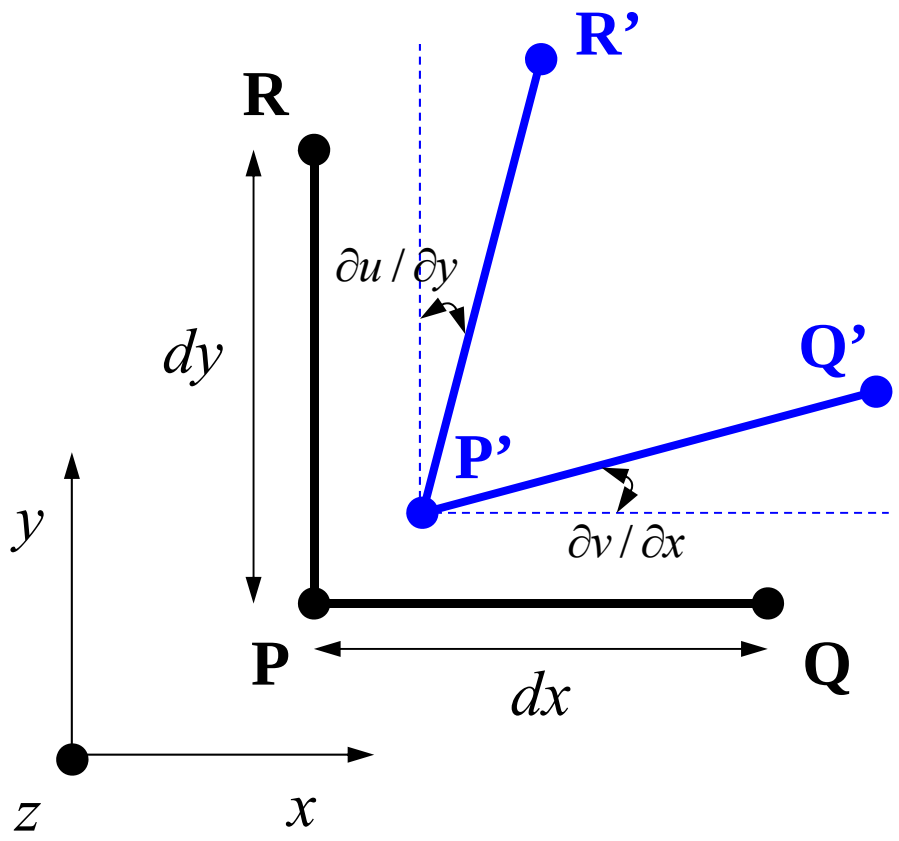
$$[D]$$

$$\{\sigma\} = [D]\{\varepsilon\}$$

垂直ひずみ～変位の関係

- PQ⇒P'Q'

$$\varepsilon_x = \frac{\left\{\left(x + dx + u + \frac{\partial u}{\partial x} dx\right) - (x + u)\right\} - dx}{dx} = \frac{\partial u}{\partial x}$$

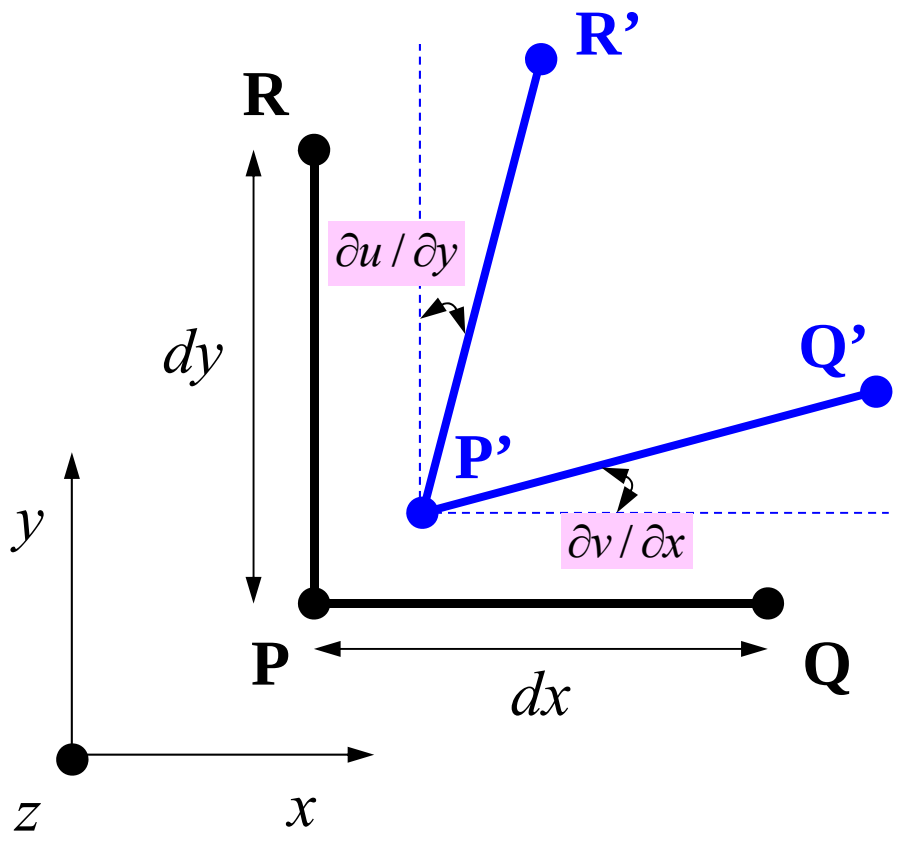


$$\varepsilon_x = \frac{\partial u}{\partial x}$$

$$\varepsilon_y = \frac{\partial v}{\partial y}$$

$$\varepsilon_z = \frac{\partial w}{\partial z}$$

せん断ひずみ～変位の関係



$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}$$

$$\gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}$$

$$\gamma_{zx} = \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}$$

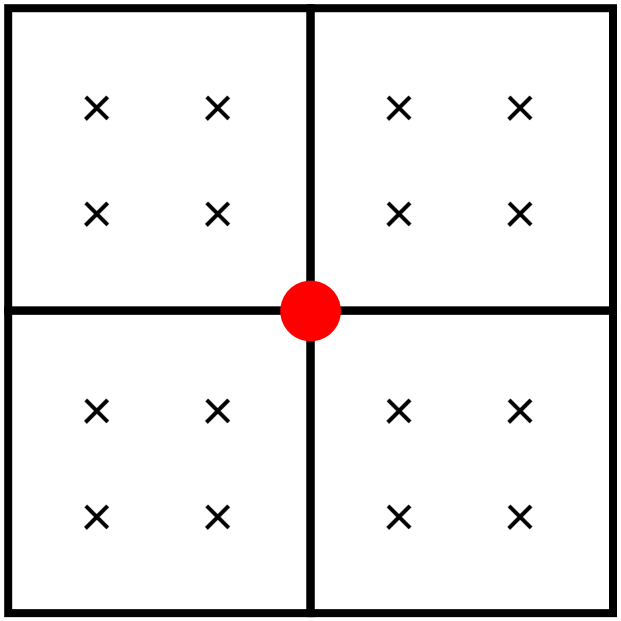
応力の計算

$$\sigma_x = \frac{E}{(1+\nu)(1-2\nu)}\left[(1-\nu)\varepsilon_x + \nu\varepsilon_y + \nu\varepsilon_z\right]$$

$$= \frac{E}{(1+\nu)(1-2\nu)}\left[(1-\nu)\frac{\partial u}{\partial x} + \nu\frac{\partial v}{\partial y} + \nu\frac{\partial w}{\partial z}\right]$$

応 力

- ここまでで求まるのは，各節点における「変位」
- 工学的に興味のあるのは「応力」
  - 「変位」の微分である「ひずみ」から計算



- 正確な応力が求められるのはガウス積分
- 節点における応力
  - 平均値

応力の計算

$$\sigma_x = \frac{E}{(1+\nu)(1-2\nu)}\left[(1-\nu)\varepsilon_x + \nu\varepsilon_y + \nu\varepsilon_z\right]$$

$$= \frac{E}{(1+\nu)(1-2\nu)}\left[(1-\nu)\frac{\partial u}{\partial x} + \nu\frac{\partial v}{\partial y} + \nu\frac{\partial w}{\partial z}\right]$$

- ガラーキン法：要素の平均応力×体積

$$\int_V [N]^T \sigma_x dV = \int_V [N]^T \left\{\frac{E}{(1+\nu)(1-2\nu)}\left[(1-\nu)u_{,x} + \nu v_{,y} + \nu w_{,z}\right]\right\} dV$$

$$= \int_V [N]^T \left\{\frac{E}{(1+\nu)(1-2\nu)}\left((1-\nu)[N_{,x}]\{U\} + \nu[N_{,y}]\{V\} + \nu[N_{,z}]\{W\}\right)\right\} dV$$

1D：要素単位での積分：$\{f\}$

$$N_i = \left(\frac{X_j - x}{L} \right), \quad N_j = \left(\frac{x - X_i}{L} \right) \qquad \frac{dN_i}{dx} = \left(\frac{-1}{L} \right), \quad \frac{dN_j}{dx} = \left(\frac{1}{L} \right)$$

$$\int_V X [N]^T dV = XA \int_0^L \begin{bmatrix} 1 - x/L \\ x/L \end{bmatrix} dx = \frac{XAL}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

物体力

A：断面積，L：要素長さ

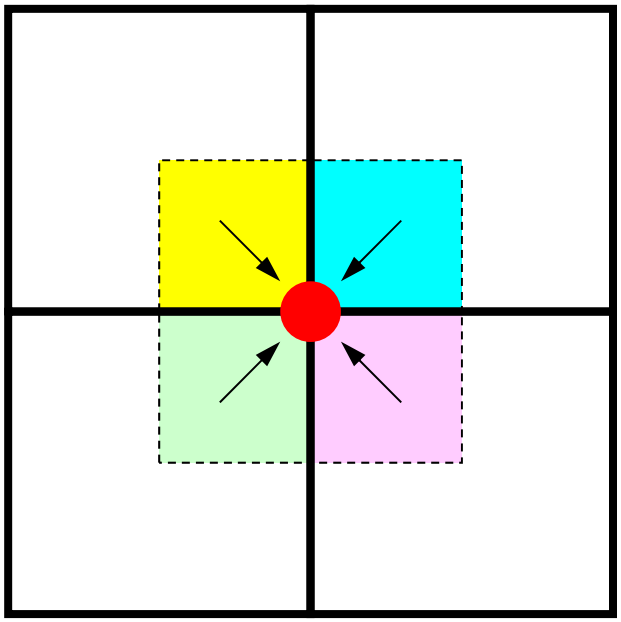
2D：節点応力の求め方

$$\int_V [N]^T \sigma_x dV$$

各節点における，各要素からの
応力×寄与体積

$$\int_V [N]^T dV$$

各節点における寄与体積



$$\therefore \quad \sigma_x = \frac{\sum \int_V [N]^T \sigma_x dV}{\sum \int_V [N]^T dV}$$

各節点における平均応力

RECOVER_STRESS：応力（1/4）

```
#include <math.h>
#include "pfem_util.h"
#include "allocate.h"
extern   void JACOBI();
extern void SOLVER_SEND_RECV_3();
void RECOVER_STRESS()
{
     int  i,k,kk,icel;
     int  ie,je,ip,jp;
     int  ipn,jpn,kpn;
     int  iiS,iiE;
     double   RB;
     double   UUi,VVi,WWi,UUj,VVj,WWj;
     double   valX,valA,valB,E0,POI0,VOL,coef;
     int  in1,in2,in3,in4,in5,in6,in7,in8;
     double   X1,X2,X3,X4,X5,X6,X7,X8;
     double   Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8;
     double   Z1,Z2,Z3,Z4,Z5,Z6,Z7,Z8;
     double   SHi,SHj;
     double   EPS_xx,EPS_yy,EPS_zz,GAM_xy,GAM_xz,GAM_yz;

     KINT nodLOCAL[8];

     SIGMA_N=(KREAL*)allocate_vector(sizeof(KREAL),3*NP);
     TAU_N  =(KREAL*)allocate_vector(sizeof(KREAL),3*NP);

     for(i=0;i<3*NP;i++)     SIGMA_N[i]=0.0;
     for(i=0;i<3*NP;i++)     TAU_N[i]=0.0;
     for(i=0;i<3*NP;i++)     B[i]=0.0;

     for( icel=0;icel<  ICELTOT;icel++){
          E0   =    ELAST;
          POI0=     POISSON;

          valA=     POI0 /    (1.e0-POI0);
          valB=     (1.e0-2.e0*POI0)/(2.e0*(1.e0-POI0));
          valX=     E0*  (1.e0-POI0)/((1.e0+POI0)*(1.e0-2.e0*POI0));

          valA=     valA *    valX;
          valB=     valB *    valX;
```

RECOVER_STRESS：応力（2/4）

```
          in1= ICELNOD[icel][0];
          in2= ICELNOD[icel][1];
          in3= ICELNOD[icel][2];
          in4= ICELNOD[icel][3];
          in5= ICELNOD[icel][4];
          in6= ICELNOD[icel][5];
          in7= ICELNOD[icel][6];
          in8= ICELNOD[icel][7];
         nodLOCAL[0]=  in1;
         nodLOCAL[1]=  in2;
         nodLOCAL[2]=  in3;
         nodLOCAL[3]=  in4;
         nodLOCAL[4]=  in5;
         nodLOCAL[5]=  in6;
         nodLOCAL[6]=  in7;
         nodLOCAL[7]=  in8;
         X1=  XYZ[in1-1][0];
         X2=  XYZ[in2-1][0];
 （中略）
         X7=  XYZ[in7-1][0];
         X8=  XYZ[in8-1][0];
         Y1=  XYZ[in1-1][1];
         Y2=  XYZ[in2-1][1];
 （中略）
         Y7=  XYZ[in7-1][1];
         Y8=  XYZ[in8-1][1];
         Z1=  XYZ[in1-1][2];
         Z2=  XYZ[in2-1][2];
 （中略）
         Z7=  XYZ[in7-1][2];
         Z8=  XYZ[in8-1][2];

/**
     JACOBIAN  &    inv-JACOBIAN
**/
         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          );
```

RECOVER_STRESS：応力（3/4）

```
/**
     MATRIX
**/
          for(ie=0;ie<8;ie++){
               ip= nodLOCAL[ie];
               for(je=0;je<8;je++){
                    jp= nodLOCAL[je];
                    UUj= X[3*jp-3];
                    VVj= X[3*jp-2];
                    WWj= X[3*jp-1];
                    UUi= X[3*ip-3];
                    VVi= X[3*ip-2];
                    WWi= X[3*ip-1];

                    EPS_xx=   0.e0;
                    EPS_yy=   0.e0;
                    EPS_zz=   0.e0;
                    GAM_xy=   0.e0;
                    GAM_xz=   0.e0;
                    GAM_yz=   0.e0;
                    GAM_xy=   0.e0;
                    GAM_xz=   0.e0;
                    GAM_yz=   0.e0;

                    VOL  =    0.e0;
                    for( ipn=0;ipn<2;ipn++){
                         for( jpn=0;jpn<2;jpn++){
                              for( kpn=0;kpn<2;kpn++){
                                   coef= fabs(DETJ[ipn][jpn][kpn])*WEI[ipn]*WEI[jpn]*WEI[kpn];
                                   SHi= SHAPE[ipn][jpn][kpn][ie]           *           coef;

                                   EPS_xx+=  SHi*PNX[ipn][jpn][kpn][je];
                                   EPS_yy+=  SHi*PNY[ipn][jpn][kpn][je];
                                   EPS_zz+=  SHi*PNZ[ipn][jpn][kpn][je];
                                   GAM_xy+=  SHi*PNX[ipn][jpn][kpn][je] * VVj
                                        +    SHi*PNY[ipn][jpn][kpn][je] * UUj;
                                   GAM_xz+=  SHi*PNX[ipn][jpn][kpn][je] * WWj
                                        +    SHi*PNZ[ipn][jpn][kpn][je] * UUj;
                                   GAM_yz+=  SHi*PNY[ipn][jpn][kpn][je] * WWj
                                        +    SHi*PNZ[ipn][jpn][kpn][je] * VVj;
                                   VOL  =    VOL + SHi; }}}
```

各節点における変位

RECOVER_STRESS：応力（3/4）

```
/**
      MATRIX
**/
          for(ie=0;ie<8;ie++){
               ip= nodLOCAL[ie];
               for(je=0;je<8;je++){
                    jp= nodLOCAL[je];
                    UUj= X[3*jp-3];
                    VVj= X[3*jp-2];
                    WWj= X[3*jp-1];
                    UUi= X[3*ip-3];
                    VVi= X[3*ip-2];
                    WWi= X[3*ip-1];

                    EPS_xx=   0.e0;
                    EPS_yy=   0.e0;
                    EPS_zz=   0.e0;
                    GAM_xy=   0.e0;
                    GAM_xz=   0.e0;
                    GAM_yz=   0.e0;
                    GAM_xy=   0.e0;
                    GAM_xz=   0.e0;
                    GAM_yz=   0.e0;

                    VOL  =    0.e0;
                    for( ipn=0;ipn<2;ipn++){
                         for( jpn=0;jpn<2;jpn++){
                              for( kpn=0;kpn<2;kpn++){
                                   coef= fabs(DETJ[ipn][jpn][kpn])*WEI[ipn]*WEI[jpn]*WEI[kpn];
                                   SHi= SHAPE[ipn][jpn][kpn][ie]          *          coef;

                                   EPS_xx+=  SHi*PNX[ipn][jpn][kpn][je];
                                   EPS_yy+=  SHi*PNY[ipn][jpn][kpn][je];
                                   EPS_zz+=  SHi*PNZ[ipn][jpn][kpn][je];
                                   GAM_xy+=  SHi*PNX[ipn][jpn][kpn][je] * VVj
                                        +    SHi*PNY[ipn][jpn][kpn][je] * UUj;
                                   GAM_xz+=  SHi*PNX[ipn][jpn][kpn][je] * WWj
                                        +    SHi*PNZ[ipn][jpn][kpn][je] * UUj;
                                   GAM_yz+=  SHi*PNY[ipn][jpn][kpn][je] * WWj
                                        +    SHi*PNZ[ipn][jpn][kpn][je] * VVj;
                                   VOL  =    VOL + SHi; }}}
```

$$u_{,x} = [N_{,x}]\{U\}, \quad u_{,y} = [N_{,y}]\{U\}, \quad u_{,z} = [N_{,z}]\{U\}$$
$$v_{,x} = [N_{,x}]\{V\}, \quad v_{,y} = [N_{,y}]\{V\}$$
$$w_{,x} = [N_{,x}]\{W\}, \quad w_{,z} = [N_{,z}]\{W\}$$

RECOVER_STRESS：応力（4/4）

```
                    EPS_xx= EPS_xx * UUj;
                    EPS_yy= EPS_yy * VVj;
                    EPS_zz= EPS_zz * WWj;

                    SIGMA_N[3*ip-3]+= valX*EPS_xx+ valA*EPS_yy + valA*EPS_zz;
                    SIGMA_N[3*ip-2]+= valA*EPS_xx+ valX*EPS_yy + valA*EPS_zz;
                    SIGMA_N[3*ip-1]+= valA*EPS_xx+ valA*EPS_yy + valX*EPS_zz;
                    TAU_N[3*ip-3]+= GAM_xy*valB;
                    TAU_N[3*ip-2]+= GAM_xz*valB;
                    TAU_N[3*ip-1]+= GAM_yz*valB;
                    if (ip==jp) B[ip-1]+=VOL;
                }
            }
        }

/****
      NODAL     VALUE
***/
      for(i=0;i<N;i++){
           RB=1.0e0/B[i];
           SIGMA_N[3*i  ]*=RB;
           SIGMA_N[3*i+1]*=RB;
           SIGMA_N[3*i+2]*=RB;
           TAU_N[3*i ]*=RB;
           TAU_N[3*i+1]*=RB;
           TAU_N[3*i+2]*=RB;}
/****
          UPDATE
***/
          WS=(KREAL*)allocate_vector(sizeof(KREAL),3*NP);
          WR=(KREAL*)allocate_vector(sizeof(KREAL),3*NP);
          SOLVER_SEND_RECV_3( NP,NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_ITEM,
                                    EXPORT_INDEX,EXPORT_ITEM, WS,WR,SIGMA_N, my_rank);
          SOLVER_SEND_RECV_3( NP,NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_ITEM,
                                    EXPORT_INDEX,EXPORT_ITEM, WS,WR,TAU_N, my_rank);

          deallocate_vector((KREAL*)WS);
          deallocate_vector((KREAL*)WR);
}
```

$$u_{,x} = [N_{,x}]\{U\}, \quad u_{,y} = [N_{,y}]\{U\}, \quad u_{,z} = [N_{,z}]\{U\}$$
$$v_{,x} = [N_{,x}]\{V\}, \quad v_{,y} = [N_{,y}]\{V\}$$
$$w_{,x} = [N_{,x}]\{W\}, \quad w_{,z} = [N_{,z}]\{W\}$$

RECOVER_STRESS：応力（4/4）

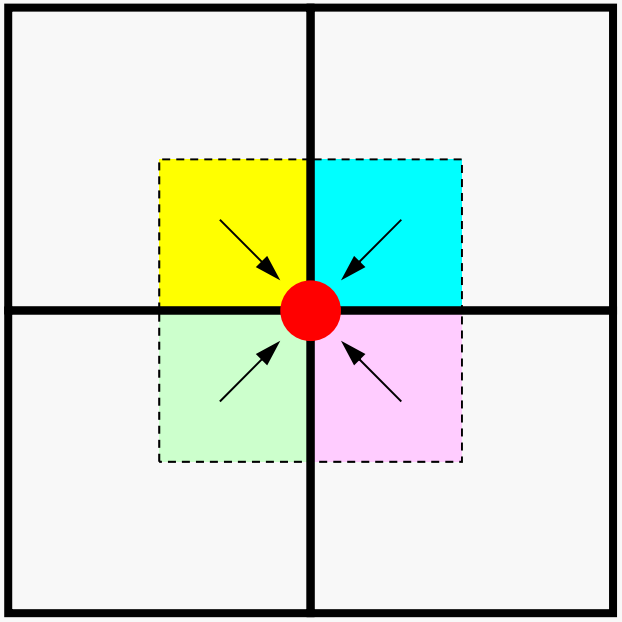
```
                    EPS_xx= EPS_xx * UUj;
                    EPS_yy= EPS_yy * VVj;
                    EPS_zz= EPS_zz * WWj;

                    SIGMA_N[3*ip-3]+= valX*EPS_xx+ valA*EPS_yy + valA*EPS_zz;
                    SIGMA_N[3*ip-2]+= valA*EPS_xx+ valX*EPS_yy + valA*EPS_zz;
                    SIGMA_N[3*ip-1]+= valA*EPS_xx+ valA*EPS_yy + valX*EPS_zz;
                    TAU_N[3*ip-3]+= GAM_xy*valB;
                    TAU_N[3*ip-2]+= GAM_xz*valB;
                    TAU_N[3*ip-1]+= GAM_yz*valB;
                    if (ip==jp) B[ip-1]+=VOL;
                }
            }
        }

/****
        NODAL     VALUE
***/
        for(i=0;i<N;i++){
            RB=1.0e0/B[i];
            SIGMA_N[3*I  ]*=RB;
            SIGMA_N[3*i+1]*=RB;
            SIGMA_N[3*i+2]*=RB;
            TAU_N[3*i ]*=RB;
            TAU_N[3*i+1]*=RB;
            TAU_N[3*i+2]*=RB;}
/****
            UPDATE
***/
            WS=(KREAL*)allocate_vector(sizeof(KREAL),3*NP);
            WR=(KREAL*)allocate_vector(sizeof(KREAL),3*NP);
            SOLVER_SEND_RECV_3( NP,NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_ITEM,
                                EXPORT_INDEX,EXPORT_ITEM, WS,WR,SIGMA_N, my_rank);
            SOLVER_SEND_RECV_3( NP,NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_ITEM,
                                EXPORT_INDEX,EXPORT_ITEM, WS,WR,TAU_N, my_rank);

            deallocate_vector((KREAL*)WS);
            deallocate_vector((KREAL*)WR);
}
```

$$\begin{Bmatrix}\sigma_x\\ \sigma_y\\ \sigma_z\\ \tau_{xy}\\ \tau_{yz}\\ \tau_{zx}\end{Bmatrix} = \begin{bmatrix}\mathrm{valX} & \mathrm{valA} & \mathrm{valA} & 0 & 0 & 0\\ \mathrm{valA} & \mathrm{valX} & \mathrm{valA} & 0 & 0 & 0\\ \mathrm{valA} & \mathrm{valA} & \mathrm{valX} & 0 & 0 & 0\\ 0 & 0 & 0 & \mathrm{valB} & 0 & 0\\ 0 & 0 & 0 & 0 & \mathrm{valB} & 0\\ 0 & 0 & 0 & 0 & 0 & \mathrm{valB}\end{bmatrix} \begin{Bmatrix}\varepsilon_x\\ \varepsilon_y\\ \varepsilon_z\\ \gamma_{xy}\\ \gamma_{yz}\\ \gamma_{zx}\end{Bmatrix}$$

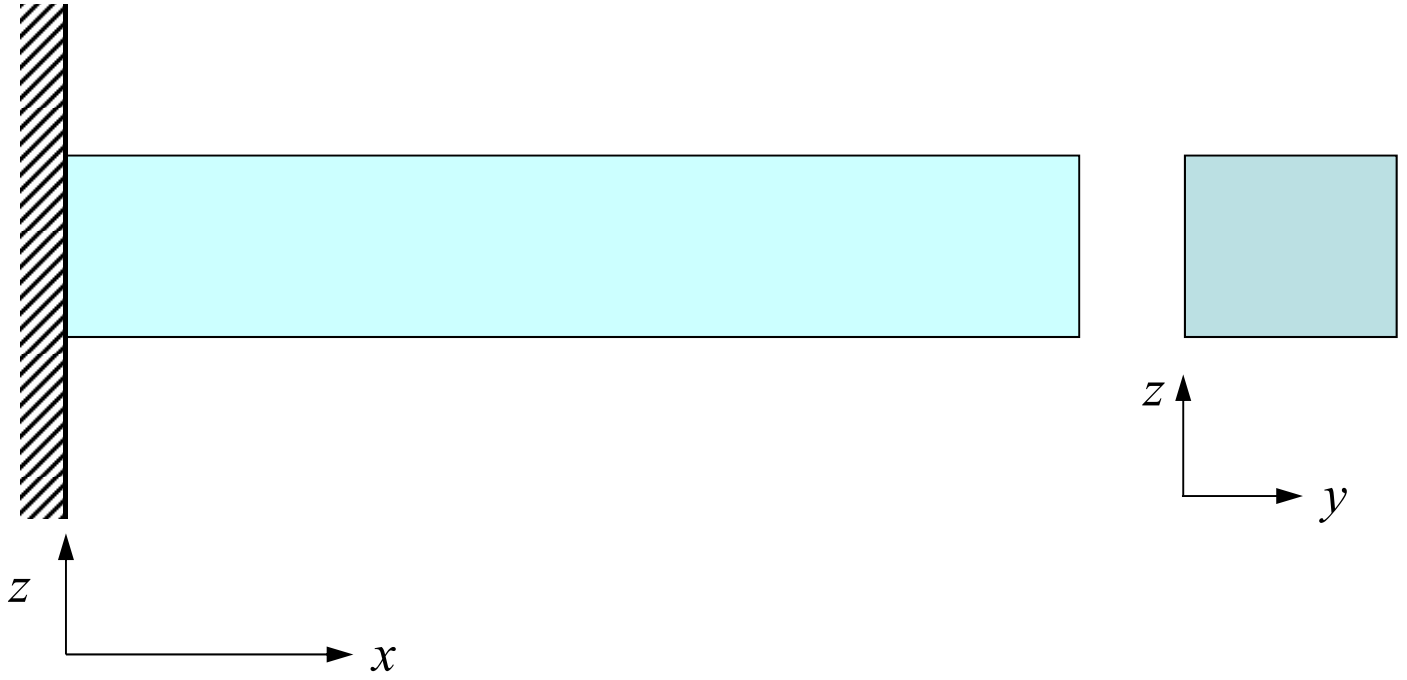
RECOVER_STRESS：応力（4/4）

```
                    EPS_xx= EPS_xx * UUj;
                    EPS_yy= EPS_yy * VVj;
                    EPS_zz= EPS_zz * WWj;

                    SIGMA_N[3*ip-3]+= valX*EPS_xx+ valA*EPS_yy + valA*EPS_zz;
                    SIGMA_N[3*ip-2]+= valA*EPS_xx+ valX*EPS_yy + valA*EPS_zz;
                    SIGMA_N[3*ip-1]+= valA*EPS_xx+ valA*EPS_yy + valX*EPS_zz;
                    TAU_N[3*ip-3]+= GAM_xy*valB;
                    TAU_N[3*ip-2]+= GAM_xz*valB;
                    TAU_N[3*ip-1]+= GAM_yz*valB;
                    if (ip==jp) B[ip-1]+=VOL;
                }
            }
        }

/****
      NODAL     VALUE
***/
      for(i=0;i<N;i++){
           RB=1.0e0/B[i];
           SIGMA_N[3*I  ]*=RB;
           SIGMA_N[3*i+1]*=RB;
           SIGMA_N[3*i+2]*=RB;
           TAU_N[3*i ]*=RB;
           TAU_N[3*i+1]*=RB;
           TAU_N[3*i+2]*=RB;}
/****
          UPDATE
***/
          WS=(KREAL*)allocate_vector(sizeof(KREAL),3*NP);
          WR=(KREAL*)allocate_vector(sizeof(KREAL),3*NP);
          SOLVER_SEND_RECV_3( NP,NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_ITEM,
                                    EXPORT_INDEX,EXPORT_ITEM, WS,WR, SIGMA_N, my_rank);
          SOLVER_SEND_RECV_3( NP,NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_ITEM,
                                    EXPORT_INDEX,EXPORT_ITEM, WS,WR, TAU_N, my_rank);

          deallocate_vector((KREAL*)WS);
          deallocate_vector((KREAL*)WR);
}
```

内点のみについて計算



レポート課題S3（1/2）



- 以下の問題を解く三次元弾性解析コードを作成し，計算を実施せよ
  - 片持ち梁（断面：長方形）
    - ヤング率=1.00，自重（密度）=0.025
    - ポアソン比=0.30
  - u=v=w=0@x=0

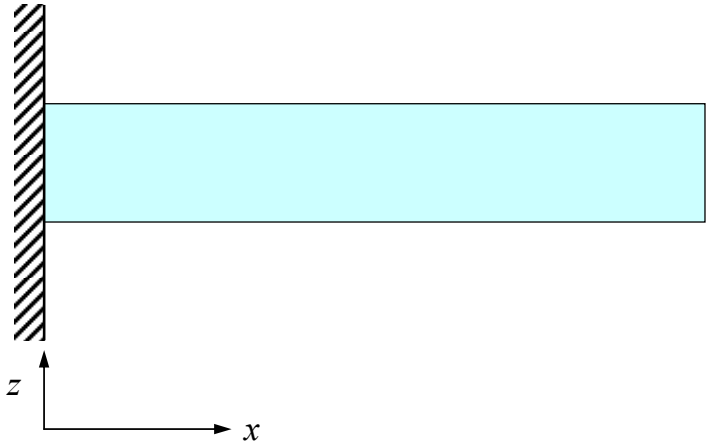
レポート課題S3（2/2）

- メッシュジェネレータの作成
- fem3d（並列版）の改造（境界条件，自重）
- 検証（解析解：次ページ），メッシュ分割の影響
- ガウス積分点の数を変えてやってみよ
  - n=2（fem3d），n=1，n=3
- レポート
  - 方針（基本設計），定式化，実装，結果，考察
  - 関数OUTPUT_UCDの処理，問題点について説明せよ
  - A4　10枚以内（＋リスト）
  - 提出：2013年2月22日（金）17:00（メール添付可）

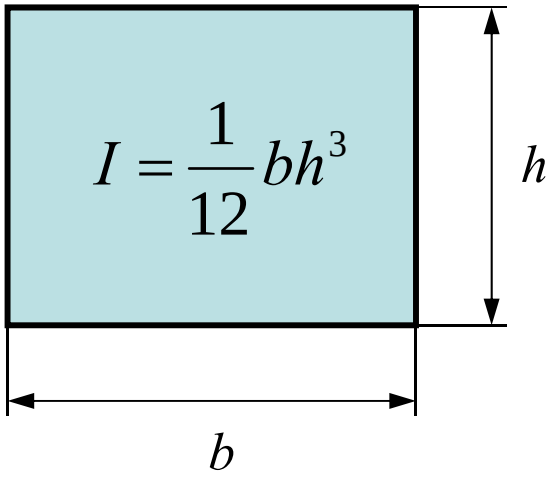
解析解：Lが充分長いときに成立

$$z = -\frac{W}{24EI}\left(x-L\right)^2 \cdot \left(x^2+2Lx+3L^2\right)$$

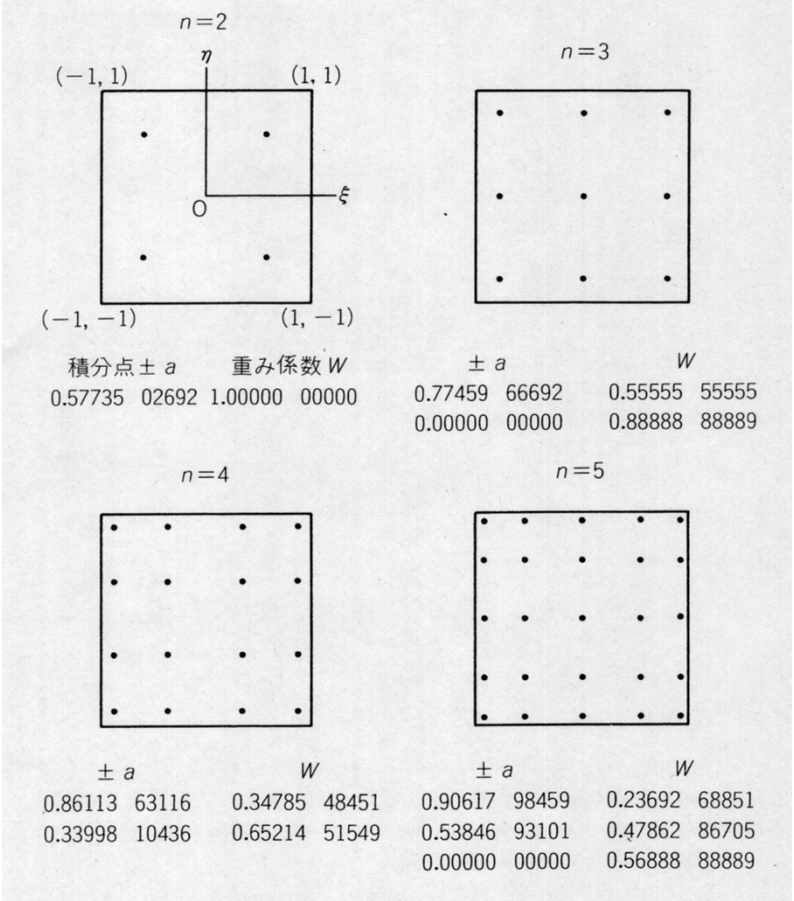
$$z_{\max} = -\frac{WL^4}{8EI} \quad where \quad \frac{L}{h} > 4$$



- z　z方向変位
- L　梁長さ
- W　長さあたり自重
- E　ヤング率
- I　断面二次モーメント
  （長方形の場合）



ガウスの積分公式



n=1

積分点座標：0.000
重み係数　：1.000
（要素中心）

自重：Z方向体積力

$$\frac{\partial \tau_{zx}}{\partial x}+\frac{\partial \tau_{xy}}{\partial y}+\frac{\partial \sigma_{z}}{\partial z}+Z=0$$

$$\tau_{zx,x}+\tau_{xy,y}+\sigma_{z,z}+Z=0$$

↓ ガラーキン法

$$\int [N]^T \{\tau_{zx,x}+\tau_{xy,y}+\sigma_{z,z}+Z\} dV = 0$$

$$\int [N]^T \{\tau_{zx,x}\} dV + \int [N]^T \{\tau_{xy,y}\} dV + \int [N]^T \{\sigma_{z,z}\} dV + \int [N]^T \{Z\} dV = 0$$