OpenFOAMによる
メッシュ操作入門

2014年5月10日

オープンCAE勉強会＠富山

富山県立大学　中川慎二

内容

blockMesh (簡単に)

- 基礎知識
- 基礎実習 : 単ブロック
- 応用実習
  - 複数ブロック
  - edgesを使って曲線
  - 格子の引き寄せ
  - 拡張版smoothing

snappyHexMesh

- 基礎知識
- 基礎実習 : 例題実行

Salome (紹介のみ)

- 基本形状の作成，STL エキスポート，snappyHexMeshでの利用
- Salomeメッシュのインポート

様々なメッシュ生成ソフト･方法

OpenFOAM

- blockMesh ユーティリティー
- snappyHexMesh ユーティリティー
- foamyHexMesh ユーティリティー ← 最新：要熟成？

その他のオープンソースソフト

- Salome-meca
- DEXCS，Engrid, gmesh など

商用ソフト

- CUBIT，Pointwise，など
- 商用ソルバのプリ機能

blockMesh 解説

blockMesh ユーティリティ

- 最も基本となるメッシュ生成方法
- 点，線，面，ブロックを，設定ファイル（blockMeshDict）に記述する
- こまかな制御が可能
- 設定ファイルの作成に，手間がかかる

エラーを出さないために

- 設計図をしっかりと描く！
- ブロック作成時に，点の順番を意識する！
  - (1) x座標が増える，(2)y座標が増える，(3)z座標が増える
- Dict を見やすく書く！
- 括弧 () の前には，空白を入れる.

blockMeshDict の使い方

- blockMeshDict
  - 数字を直接書き込む（変数も使用可）
    - 基本
    - 形状変更時に手間がかかる
  - m4 を利用して，汎用化
    - マクロ言語プロセッサ m4 を利用
    - blockMeshDict を生成するためのファイルを作成
    - 形状変更等が容易になる
  - Dictionary にコード（プログラム）を書いて，汎用化
    - 実行時コード実行の許可が必要
    - 形状変更等が容易になる

blockMeshDict の全体構造（基本）

```
/*--------------------------------*- C++ -*----------------------------------*¥
| =========                 |                                                 |
| ¥¥      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  ¥¥    /   O peration     | Version:  2.1.0                                 |
|   ¥¥  /    A nd           | Web:      www.OpenFOAM.org                      |
|    ¥¥/     M anipulation  |                                                 |
¥*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
convertToMeters 1;

vertices   (               );

blocks     (               );

edges      (               );

boundary (               );
// ************************************************************************* //
```

blockMeshDict: vertices (節点)

```
vertices
(
        (0 0 0)     // 0, 1コ目の節点
        (1 0 0)     // 1, 2コ目の節点
);
```

C++のソースコードと同様に，記号「//」を付けるとコメントになる。その後ろは，ただのメモ書き．

後で楽をするために，自分で番号を書いておくと良い。急がば回れ・・・

blockMeshDict: blocks(ブロック)

Hexahedra（六面体）を指定する

```
blocks
(
    hex
    (0 1 2 3 4 5 6 7)
    (20 8 8)      // 各方向(ローカル座標)のセル数
    simpleGrading (1 1 1)   // セルの拡大率
);
```

六面体の頂点となる節点を指定する

各辺の拡大率を指定することもできる
edgeGrading として　*Fig.5.4の辺番号順に指定*

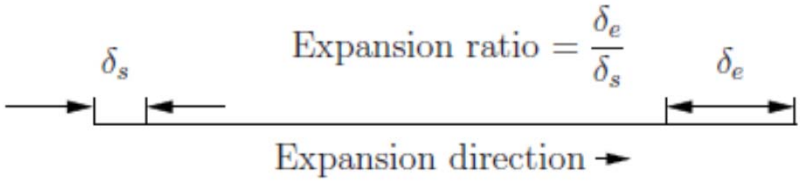


Figure 5.5: Mesh grading along a block edge

blockMeshDict: blocks(ブロック)

一番小さい座標の点を選ぶ

0番から，x1方向に進んだ点を書く

1番から，x2方向に進んだ点を書く

Z座標の小さい面で残った点

0番から，x3方向に進んだ点を書く

x1，x2，x3方向の分割数

hex (0 1 2 3 4 5 6 7) (20 8 8)

Z座標の大きい面の4点

Z座標の小さい面の4点

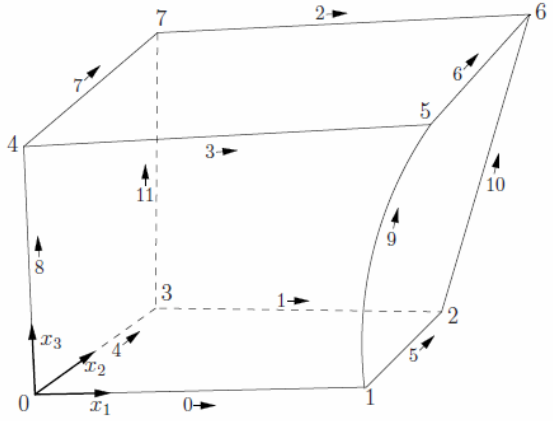
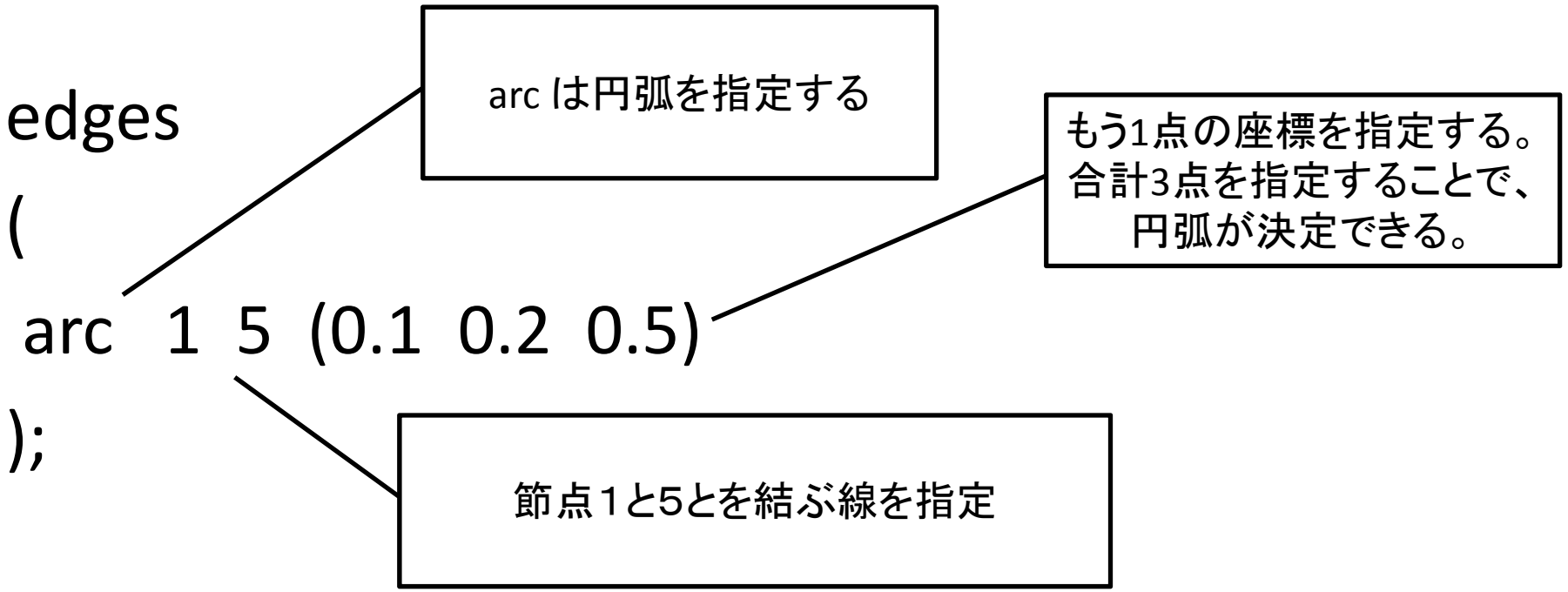


Figure 5.4: A single block

※ローカル座標 (x1, x2, x3) とグローバル座標 (x, y, z) を一致させると間違いが少ない

各辺の拡大率を指定することもできる
edgeGrading として　*Fig.5.4の辺番号順に指定*

blockMeshDict: edges (線)



2つの節点間を結ぶ線の種類を指定できる。

指定をしなければ、直線で結ぶ。

edges (線) の種類

指定するキーワード	説明	追加で指定する情報
arc	円弧	途中の1点
spline	スプライン曲線	途中の点のリスト
polyLine	多角線	途中の点のリスト
line	直線	

blockMeshDict: boundary (境界面)

```
boundary
(
 name
   {
      type patch;
      faces
      (
         (3 7 6 2) // 4個の節点で面を指定する
         (1 5 4 0)
      );
   }
);
```

名前：任意に付けて良い。ただし，他のファイルの情報（boudary, U, p など）と一致させること。

patchのタイプ：境界条件に応じた型を与える。

境界条件については下記を参照のこと。
http://foam.sourceforge.net/docs/cpp/a00001.html

blockMeshDict をGUIでチェック

- blockMeshDictをGUIでチェックする。
- pyFoamの機能 pyFoamDisplayBlockMesh を利用する。

- ケースディレクトリから下記コマンドを実行する

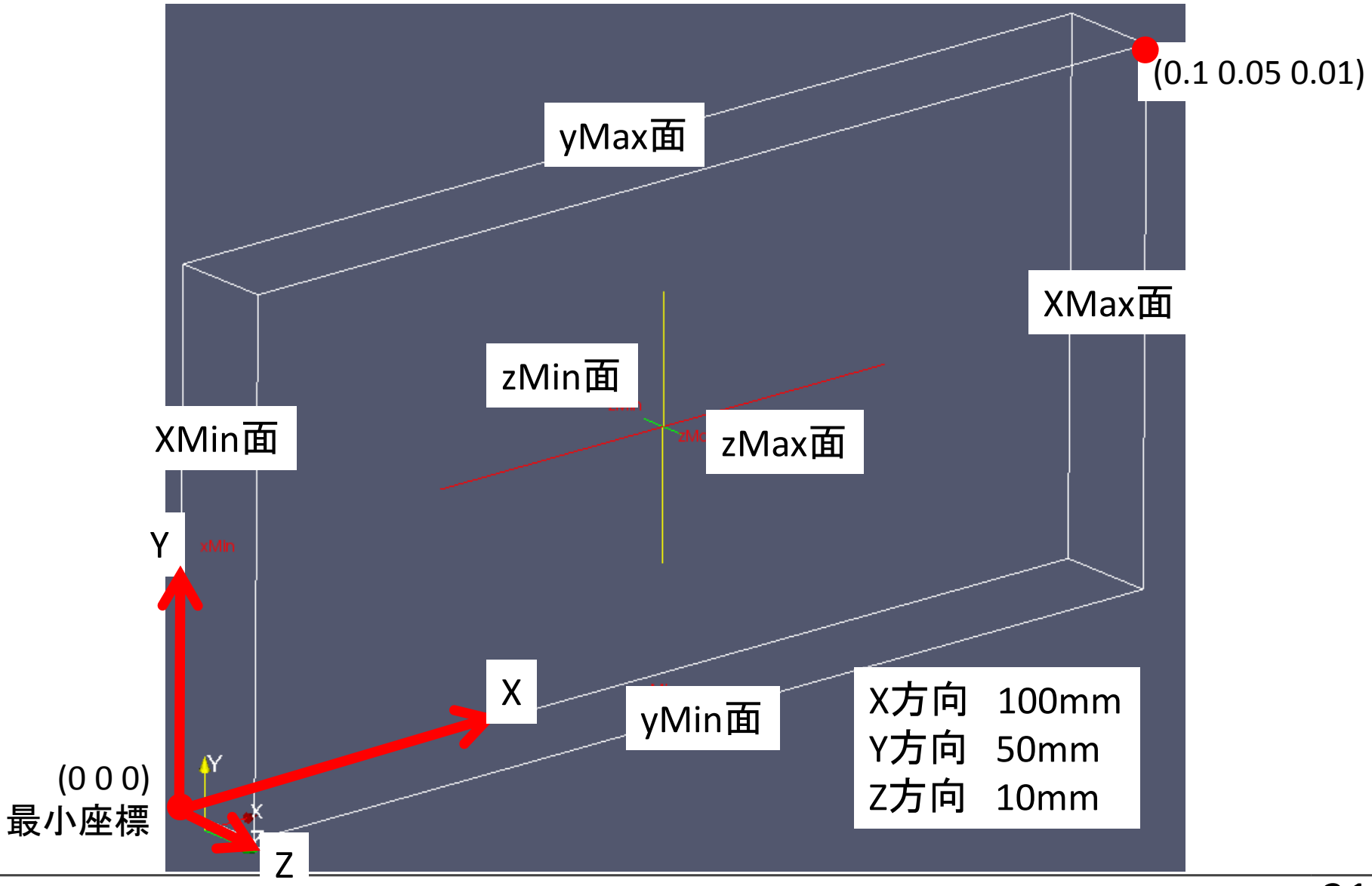
pyFoamDisplayBlockMesh.py constant/polyMesh/blockMeshDict

blockMesh 実習1
基本

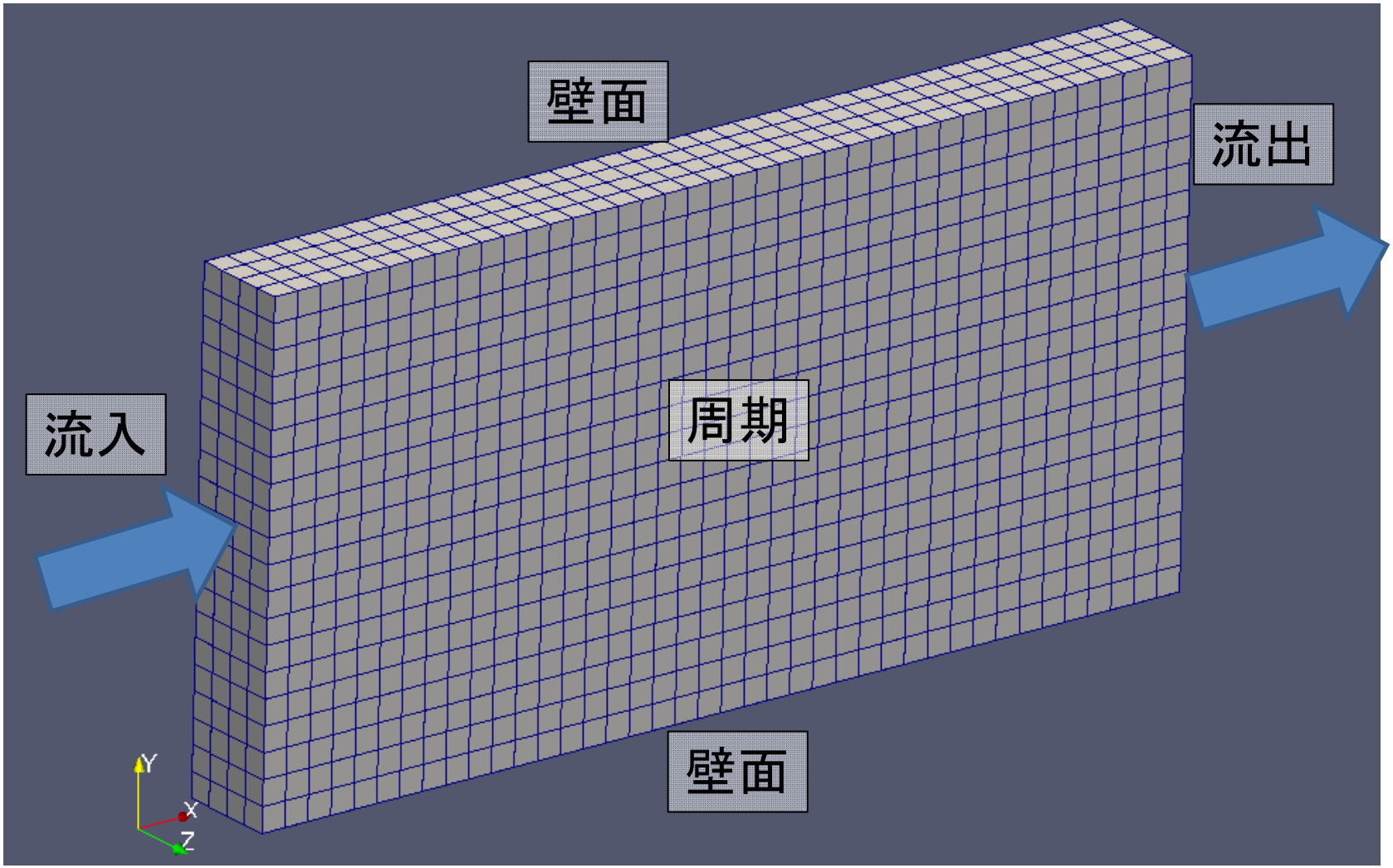
題材

- 直方体（1ブロック）
  - 例題ケース　bmTest01
  - このケースを基本として、下記ケースを作成
- 角柱を置いた流路（複数ブロック）
  - 上記直方体から，障害物を除いた領域
  - 例題ケース　bmTest03
- 角柱の一部を曲線にした場合
  - edge の利用
  - 例題ケース　bmTest05

計算領域


(0.1 0.05 0.01)
yMax面
XMax面
zMin面
zMax面
XMin面
Y
X
Z
yMin面
(0 0 0)
最小座標
X方向　100mm
Y方向　50mm
Z方向　10mm

テスト ケース1 : bmTest01



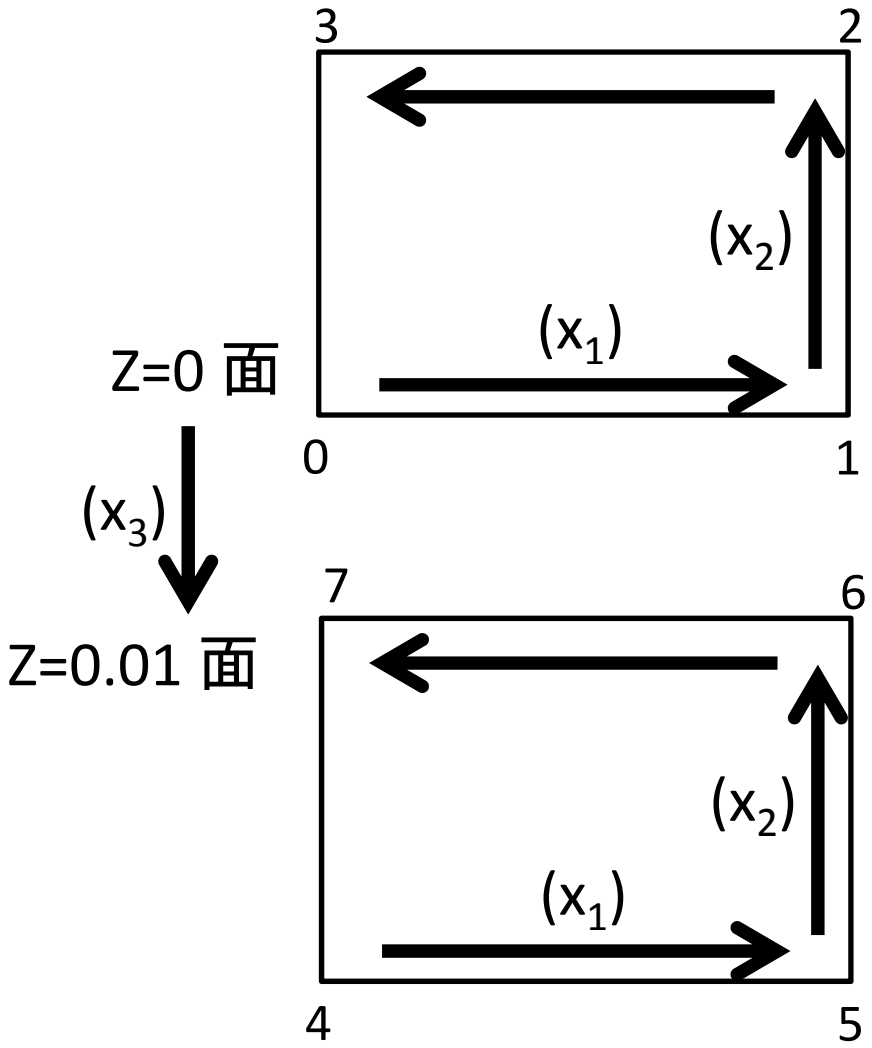
blockMesh利用 メッシュ生成手順

- ブロック構造を決める
  - 境界条件をどこに設定するか？
  - 面を分ける必要があるか？
  - 何個のブロックを使うか？
- 節点の座標を決める
- ブロックを構成する点の並び順を決める
- 面を構成する点の並び順を決める

bmTest01: blockMeshDict前半

```
convertToMeters 1;

vertices
(
    (0.0  0.0   0.0)  // 0    zMin plane (Z=0)
    (0.1  0.0   0.0)  // 1
    (0.1  0.05 0.0)  // 2
    (0.0  0.05 0.0)  // 3

    (0.0  0.0   0.01) // 4   zMax plane (Z=0.01)
    (0.1  0.0   0.01) // 5
    (0.1  0.05 0.01) // 6
    (0.0  0.05 0.01) // 7
);

blocks
(
    hex (0 1 2 3  4 5 6 7) (40     20     4)
    simpleGrading (1 1 1)
);

edges (   );
```

x_1 x_2 x_3



bmTest01: blockMeshDict後半

```
boundary
(
    xMin
    {
        type patch;
        faces
        (
            (0 4 7 3)
        );
    }
    xMax
    {
        type patch;
        faces
        (
            (1 2 6 5)
        );
    }
    yMin
    {
        type wall;
        faces
        (
            (0 1 5 4)
        );
    }
    yMax
    {
        type wall;
        faces
        (
            (7 6 2 3)
        );
    }
    zMin
    {
        type cyclic;
        neighbourPatch zMax;
        faces
        (
            (3 2 1 0)
        );
    }
    zMax
    {
        type cyclic;
        neighbourPatch zMin;
        faces
        (
            (4 5 6 7)
        );
    }
);
```

作業 (bmTest01例題)

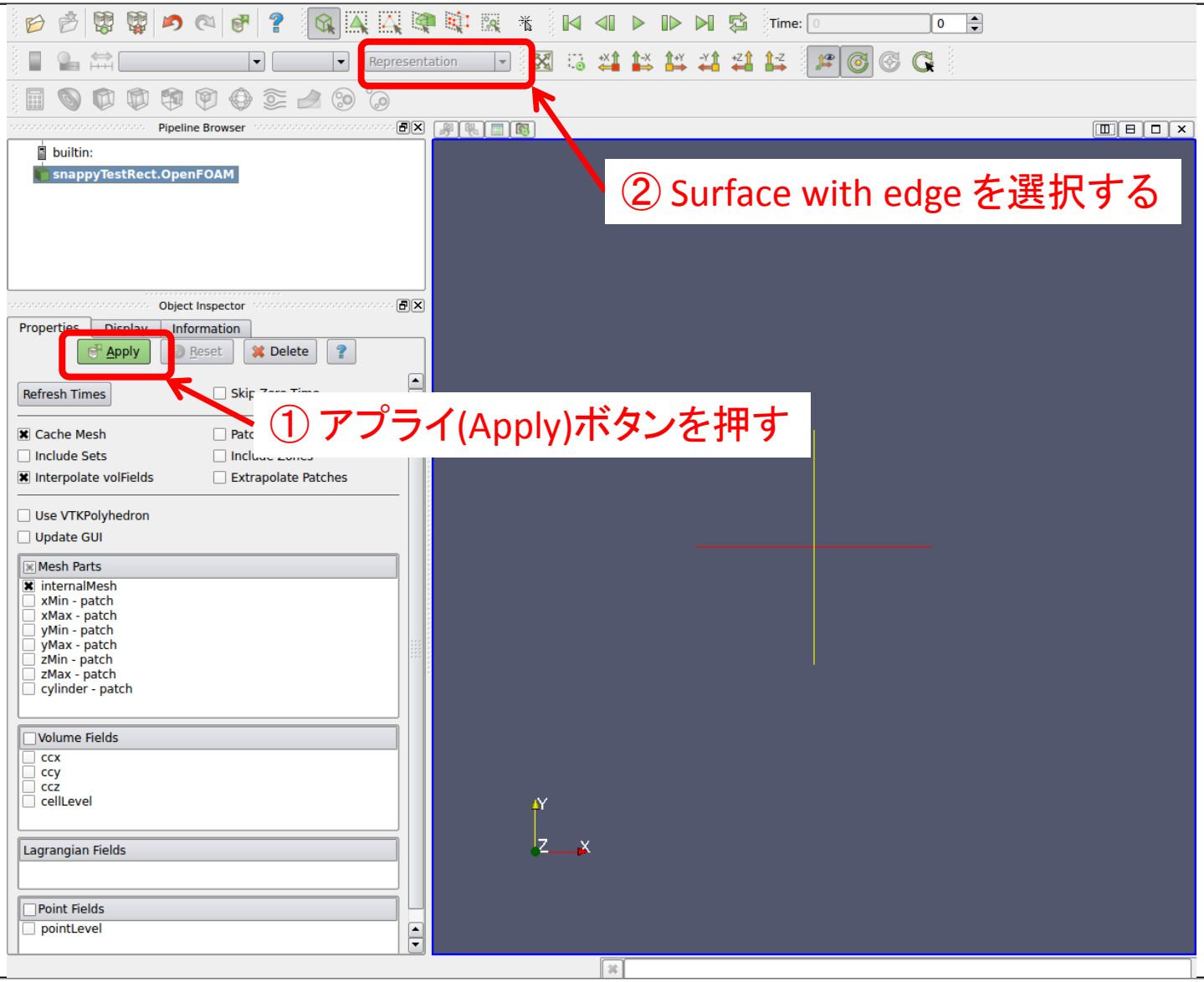
- ファイルマネージャーを起動し、例題ケースディレクトリを右クリックして、「端末で開く」を選択する
- （端末を起動する → 例題ケースディレクトリへ移動する cd Desktop/meshTraining/bmTest01)
- blockMesh ユーティリティを実行する

blockMesh

- エラーが出力されないことを確認する
- paraFoamを実行する

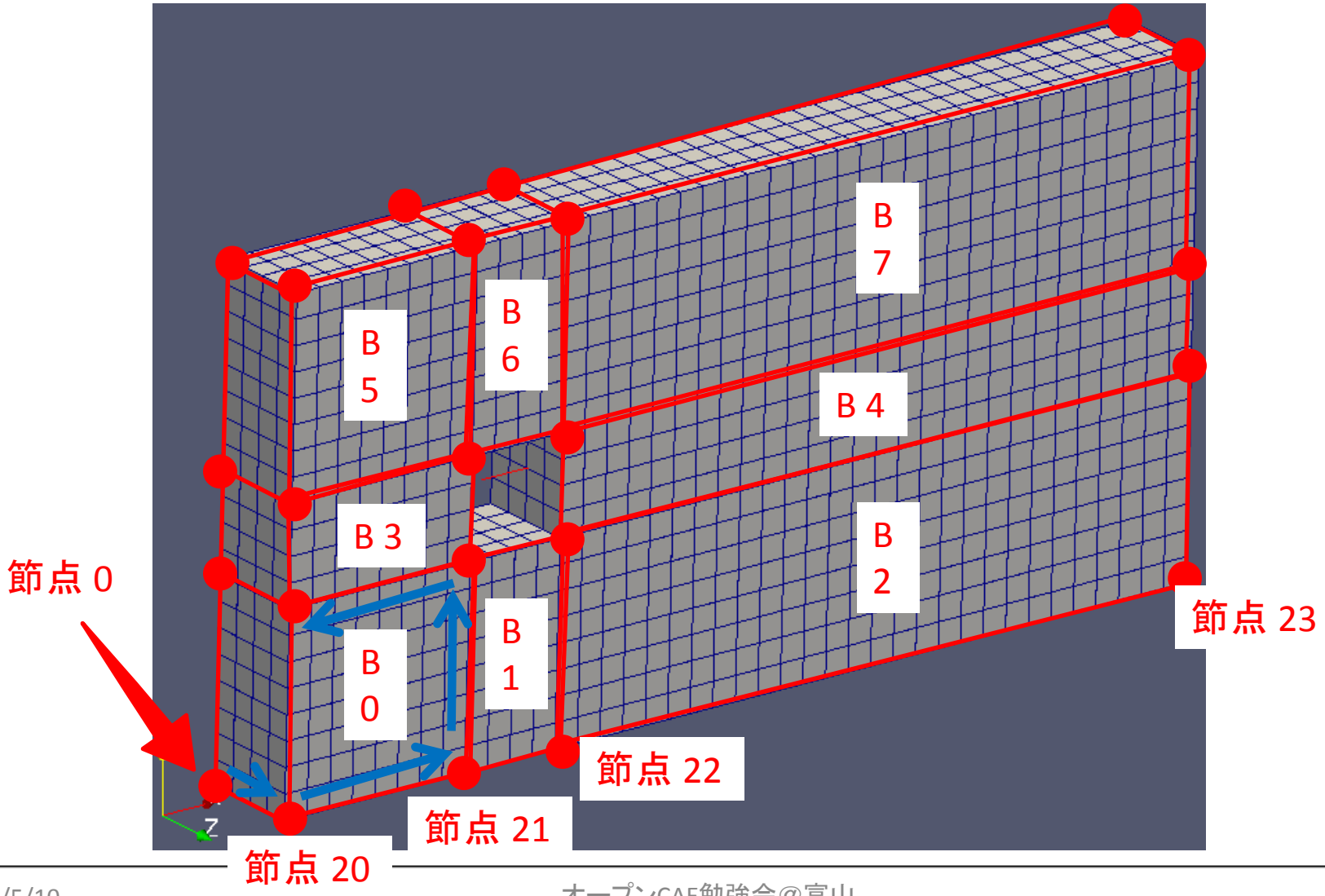
paraFoam

メッシュの確認：paraview



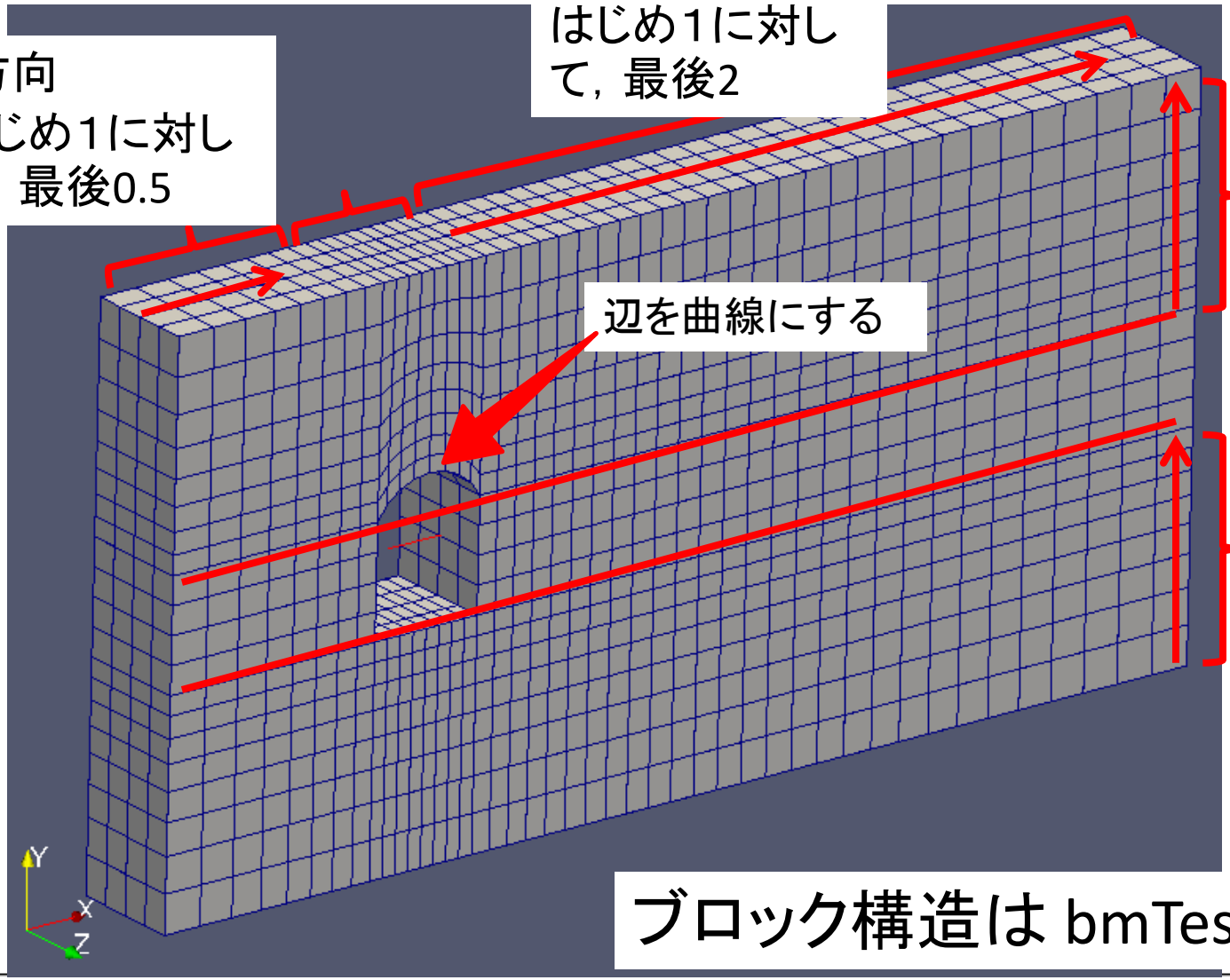
発展：テスト ケース bmTest03

ケース有り。
終了後に見てください。



発展：テスト ケース bmTest05

ケース有り。
終了後に見てください。



ブロック構造は bmTest03 と同じ

発展：テスト ケース bmTest05

ケース有り。
終了後に見てください。

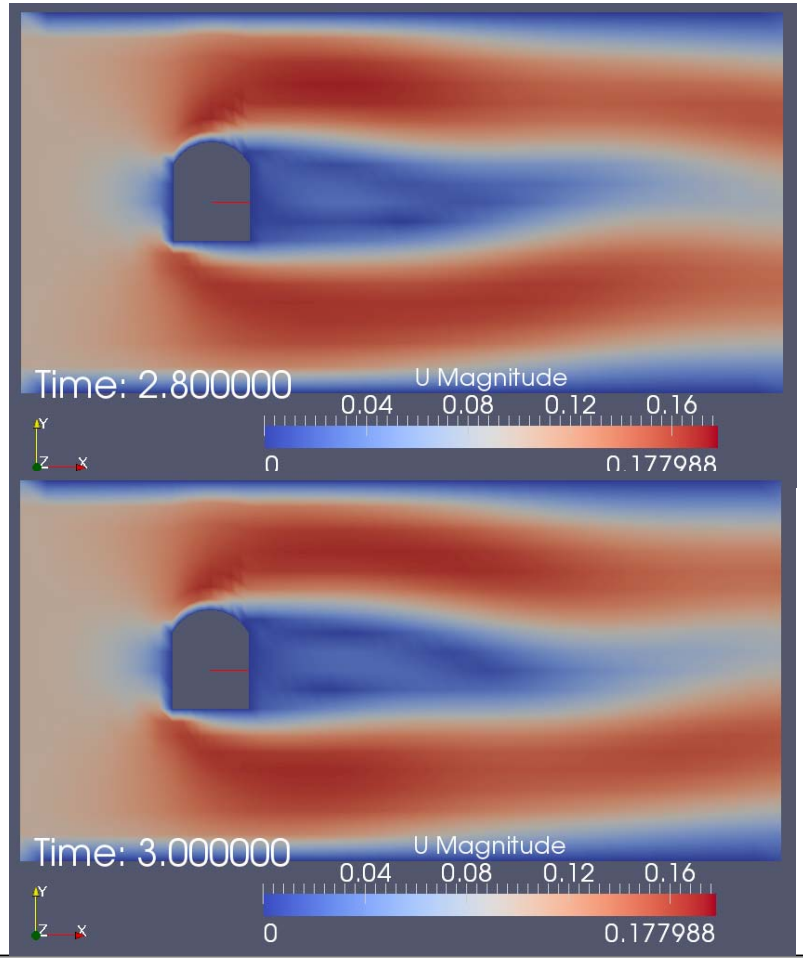
- 角柱の一部の形状を変える
- edge機能を使う

```
arc  9 10 (0.025 0.033 0)     //zMin plane
arc 29 30 (0.025 0.033 0.01)  //zMax plane
```

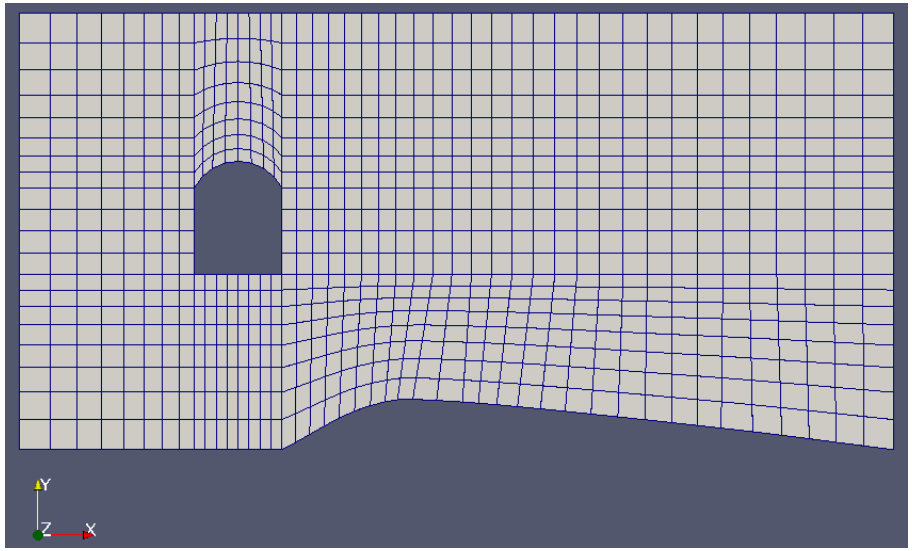
- セル数を少し増やす
- セル拡大率を調整し，角柱付近を細かくする

詳細は，blockMeshDict を参照

計算結果例

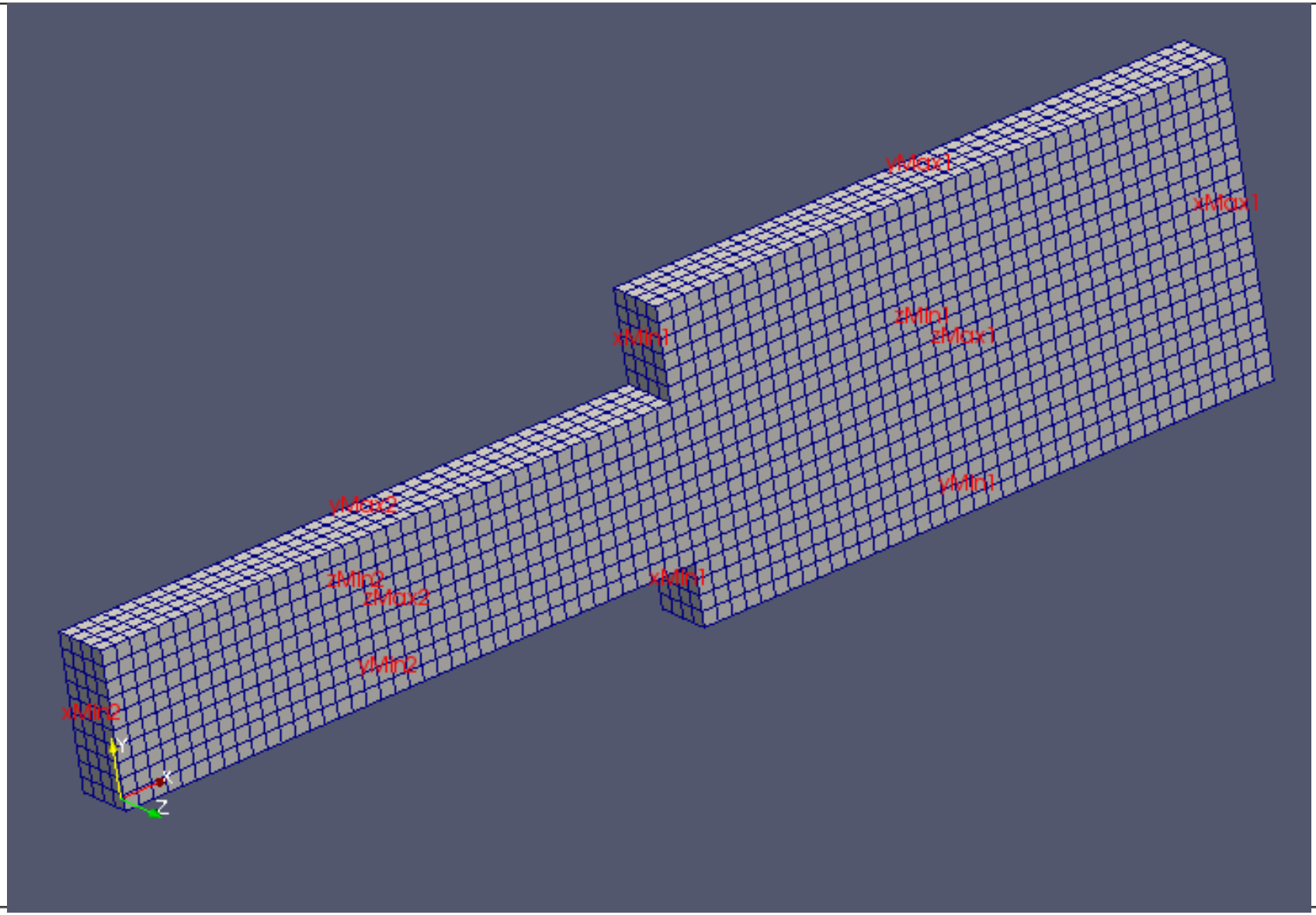


spline 使用例

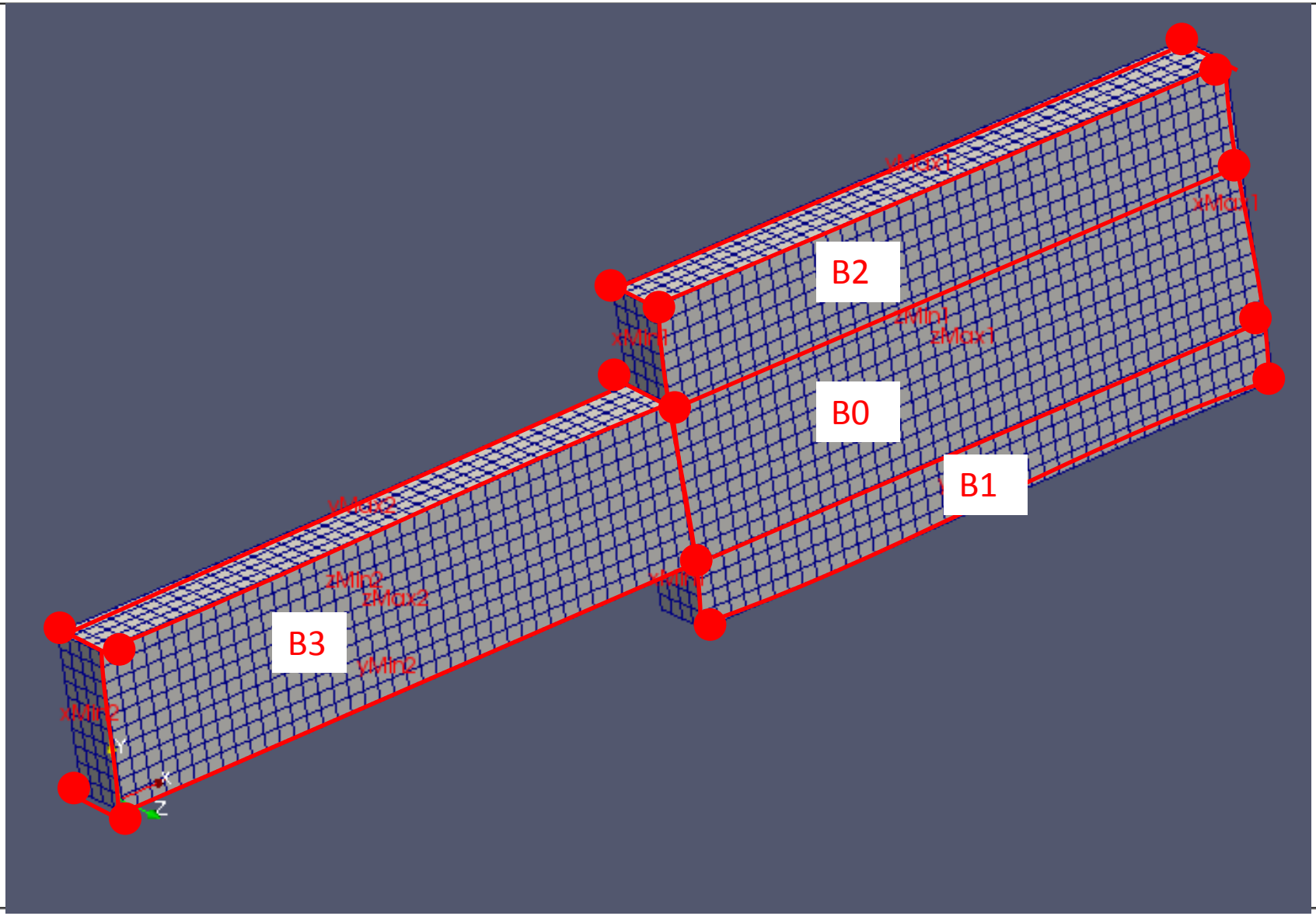


```
spline  2  3 (    (0.04 0.005 0)
                   (0.05 0.0055 0)
                   (0.08 0.0025 0)    )
spline 22 23 ( (0.04 0.005 0.01)
                   (0.05 0.0055 0.01)
                   (0.08 0.0025 0.01) )
```

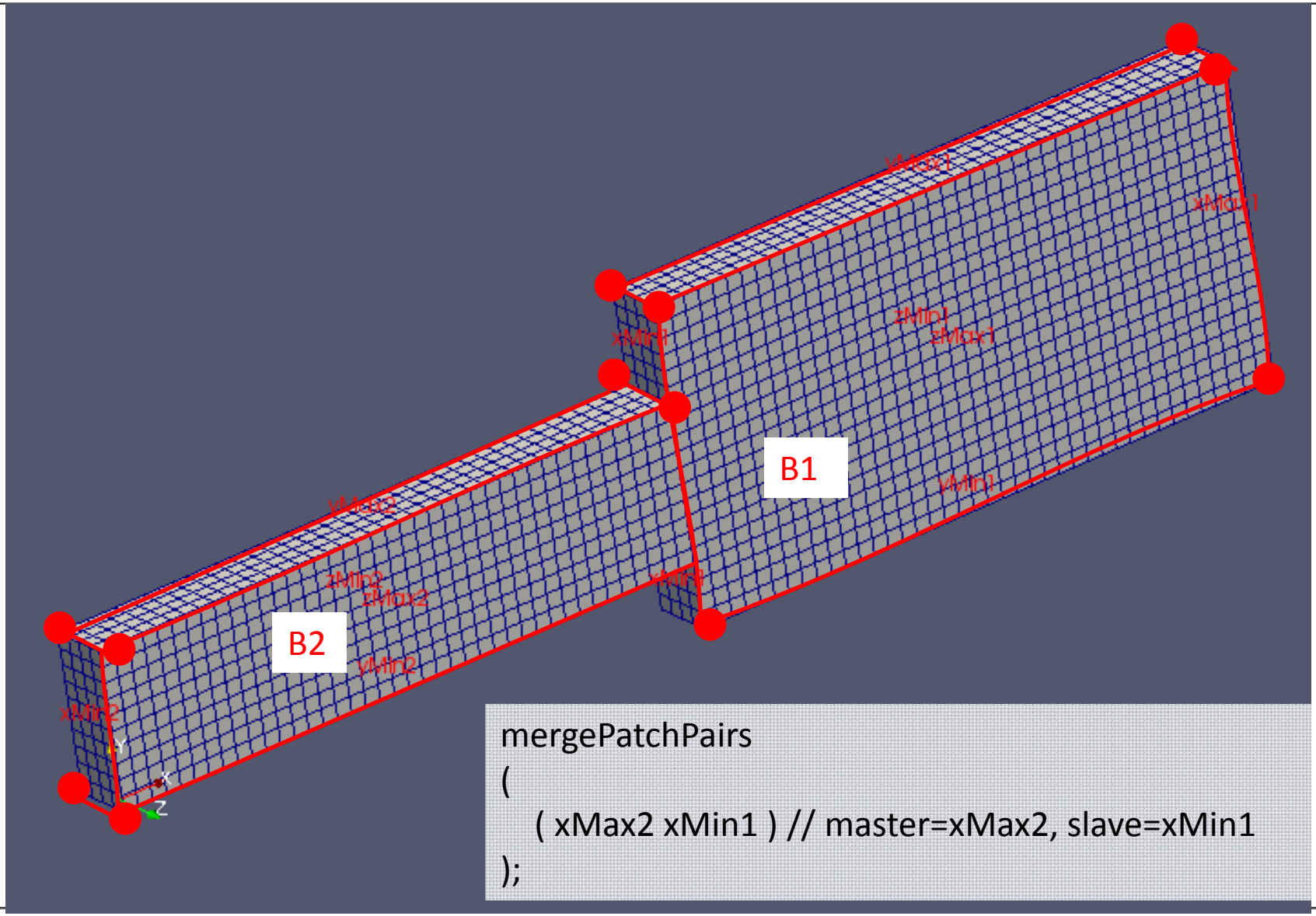
テスト ケース1 : bmTest06

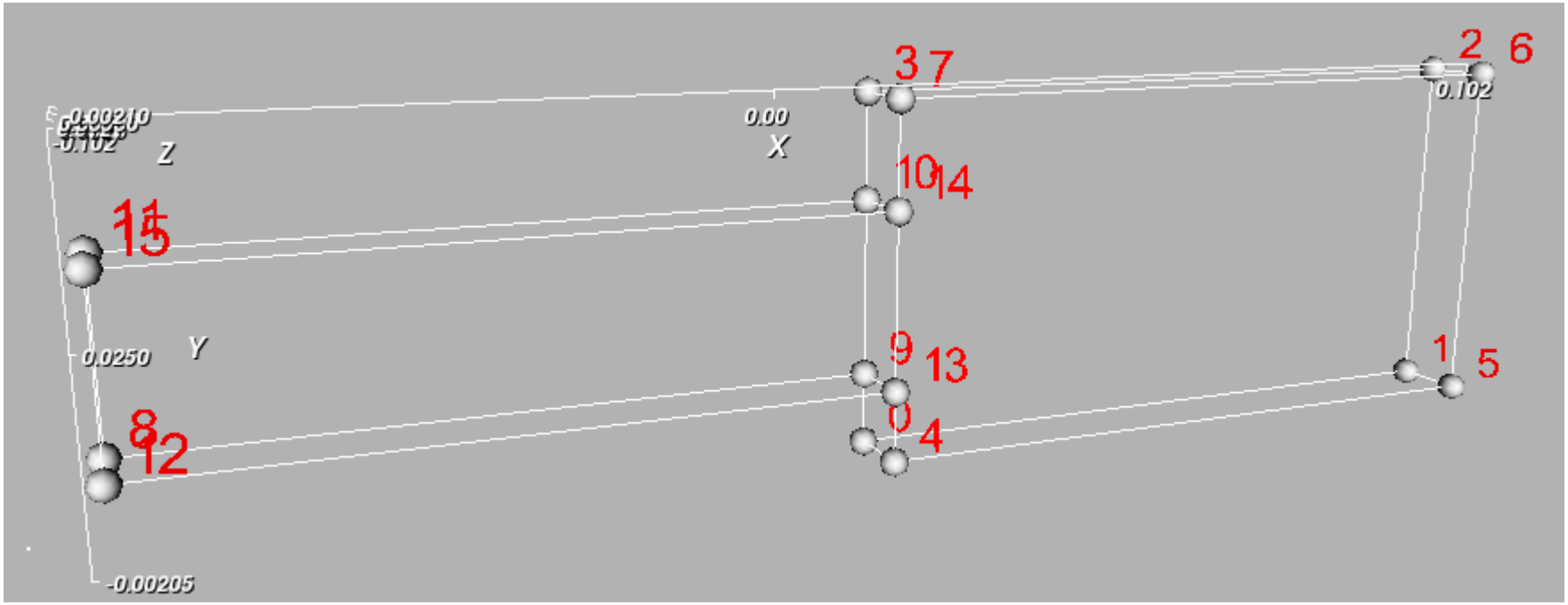


ブロック構成1



ブロック構成2：mergePatchPairs




0.00210
-0.102
Z
0.00
X
3 7
2 6
0.102
10 14
11
15
0.0250
Y
9 13
1 5
8
12
0 4
-0.00205

5.3.2 Multiple blocks

A mesh can be created using more than 1 block. In such circumstances, the mesh is created as has been described in the preceeding text; the only additional issue is the connection between blocks, in which there are two distinct possibilities:

- face matching
  - the set of faces that comprise a patch from one block are formed from the same set of vertices as a set of faces patch that comprise a patch from another block;
- face merging
  - a group of faces from a patch from one block are connected to another group of faces from a patch from another block, to create a new set of internal faces connecting the two blocks.

http://www.openfoam.org/docs/user/blockMesh.php

作業

- bmTest06/constant/polyMesh/blokMeshDictを開いて、mergePatchPairs の部分をコメントにする。（行頭に // を追加する。）
- blockMesh ユーティリティを実行する

blockMesh

- paraFoamを実行する

paraFoam

- 境界面が重複していることを確認する。
- mergePatchPairsを有効にして同様に実行。重複した部分がpatchではなくなることを確認する。

extBlockMesh

- ユーザが開発、gitHubで公開
- blockMeshをベースとして、メッシュ品質を向上させるためのsmoothing 処理を行う
- OpenFOAMとは別に、インストールが必要

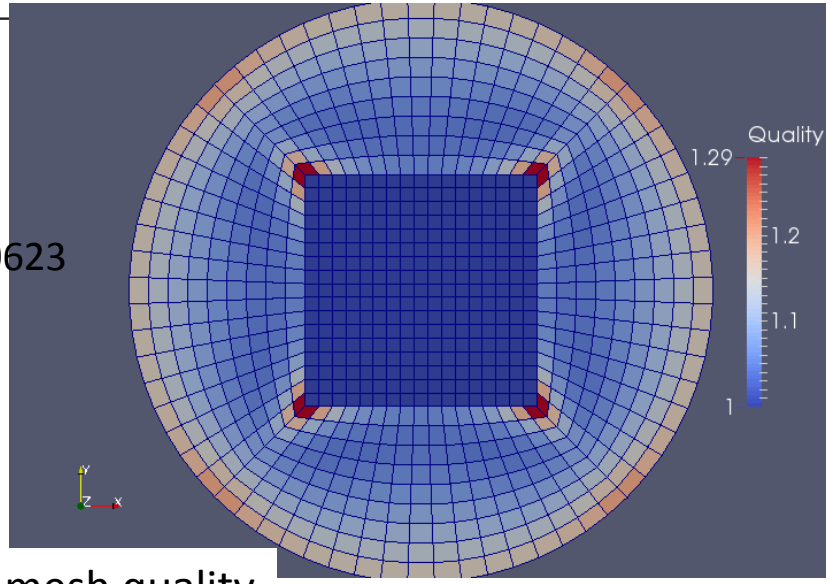
- blockMeshDictに、smooth処理用の設定を追加

http://www.etudes-ng.net/notre-savoir-faire/applications/mesh-smoothing

作業（bmTest07 例題）

- blockMesh
- checkMesh
- paraFoam

Mesh non-orthogonality
Max: 33.867 average: 6.80623

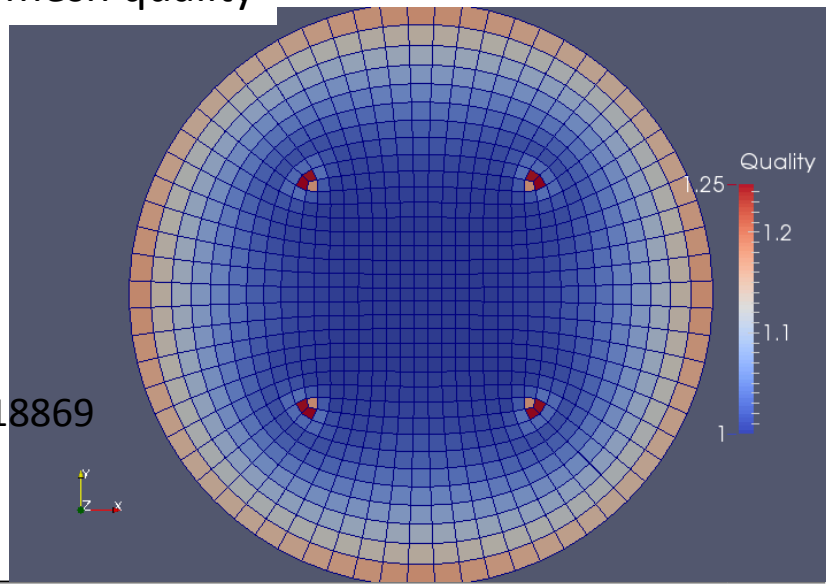


mesh quality

- ./Allclean

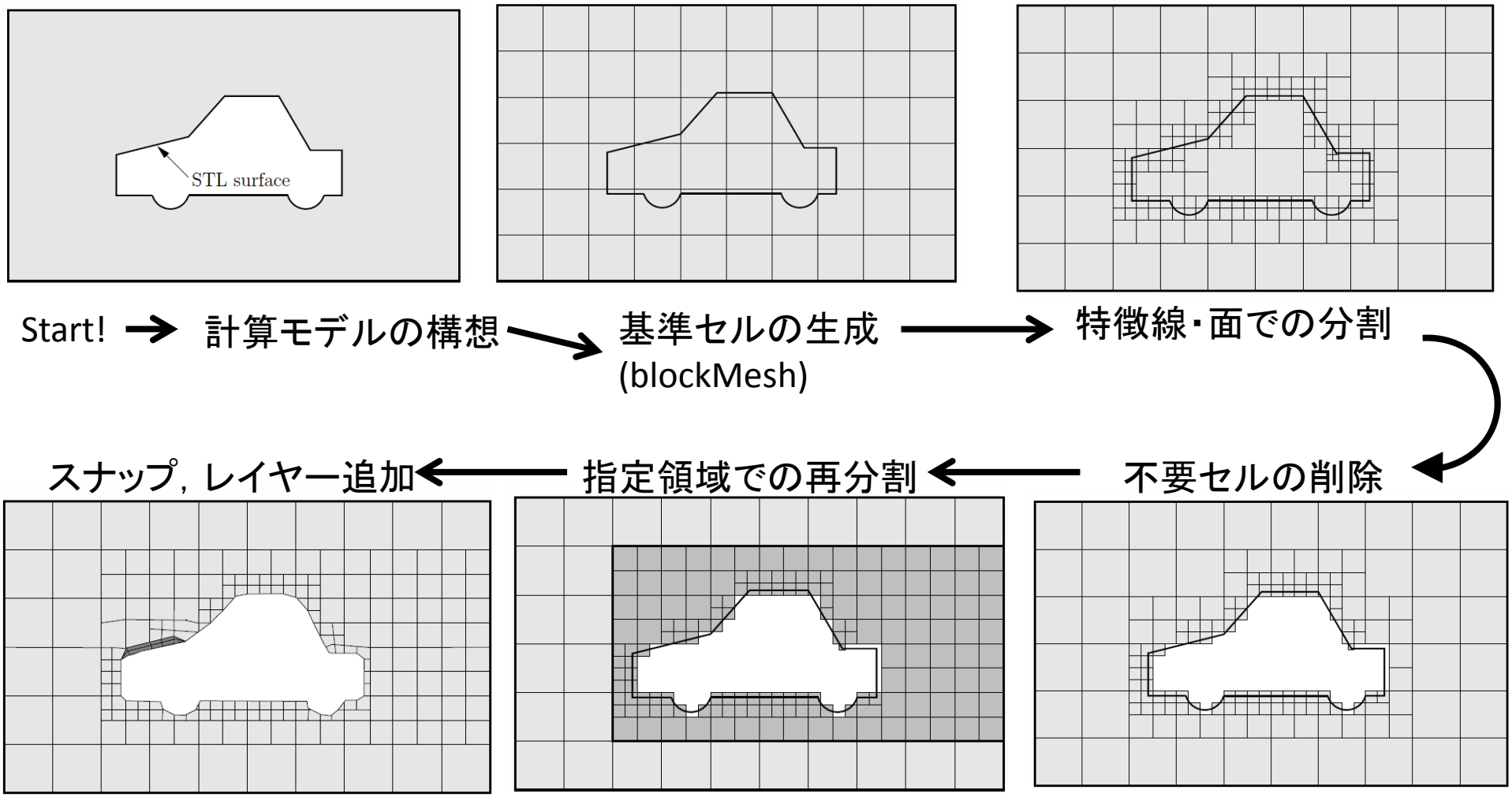
- extBlockMesh
- checkMesh
- paraFoam

Mesh non-orthogonality
Max: 8.13573 average: 2.18869



snappyHexMesh解説

どんなセルができていくか？



作業の流れ

- 計算領域の構造を決定
  - 領域の広さ;境界面の分け方(境界条件の設定)
  - 必要なセルの大きさ(場所によって異なる?)
  - 許容されるセル数
- 基準となるセル(レベル0)の作成
- 特徴線・面でのセル分割
- 不要セルの削除
- セルの再分割
- 面へ沿わせたセルの変形(スナッピング)

snappyHexMeshの実行前に

- 必要なら：STL形式の形状データを，ケースディレクトリ下のconstant/triSurfaceディレクトリに置く。
- 計算領域の大きさ および 基準メッシュの大きさを決める ヘキサメッシュを作っておく
  - blockMeshユーティリティーを使うことが多い
- ケースディレクトリ下のsystemディレクトリに，snappyHexMeshDictファイルを作成し，設定を記述する。

snappyHexMeshの設定項目

Keyword	Description	Example
castellatedMesh	Create the castellated mesh?	true
snap	Do the surface snapping stage?	true
doLayers	Add surface layers?	true
mergeTolerance	Merge tolerance as fraction of bounding box of initial mesh	1e-06
debug	Controls writing of intermediate meshes and screen printing	
	— Write final mesh only	0
	— Write intermediate meshes	1
	— Write volScalarField with cellLevel for post-processing	2
	— Write current intersections as *.obj* files	4
geometry	Sub-dictionary of all surface geometry used	
castellatedMeshControls	Sub-dictionary of controls for castellated mesh	
snapControls	Sub-dictionary of controls for surface snapping	
addLayersControls	Sub-dictionary of controls for layer addition	
meshQualityControls	Sub-dictionary of controls for mesh quality	

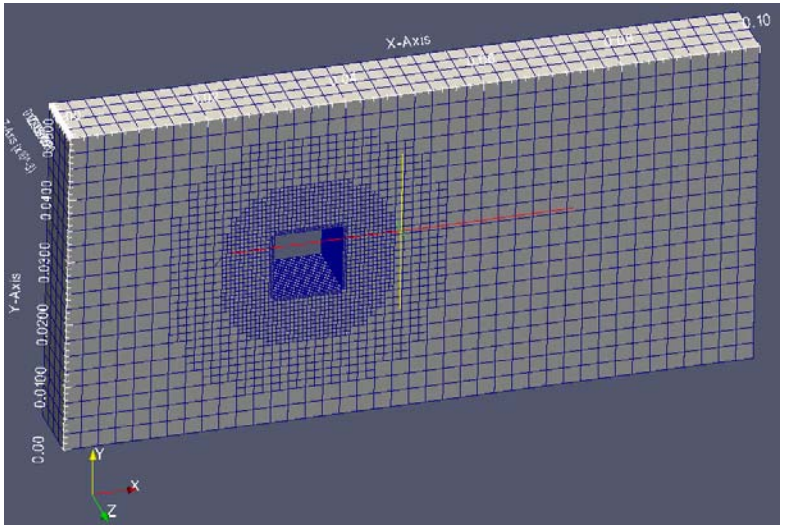
Table 5.7: Keywords at the top level of *snappyHexMeshDict*.

OpenFOAM UserGuide U-146

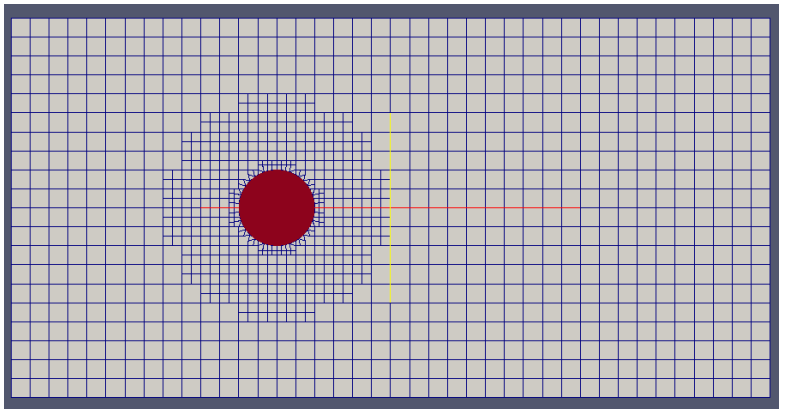
snappyHexMesh 実習1：
既存ケースの実行

例題

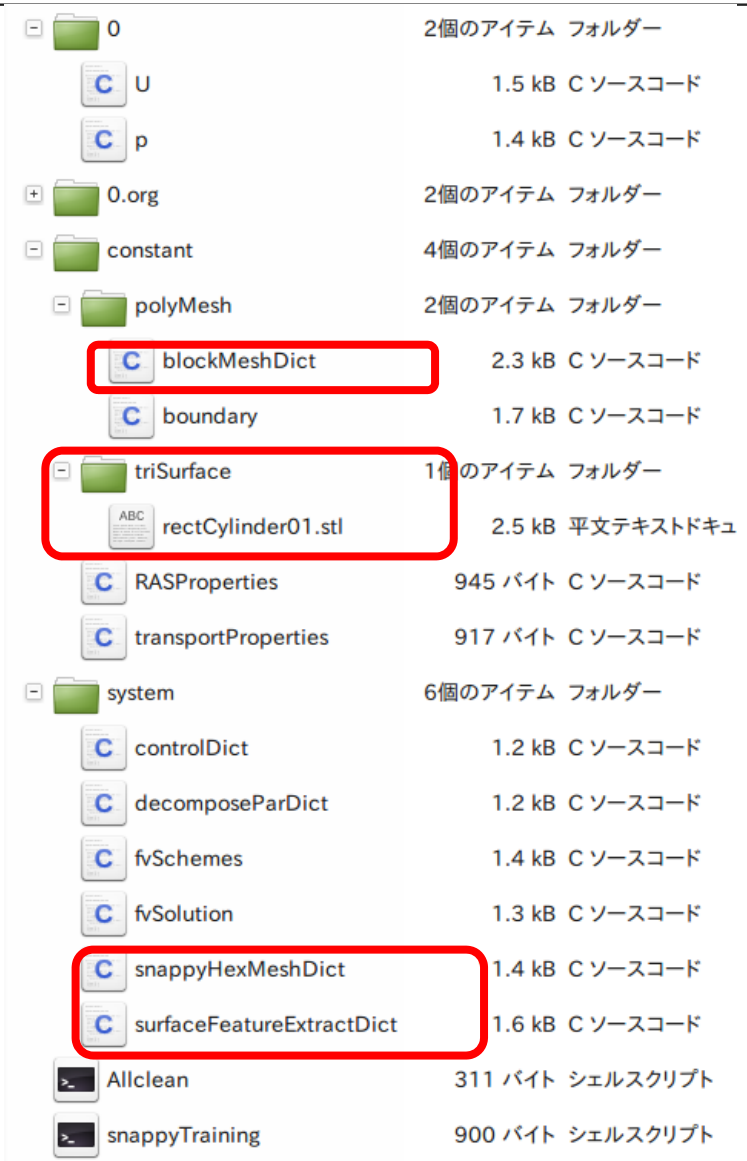
- snappyTestRect
  - 計算領域：直方体
  - 内部に四角柱を設置
  - 角柱周りのセルを細分化



- snappyTestCirc
  - 計算領域：直方体
  - 内部に円柱を設置
    - SalomeでSTLファイル作成
  - 円柱周りのセルを細分化

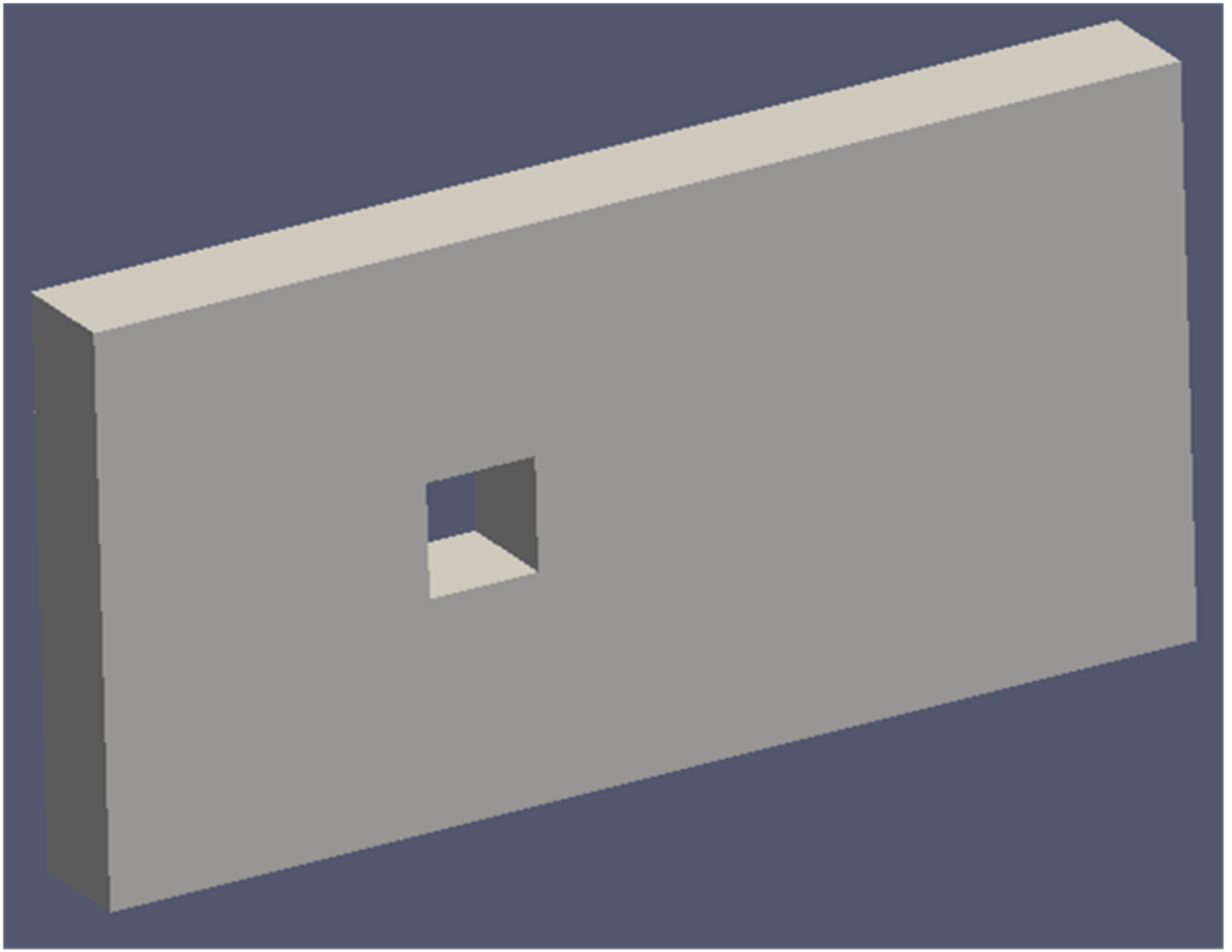


snappyTestRectケース：ファイル構成

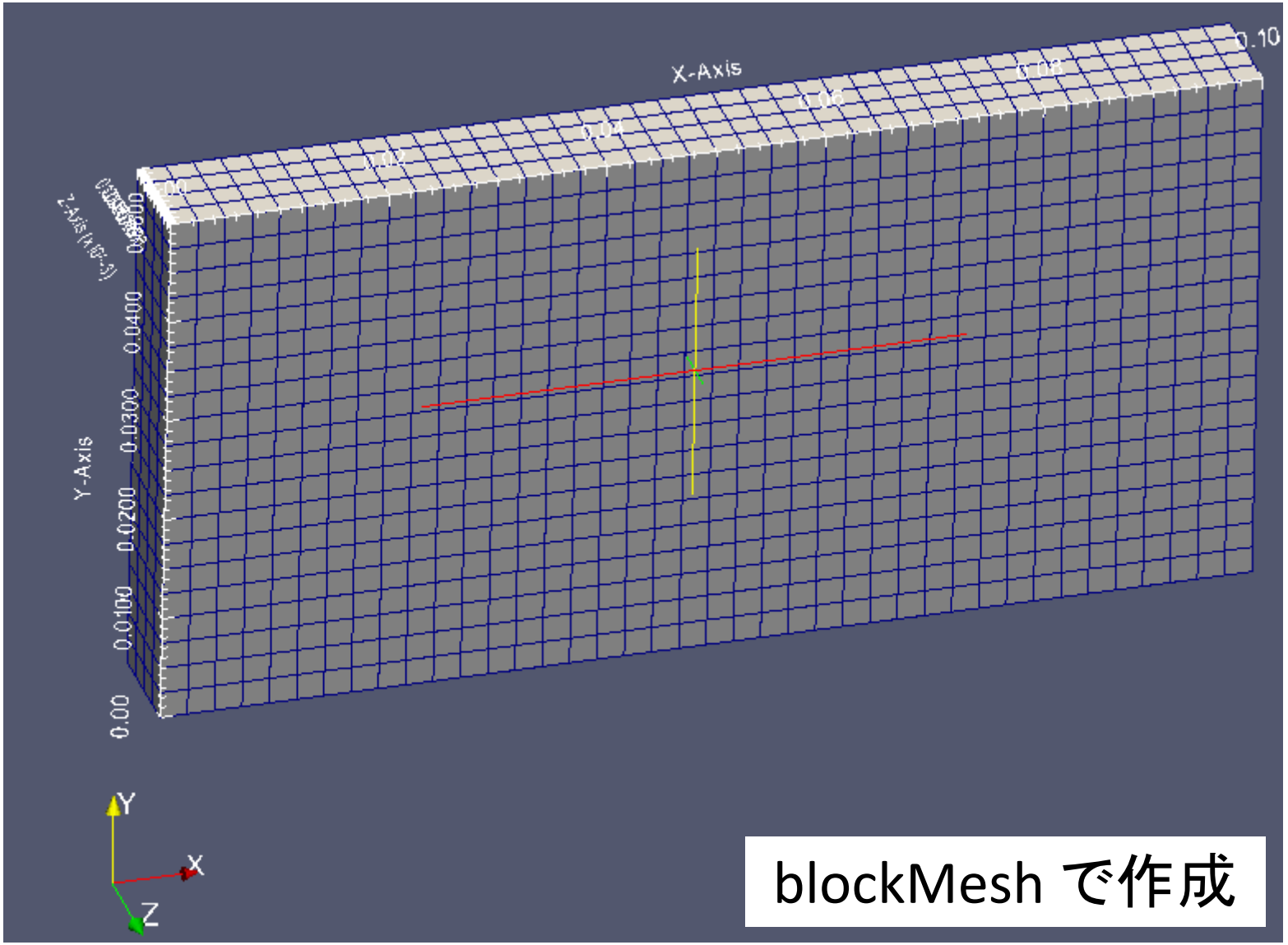


- 形状STLファイル
  - constant/triSurfaceに置く
  - rectCylinder01.stl
- 基準メッシュ設定
  - constant/polyMesh/blockMeshDict
- 特徴抽出設定
  - system/surfaceFeatureExtractDict
- 詳細メッシュ設定
  - system/snappyHexMeshDict

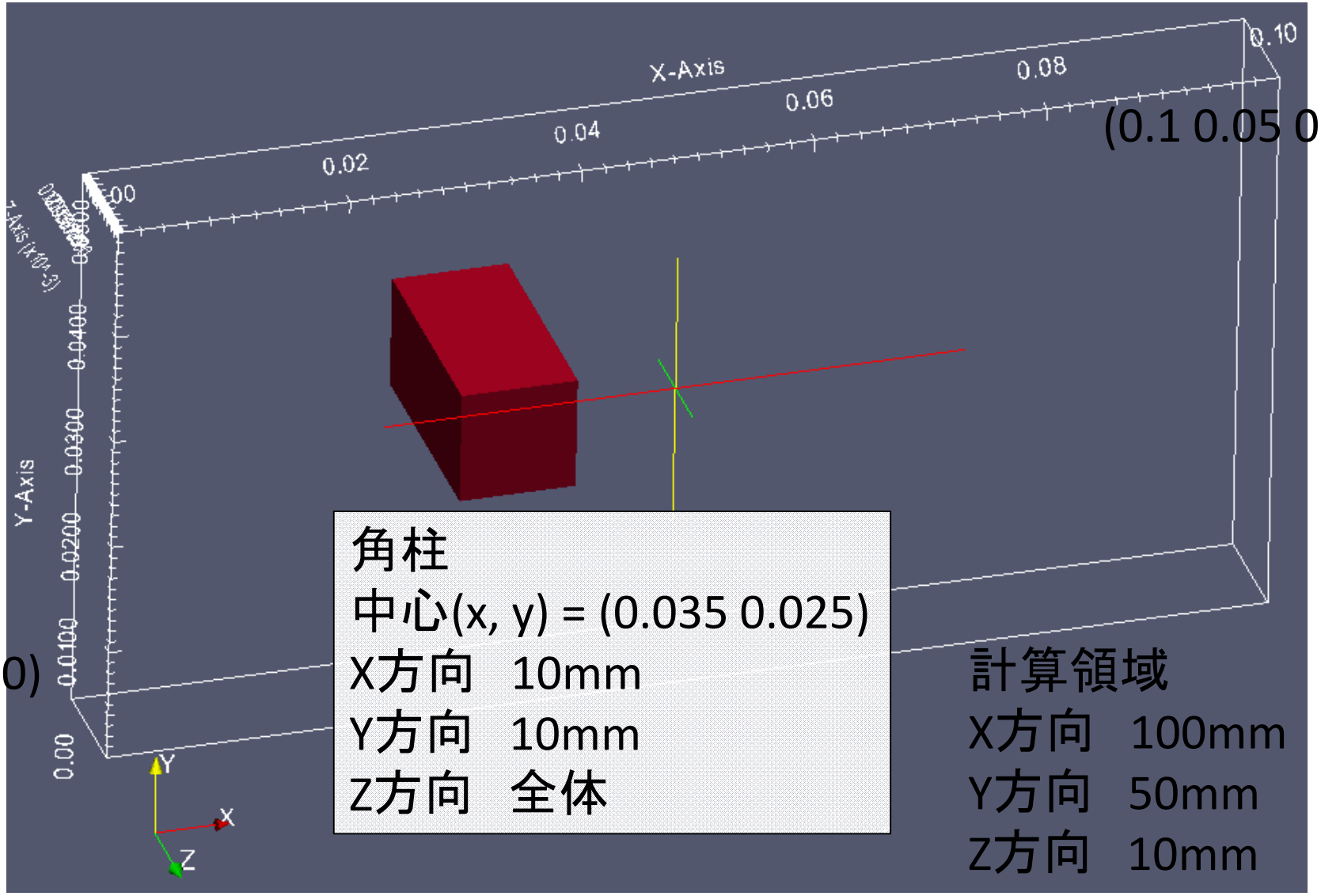
作成したい計算領域



基準となるメッシュ（level 0）



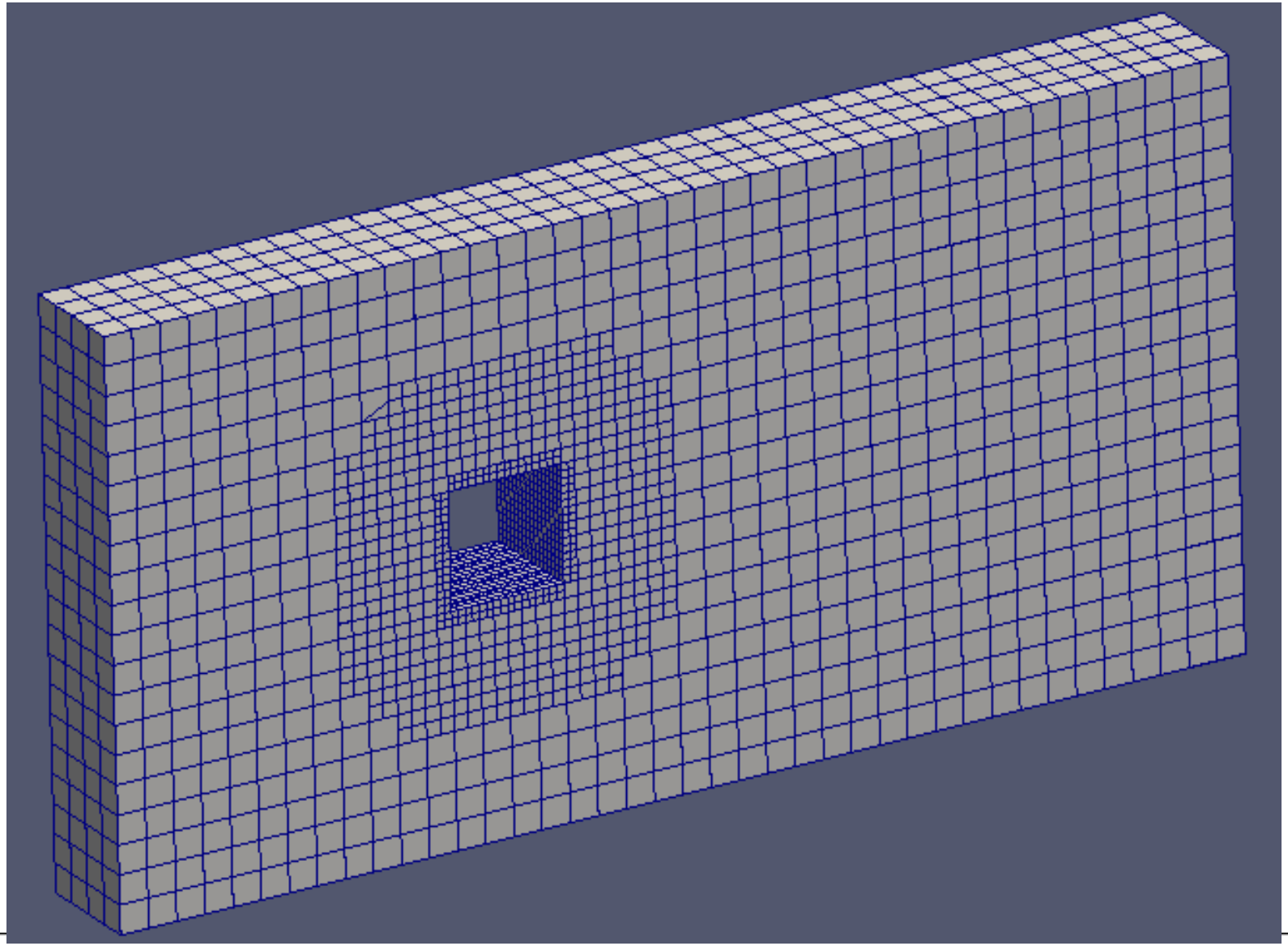
領域内に置いた角柱(STL)



(0.1 0.05 0.01)

(0 0 0)

完成図（例1）



全体の構造を考える

- 計算領域
  - 直方体
    - 座標範囲[m]　(0 0 0) – (0.1 0.05 0.01)
    - 大きさ[m]　(x y z) = (0.1 0.05 0.01)
  - 基準セル
    - 全体を立方体で埋める
    - 1辺の長さ　0.0025m　← 挿入する物体の1辺を4分割
    - 計算領域全体の分割数は？
      - (0.1 0.05 0.01) / 0.0025 = (40 20 4) → 3200
  - 各面ごとを，一つの境界面(patch)とする

blockMeshDict

```
convertToMeters 1;

vertices
(
    (0.0 0.0  0.0)  // 0
    (0.1 0.0  0.0)  // 1
    (0.1 0.05 0.0)  // 2
    (0.0 0.05 0.0)  // 3

    (0.0 0.0  0.01) // 4
    (0.1 0.0  0.01) // 5
    (0.1 0.05 0.01) // 6
    (0.0 0.05 0.01) // 7
);

blocks
(
    hex (0 1 2 3  4 5 6 7)
    (40 20 4)
    simpleGrading (1 1 1)
);
edges();
```

blockMeshDict

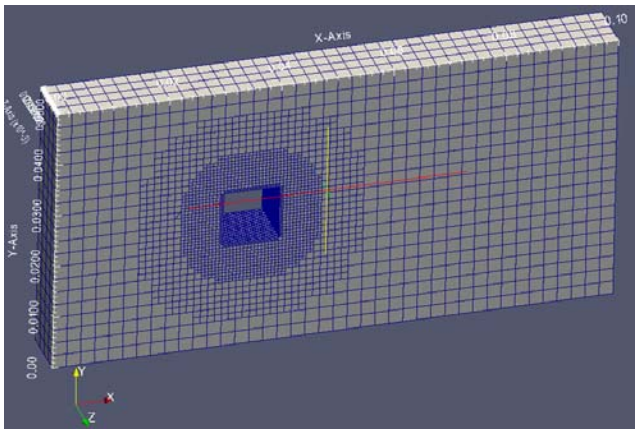
```
boundary(
    xMin
    {
        type patch;
        faces       (       (0 4 7 3)       );
    }
    xMax
    {       type patch;
        faces       (       (1 2 6 5)       );
    }
    yMin
    {
        type wall;
        faces       (       (0 1 5 4)       );
    }
    yMax
    {
        type wall;
        faces       (       (7 6 2 3)       );
    }
    zMin
    {
        type cyclic;
        neighbourPatch zMax;
        faces       (       (3 2 1 0)       );
    }
    zMax
    {
        type cyclic;
        neighbourPatch zMin;
        faces       (       (4 5 6 7)       );
    }
);
```

全体の構造を考える

- 物体
  - 直方体
    - 入口(xMin)面から30mm下流に設定
    - 中心(x, y) = (0.035 0.025)
    - 10mm角の棒
      - 大きさ[m]　(x y z) = (0.01 0.01 領域全体)
    - STLファイルを用意する
      - 今回は，簡単な形状なので，シンプルなソフトで作成した
      - 無料の三次元CG/形状処理ソフトウェア　StoneyDisigner (Winのみ)
      - http://www.stoneydesigner.com/
  - セル
    - 角柱の周囲は，細かなセルを配置
    - 例：レベル1のセル 0.00125=1.25mm角，レベル2　0.625mm角
    - レベル2のセルを16個並べると，角柱の1辺に相当する

snappyHexMeshDict よく使う項目

- geometry
  - STLファイルの読み込み
    - ファイル名，内部での呼び方などを記載
- snapControls
  - nFeatureSnapIter
    - 特徴線を維持するには，指定しておく必要がある
- addLayersControls
  - layers
    - レイヤーを追加したい面を指定



snappyHexMeshDict よく使う項目

- castellatedMeshControls
  - Features
    - STLファイルから作ったeMeshファイルを指定
    - 先にsurfaceFeatureExtractコマンドを実行しておく必要あり。
  - refinementSurfaces
    - 細分化したい面を指定
  - refinementRegions
    - 細分化したい領域を指定
      - distance　面からの距離が指定値以内
      - inside または outside　事前に用意した領域名で指定（領域内or外）
  - locationInMesh
    - メッシュを生成したい領域内部の点を指定
    - この指定間違いで，メッシュ生成できないことがある。

snappyHexMeshDict (geometry)

```
castellatedMesh true;
snap            true;
addLayers       true;
```

```
geometry
{
    rectCylinder01.stl  //STL filename
    {
        type triSurfaceMesh;
        name       cylinder;
        regions
        {
            StoneyDesigner_solid
            {
                name      cylinder;
            }
        }
    }
};
```

snappyHexMeshDict (castellatedMeshControls)

```
castellatedMeshControls
{
    maxLocalCells 100000;
    maxGlobalCells 2000000;
    minRefinementCells 10;
    maxLoadUnbalance 0.10;
    nCellsBetweenLevels 1;

    features
    (
        {
            file "rectCylinder01.eMesh";
            level 2;
        }
    );

    refinementSurfaces
    {
        cylinder
        {
            // Surface-wise min and max
    refinement level
            level (2 2);
        }
    }

    resolveFeatureAngle 60;

    refinementRegions
    {
        cylinder
        {
            mode distance;
            levels ( (0.01 1) );
        }
    }

    locationInMesh (0.0001 0.0001 0.0001);
    allowFreeStandingZoneFaces true;
}
```

snappyHexMeshDict(snap & addLayers)

```
snapControls
{
    nSmoothPatch 3;
    tolerance 4.0;
    nSolveIter 0;
    nRelaxIter 5;
    nFeatureSnapIter 10;
}

addLayersControls
{
    relativeSizes true;
    layers
    {
    }
    expansionRatio 1.0;
    finalLayerThickness 0.3;
    minThickness 0.1;
    nGrow 0;
    featureAngle 30;
    nRelaxIter 3;
    nSmoothSurfaceNormals 1;
    nSmoothNormals 3;
    nSmoothThickness 10;
    maxFaceThicknessRatio 0.5;
    maxThicknessToMedialRatio 0.3;
    minMedianAxisAngle 90;
    nBufferCellsNoExtrude 0;
    nLayerIter 50;
}
```

snappyHexMeshDict(meshQual. etc.)

```
meshQualityControls
{
    maxNonOrtho 65;
    maxBoundarySkewness 20;
    maxInternalSkewness 4;
    maxConcave 80;
    minVol 1e-13;
    minTetQuality 1e-30;
    minArea -1;
    minTwist 0.02;
    minDeterminant 0.001;
    minFaceWeight 0.02;
    minVolRatio 0.01;
    minTriangleTwist -1;

    nSmoothScale 4;
    errorReduction 0.75;
}

debug 0;
mergeTolerance 1e-6;
```

実行手順

- 端末を起動する
- 例題ディレクトリ snappyTestRect に移動する。

cd Desktop/meshTraining/snappyTestRect

- ./snappyTraining と入力して，実行する。自動的に下記コマンドが実行される。
  - blockMesh の実行
    - 基準メッシュ生成
  - surfaceFeatureExtract の実行
    - 角柱の特徴線抽出 → .eMeshファイル生成
  - snappyHexMesh の実行

スクリプト snappyTraining の内容

```
#!/bin/sh
cd ${0%/*} || exit 1    # run from this directory

# Source tutorial run functions
. $WM_PROJECT_DIR/bin/tools/RunFunctions

runApplication blockMesh
runApplication surfaceFeatureExtract

runApplication snappyHexMesh
# runApplication snappyHexMesh -overwrite
# mpirun -np 2 snappyHexMesh -parallel
# reconstructParMesh

# force removal of fields generated by snappy
#¥rm -rf 0
#cp -rf 0.org 0
#runApplication `getApplication`

echo “ ….この後は実行後のメッセージ
```

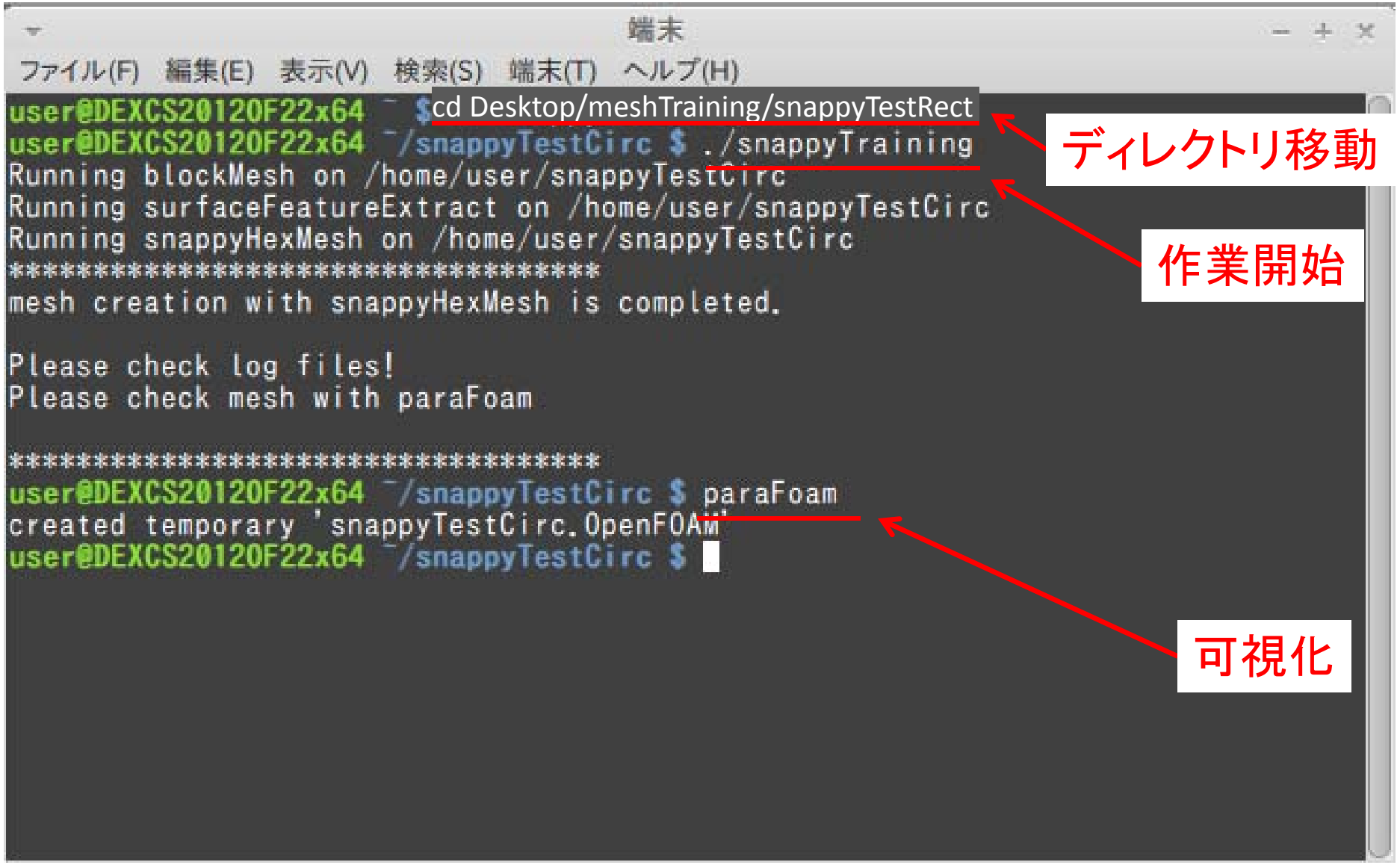
OpenFOAM 2.2.0 からディクショナリを使うため，オプション不要
旧バージョンではオプション必要

途中経過も残す
(0, 1, 2, 3 のディレクトリ)
最終結果だけなら，-overwriteオプションを付ける

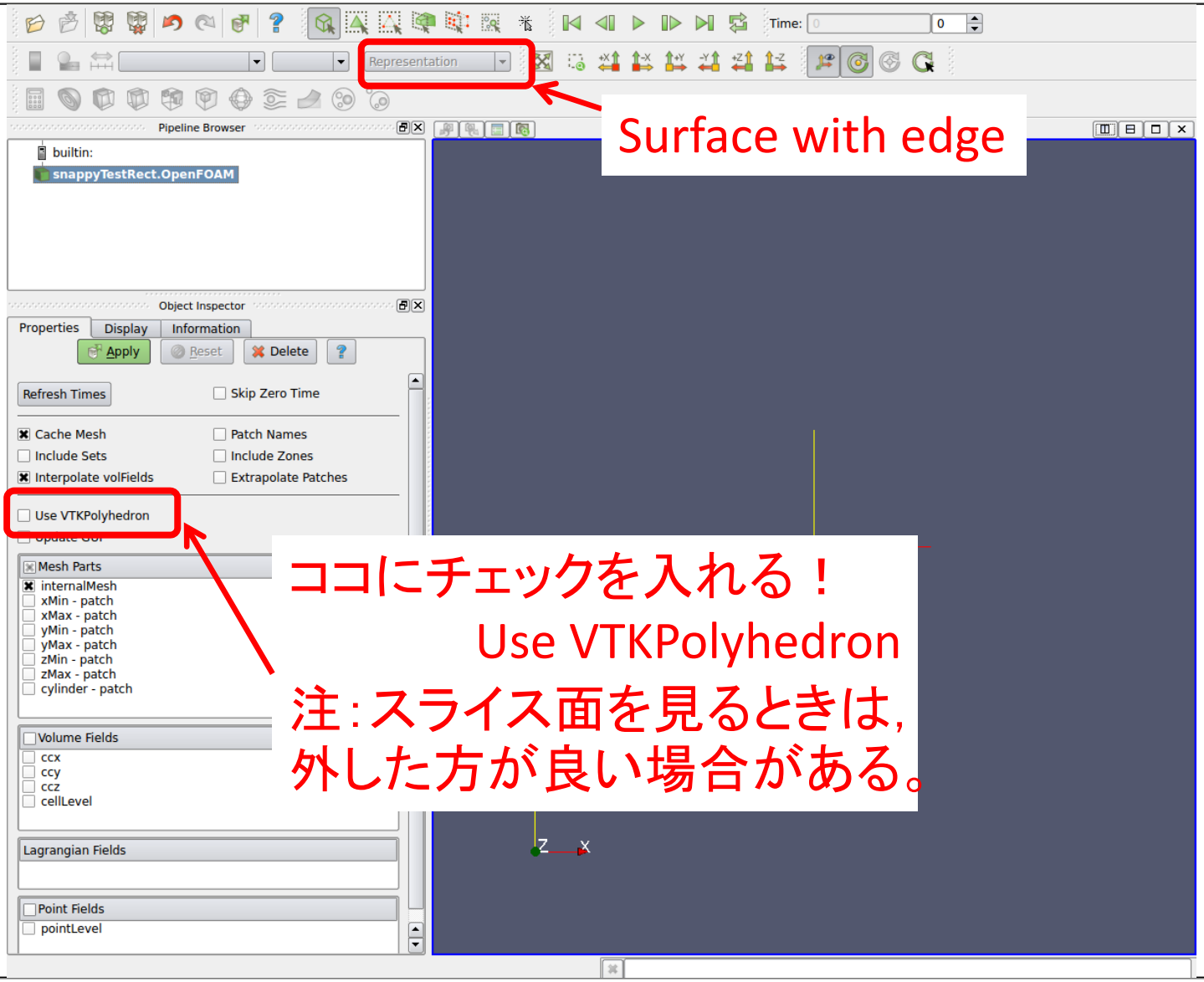
#現在はコメントアウト
並列実行も可能
実行後に結合する

#現在はコメントアウト
計算実行

作業


端末
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
user@DEXCS20120F22x64 ~ $cd Desktop/meshTraining/snappyTestRect
user@DEXCS20120F22x64 ~/snappyTestCirc $ ./snappyTraining
Running blockMesh on /home/user/snappyTestCirc
Running surfaceFeatureExtract on /home/user/snappyTestCirc
Running snappyHexMesh on /home/user/snappyTestCirc
mesh creation with snappyHexMesh is completed.
Please check log files!
Please check mesh with paraFoam
user@DEXCS20120F22x64 ~/snappyTestCirc $ paraFoam
created temporary 'snappyTestCirc.OpenFOAM'
user@DEXCS20120F22x64 ~/snappyTestCirc $
ディレクトリ移動
作業開始
可視化

メッシュをキレイに見るために



実行手順（おまけ）

- 作業に失敗したときや，条件を変えて再度実行したい場合には，./Allclean と入力して，実行する。
- 自動的に不要なファイルが削除され，元の状態に戻る。

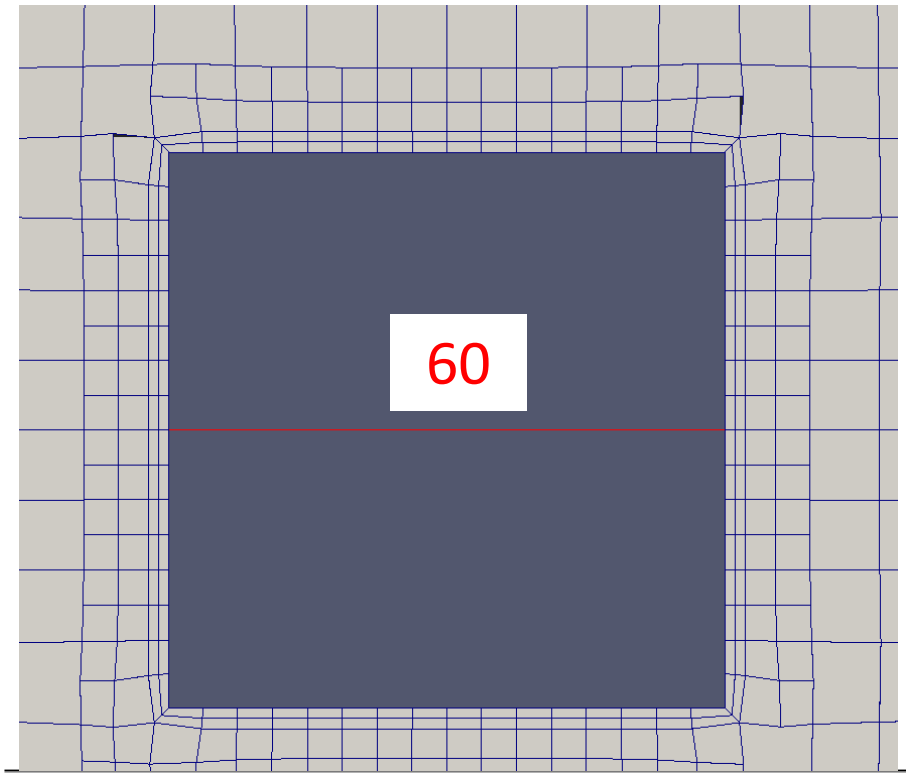
トライ:改造してみる

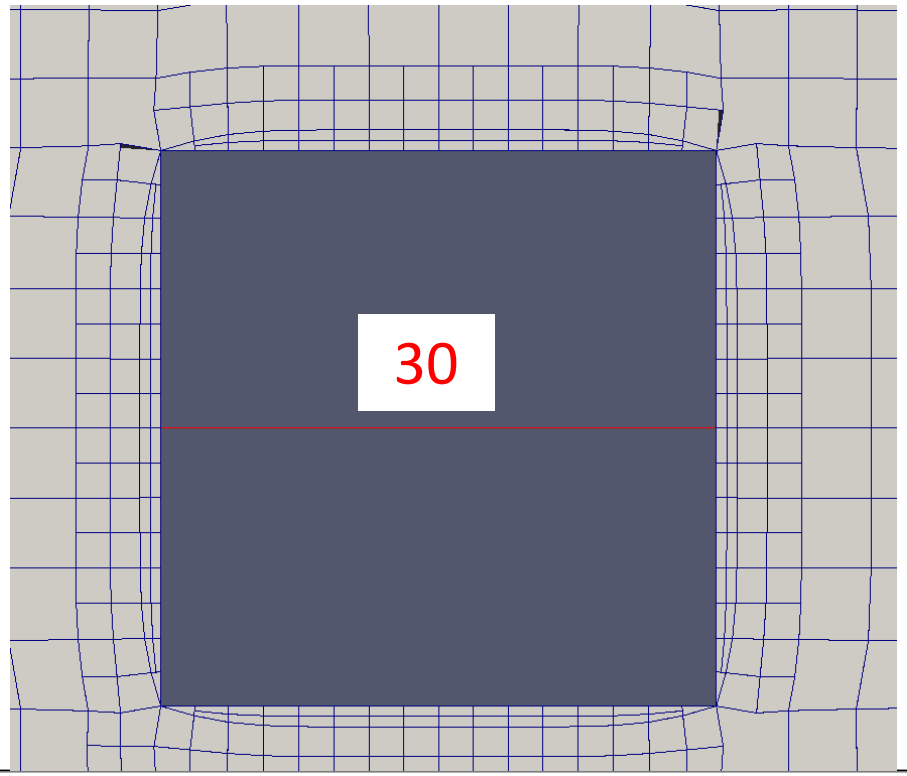
- ./Allclean の実行
- snappyHexMeshDictの編集
  - たとえば
    - Levelを変えてみる
    - 領域再分割の範囲を変更してみる
    - 領域再分割の範囲指定方法を変えてみる
    - Layerを入れてみる
    - locationInMeshの座標を物体内部(0.0301 0.0201 0.0001)にしてみる
    - などなど
- ./snappyTraining の実行
- paraFoam で変化の確認

Layer追加例

//- When not to extrude surface. 0 is flat surface, 90 is when two faces make straight angle.

addLayersControls の featureAngle 60;





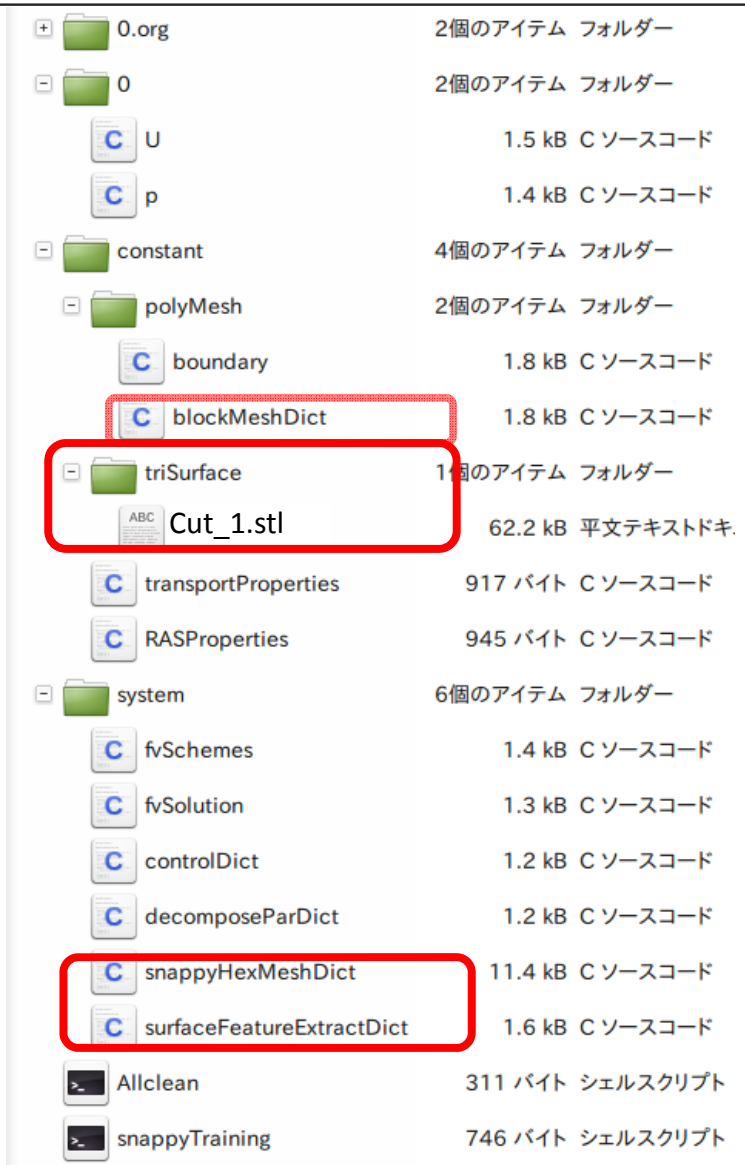
次は・・・

- Salome-meca で形状を作成 → STL形式でエキスポート
- このSTL形式ファイルを使って，snappyHexMeshを実行する
  - ファイルをconstant/triSurface に置く。
  - surfaceFeatureExtractDict を編集し，STLファイル名を変更する。
  - snappyHexMeshDict を編集し，STLファイルに関連する項目を変更する。

講師の
実演のみ

障害物の形状を変える
SALOME-MECA で形状作成

snappyTestCircTryケース：ファイル構成



- 形状STLファイル
  - constant/triSurfaceに置く
  - **Cut_1.stl**
    - **Salome から Export したファイル**
- 基準メッシュ設定
  - constant/polyMesh/blockMeshDict
- 特徴抽出設定
  - system/surfaceFeatureExtractDict
- 詳細メッシュ設定
  - system/snappyHexMeshDict

snappyTestCircTry

例題チュートリアル snappyTestCircTry

- 先の例題 snappyTestRect から，STLファイルを除いた。
- その代わりに，Salome で希望する形状を作成
→ Export してSTLファイルをつくる。（講師実演済み）
- このSTLファイルを使って，メッシュを作ってみる。（salomeFilesディレクトリに格納してある。）

STLファイルと関連する変更箇所

- surfaceFeatureExtractDict

- snappyHexMeshDict
  - geometry
    - STLファイル名，regions名など
  - castellatedMeshControls
    - Features
      - eMeshファイル名
  - refinement関係
    - 細分化指定で，関係する所

作業:ディクショナリ等の修正

- constant/triSurface に STLファイル（Cut_1.stl）を置く。
- STLファイルをエディタで開いて、ファイル冒頭にあるregion名（今回は solid ）を確認する。
- system/surfaceFeatureExtractDict を開き，STLファイル名（Cut_1.stl）を修正する。
- system/snappyHexMeshDict の下記を修正する。
  - geometry のSTLファイル名を修正する。
    （ rectCylinder01.stl から Cut_1.stl へ）
  - geometry のSTLファイル設定内のregion名を修正する。
    (StoneyDesigner_solid から solid)
  - castellatedMeshControls の features の ファイル名を修正する。（rectCylinder01.eMesh を Cut_1.eMesh）

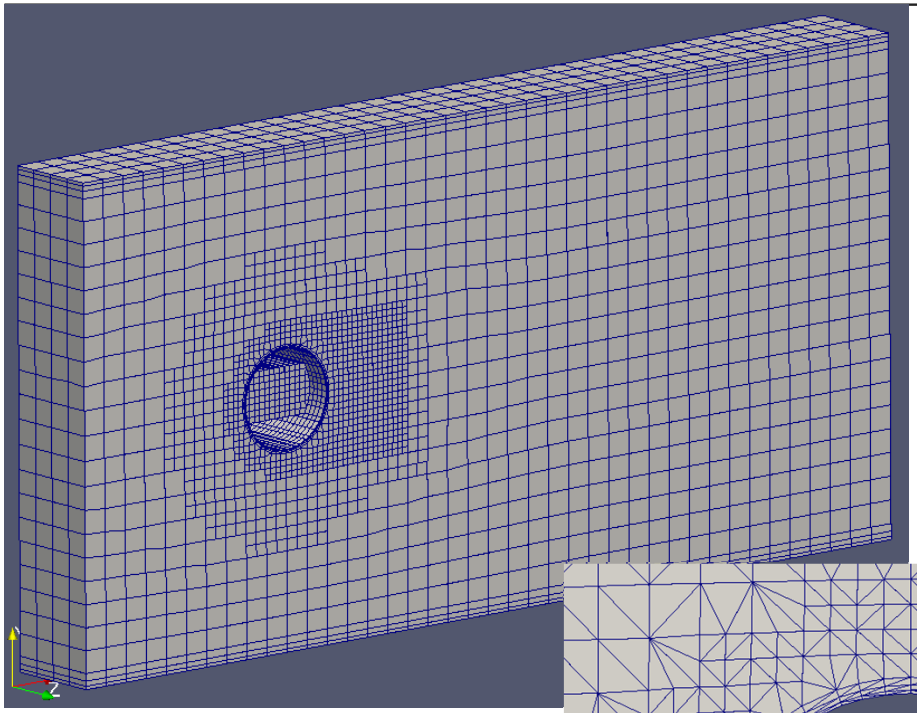
作業:実行手順

- 端末を起動する
- 例題ディレクトリ snappyTestCircTry に移動する。

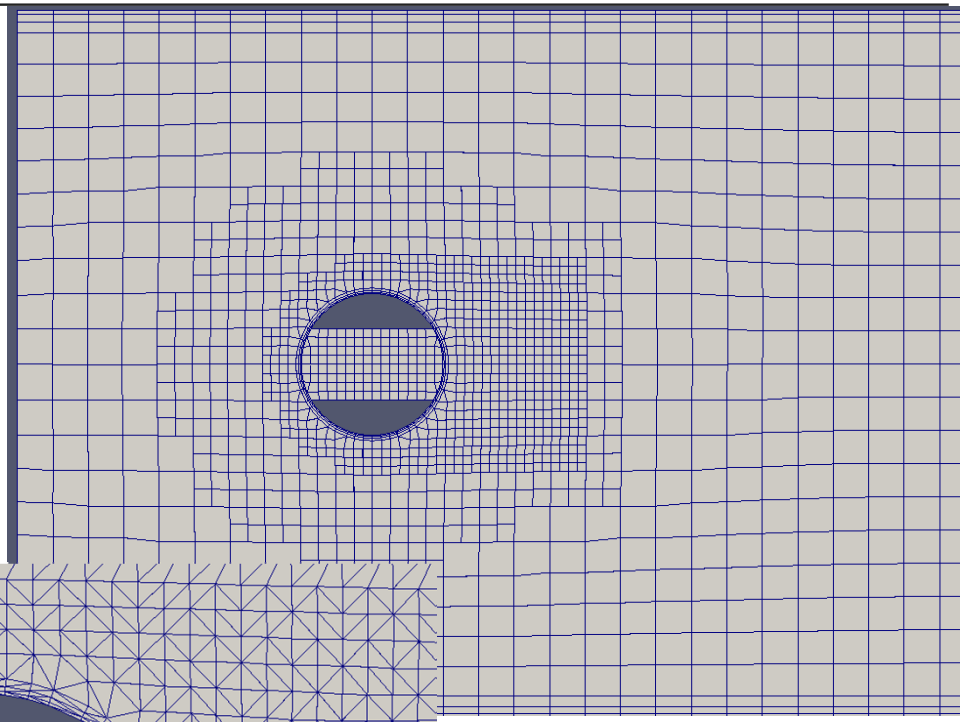
cd Desktop/meshTraining/snappyTestCircTry

- ./snappyTraining と入力して，実行する。自動的に下記コマンドが実行される。
  - blockMesh の実行
    - 基準メッシュ生成
  - surfaceFeatureExtract の実行
    - 角柱の特徴線抽出 → .eMeshファイル生成
  - snappyHexMesh の実行

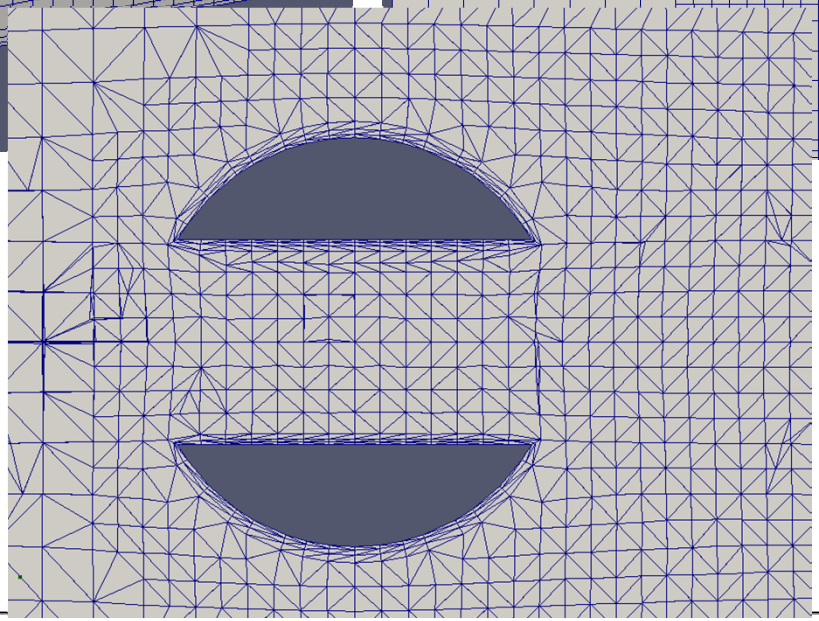
メッシュ完成図



↑　全体



↑ 物体付近 正面



物体付近 断面 →

講師の
実演のみ

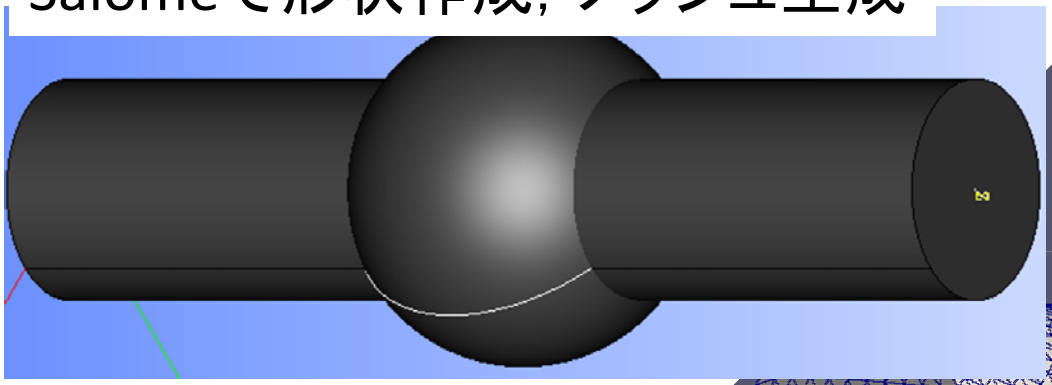
時間的制約のため，Salome使用部分は，講師による実演とします。
興味のある方は，講習後におためしください。

SALOME-MECA 形状とメッシュ作成

計算対象例：膨らみ部のあるパイプ

例題：salomeMeshImported

Salomeで形状作成，メッシュ生成



できたメッシュを
OpenFOAM形式に
変換

OpenFOAMで計算

salomeMeshImportedTry ケース

- blockMesh例題と同様なケース

- salome-meca からエクスポートしたUNVメッシュファイル（cylinderSphere.unv）をケースディレクトリ内に置く。（salomeFilesディレクトリに格納してある。）
- ケースディレクトリから，下記コマンドを実行して，UNVメッシュをOpenFOAM形式に変換する。

ideasUnvToFoam cylinderSphere.unv

作業:実行手順

- 端末を起動する
- ディレクトリ salomeMeshImportedTry に移動

cd Desktop/meshTraining/ salomeMeshImportedTry

- Ideas-UNVメッシュの変換ユーティリティ ideasUnvToFoam を実行する。オプションにUNV ファイル名を与える。

ideasUnvToFoam　cylinderSphere.unv

- paraFoamを実行する

paraFoam

謝辞

- 様々なオープンソースソフトウェアの開発者の皆様に感謝します。
- 様々なオープンソースソフトウェアに関する情報を公開してくださる皆様に感謝します。
- 様々な技術情報・ノウハウを公開してくださる皆様に感謝します。

おまけ：商用メッシャ

CubitでOpenFOAM用メッシュを生成

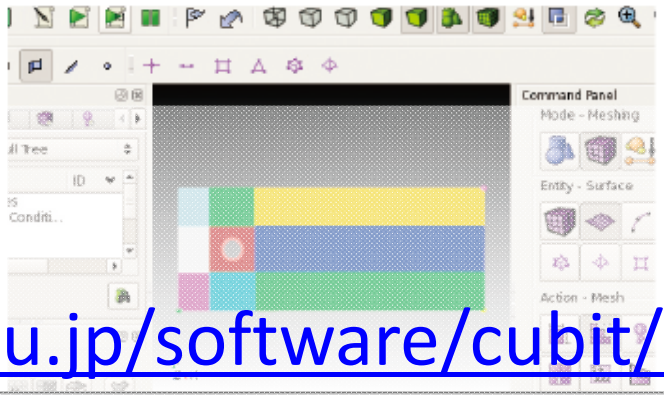
CUBIT 13.1
CAE Pre-Processing ToolKit

■CubitでOpenFOAM用メッシュを生成

Cubitには、優秀な領域分割機能(Decomposition Tools)が備わっており、手軽にヘキサメッシングをすることが出来ます。またFluent形式に出力することでOpenFOAM用のメッシャーとして利用することができます。
ここでは円柱周り流れ解析用メッシュの生成手順を紹介します。

●直感的で使いやすいGUI

簡単な操作はマニュアル無しですらすらと進められます。



●GUIとCUI

CubitではGUIの操作を行うと、対応したコマンドがジャーナルファイルとして作成されます。このファイルを参照することで、モデルやメッシュの編集が簡単に実行できます。

■ジャーナルファイル

1 形状データを作成

```
create vertex -0.5 -0.2 0
create vertex 0.5 -0.2 0
create vertex 0.5 0.2 0
create planar surface with plane vertex 1 vertex 2 vertex 3
create surface circle radius 0.03 zplane
move Surface 2 location -0.3
include_merged
```

CUBIT
爆発研究所
http://bakuhatsu.jp/software/cubit/ofmesh/

おまけ：商用メッシャ

Pointwise

- OpenFOAM形式で直接出力
- 構造格子、非構造格子