OpenFOAM講習：
メッシュ生成超入門　実習

6/10改訂版

2013年6月8日

オープンCAE勉強会＠富山

中川慎二

注意

- この資料と例題ケースは，OpenFOAM 2.2.0 をベースにして作成しました。(2.1.0用のケースもあります。)

配布ケースについて

- blockMesh 練習ケース
  - OF2.2.0 と 2.1.0 で同じものが使える
  - blockMeshCases 内の，bmTest01　～　bmTest05
- snappyHexMesh 練習ケース
  - OF2.2.0 と 2.1.0 で，実行用スクリプトが異なる。（surfaceFeatureExtract 実行時のオプション）
  - OF2.2.0 では，surfaceFeatureExtractDict を system ディレクトリに格納。
  - snappyHexMeshCases 内
    - OF220_case　　OF2.2.0用
    - OF210_case　　OF2.1.0用

メッシュ生成ソフト

OpenFOAM

- blockMesh ユーティリティー
- snappyHexMesh ユーティリティー

その他のオープンソースソフト

- Salome-meca
- 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 () の前は，空白を入れる.

blockMeshDict の使い方

- blockMeshDict
  - 数字を直接書き込む
  - m4 を利用して，汎用化
  - Dictionary にコード（プログラム）を書いて，汎用化

blockMeshDict の全体構造

```
/*--------------------------------*- 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*

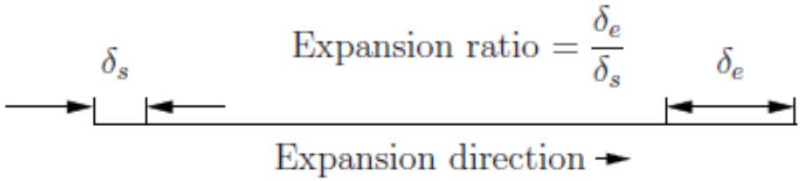


Figure 5.5: Mesh grading along a block edge

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

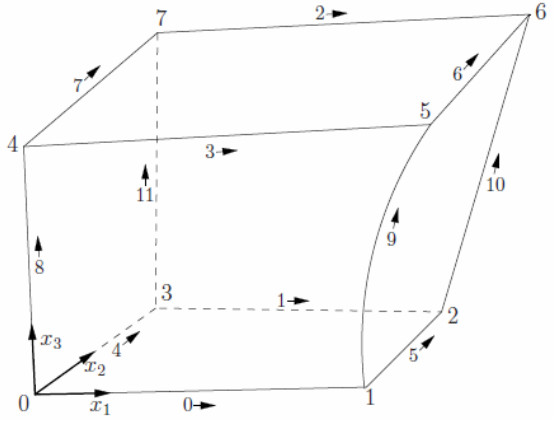
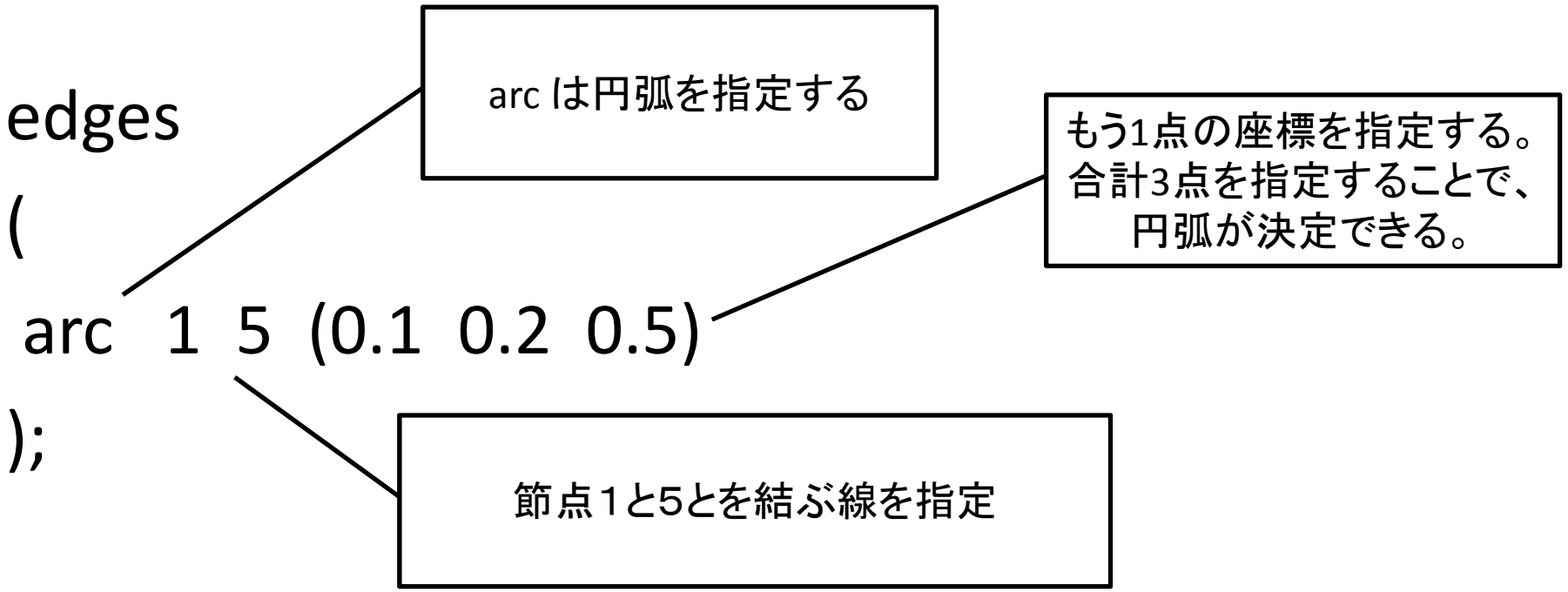


Figure 5.4: A single block

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
simpleSpline	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
polySpline	スプライン線の組	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)
      );
   }
);
```

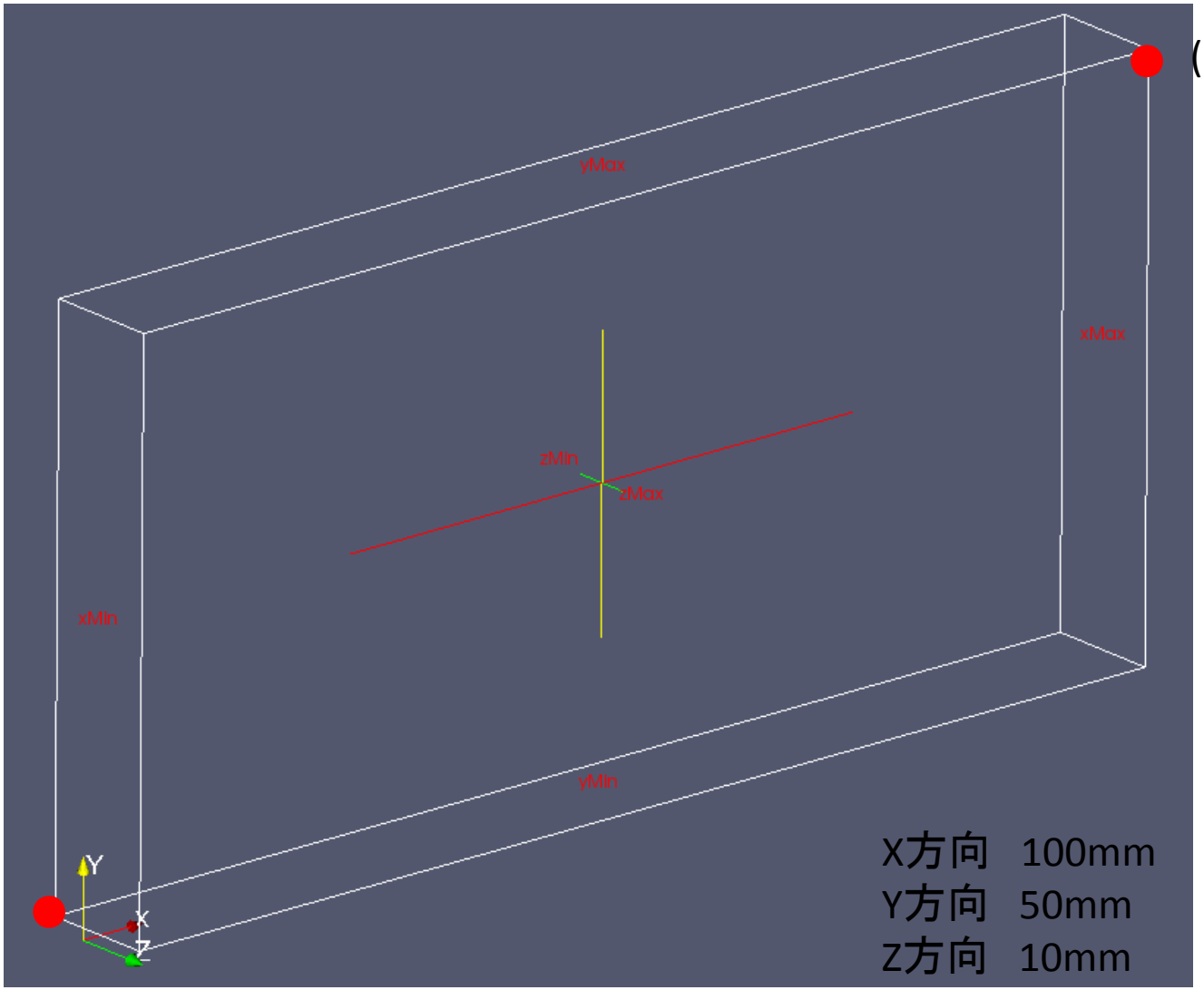
名前：任意に付けて良い。ただし，他のファイルの情報（boudary, U, p など）と一致させること。

patchのタイプ：境界条件に応じた型を与える。

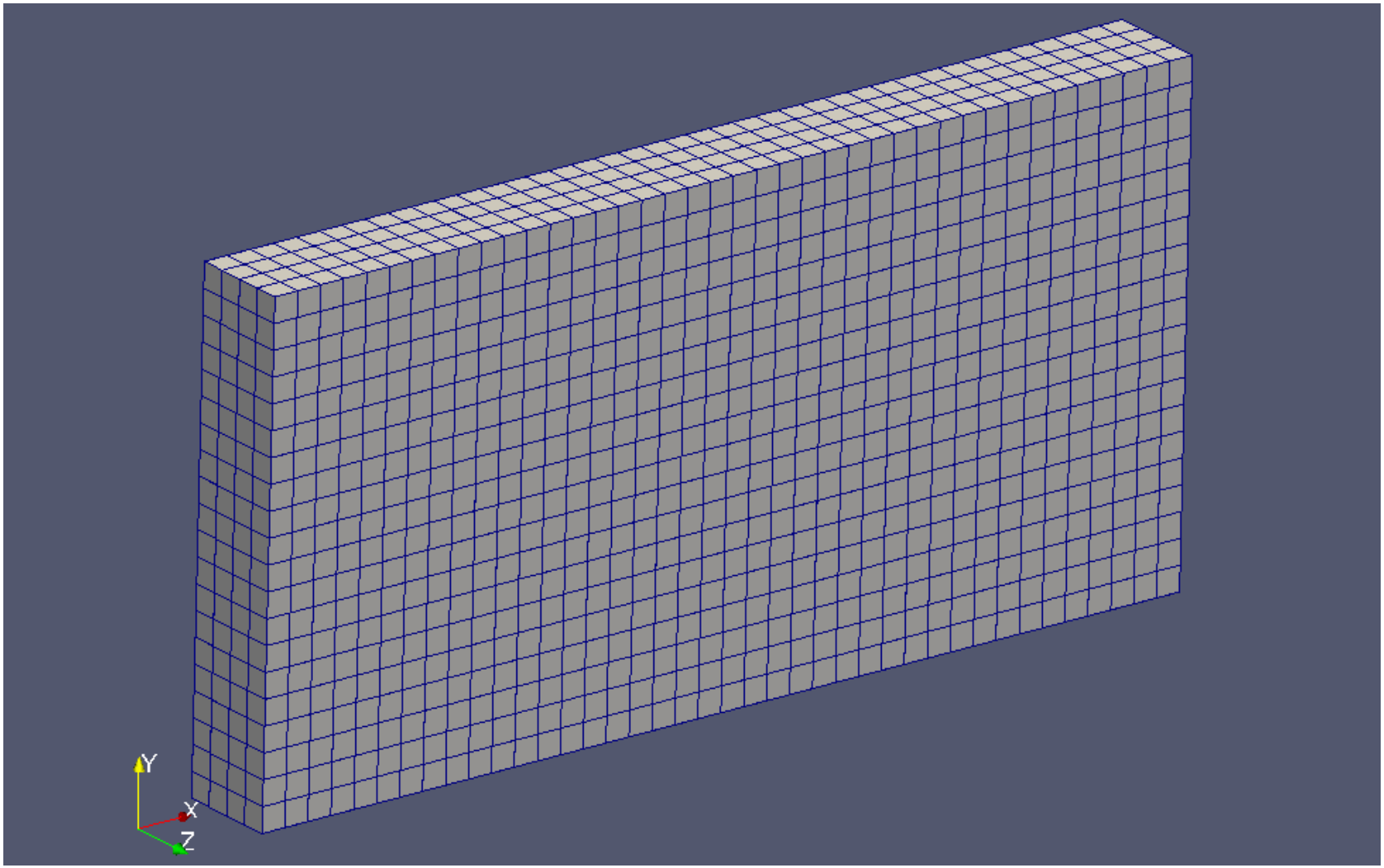
blockMesh 実習

題材

- 直方体（1ブロック）
- 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
- （角柱＋円柱）を置いた流路
  - edge の利用


(0.1 0.05 0.01)
yMax
xMax
zMin
zMax
xMin
yMin
(0 0 0)
Y
X
Z
X方向　100mm
Y方向　50mm
Z方向　10mm

テスト ケース1 : bmTest01



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？
- 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

作業

- ケースディレクトリ /constant/polyMesh/blockMeshDict ファイルの内容を確認する。

- 端末を起動し，ケースディレクトリに移動する。
- blockMesh ユーティリティを実行する。

  blockMesh
- paraFoam を実行して，メッシュを確認する。

  paraFoam

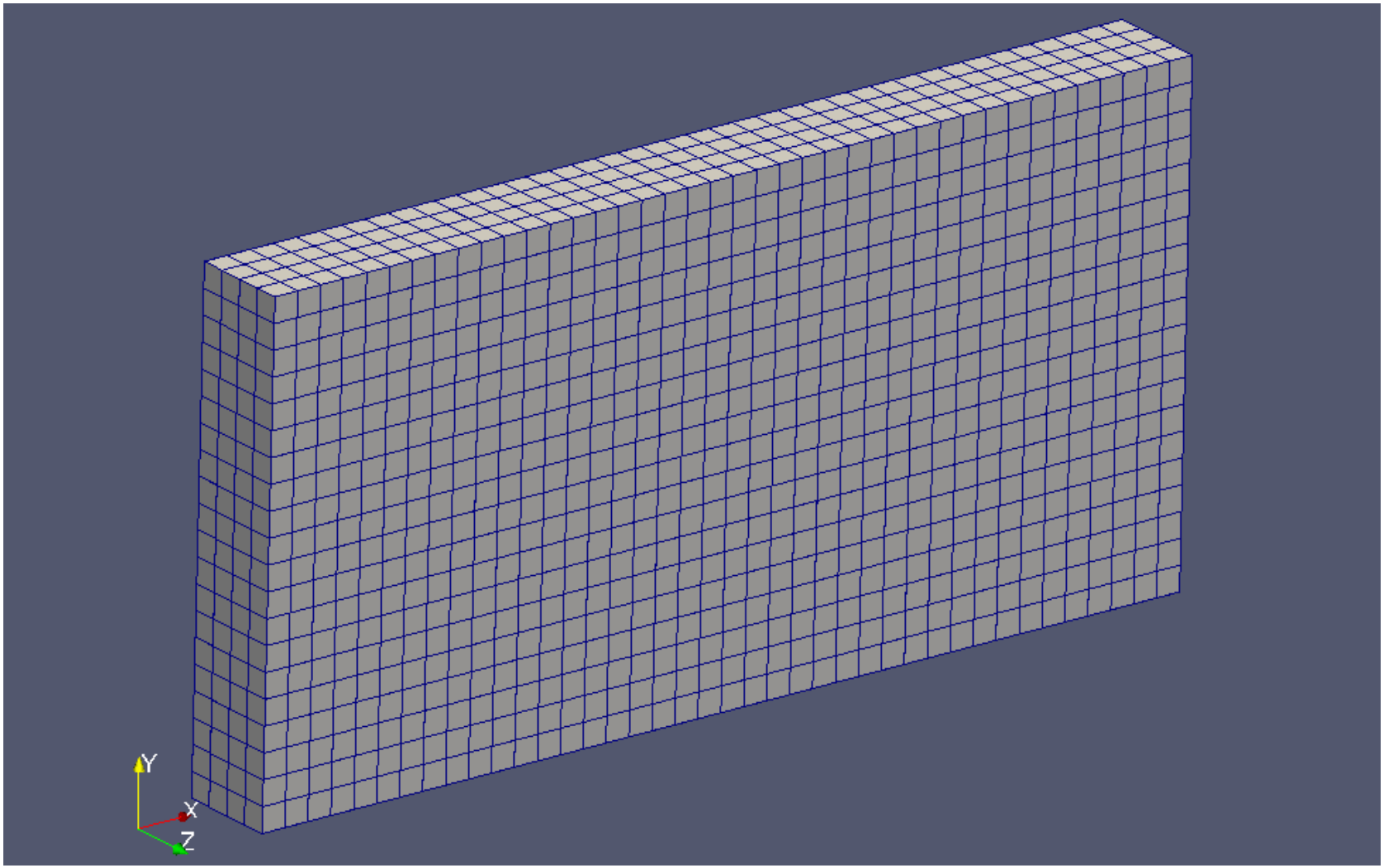
bmTest01: blockMeshDict前半

```
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
(
);
```

bmTest01: blockMeshDict後半

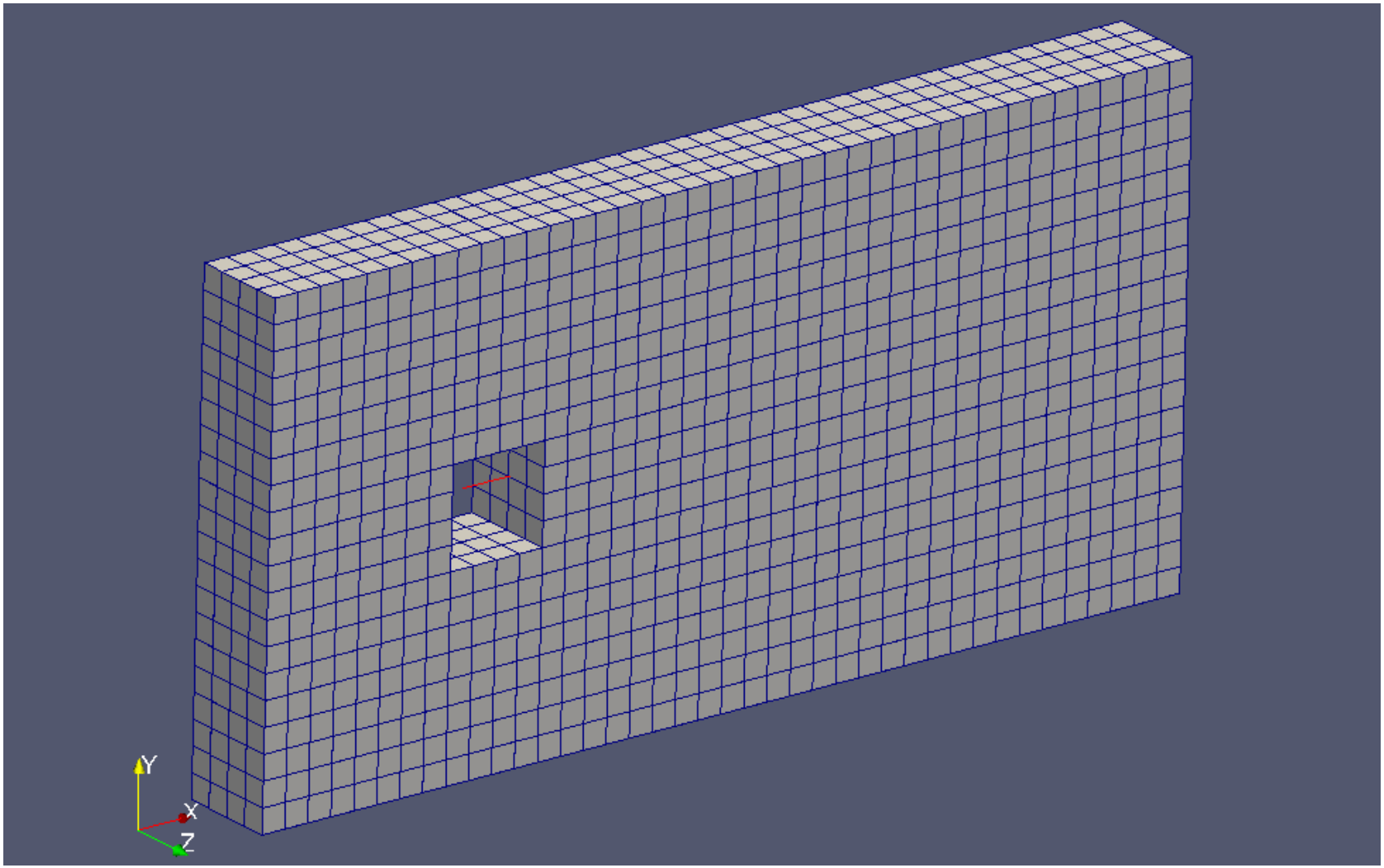
```
boundary
(
    xMin
    {
        type patch;
        faces
        (
            (0 4 7 3)
        );
    }
    xMax
    {
        type patch;
        faces
        (
            (1 2 6 5)
        );
    }
    yMin
    {
        type wall;
        faces
        (
            (0 1 5 4)
        );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

テスト ケース1 : bmTest02

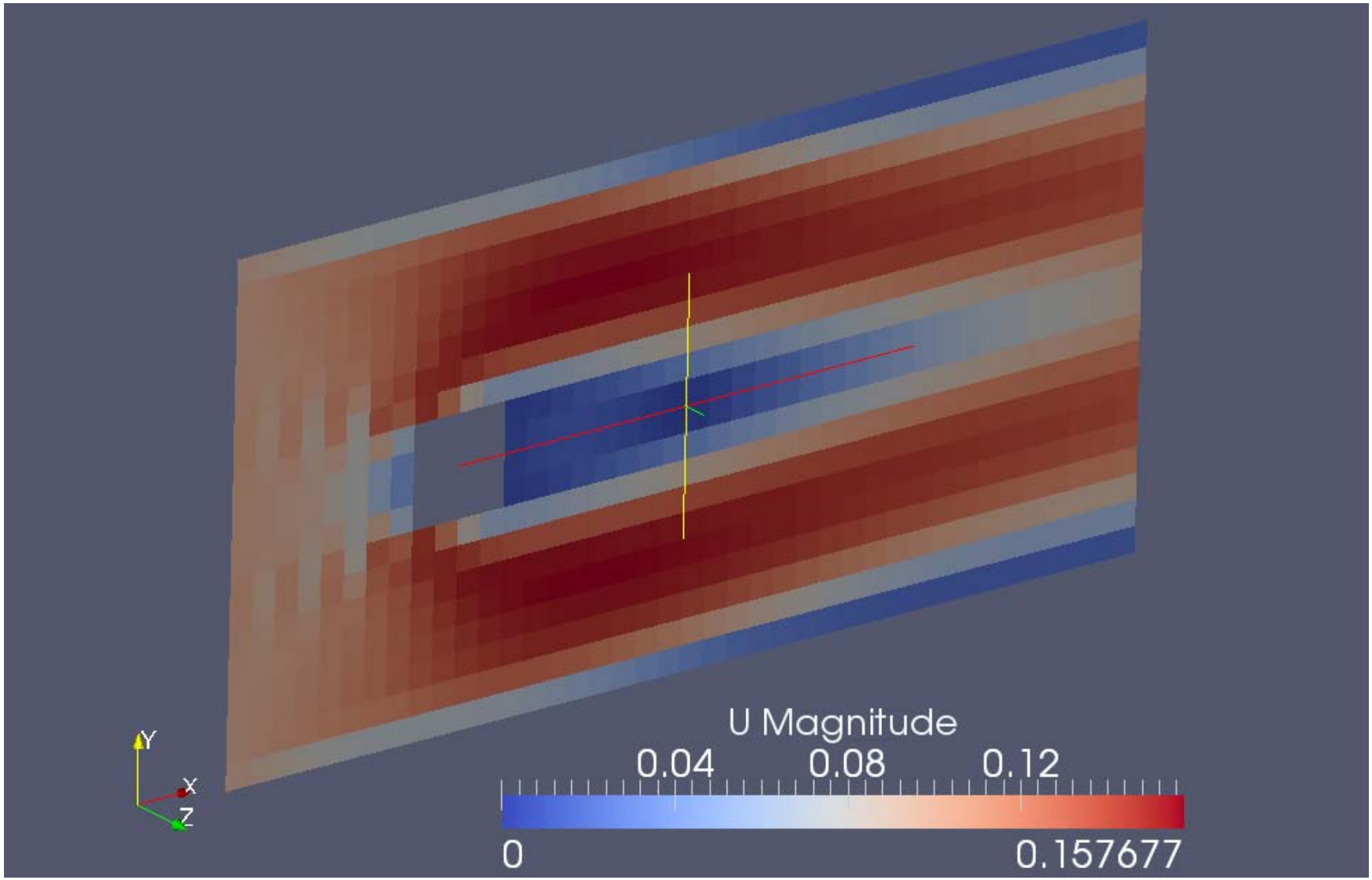


- 次のテストケースに向けて，bmTest01と同じものを，複数のブロックで構成

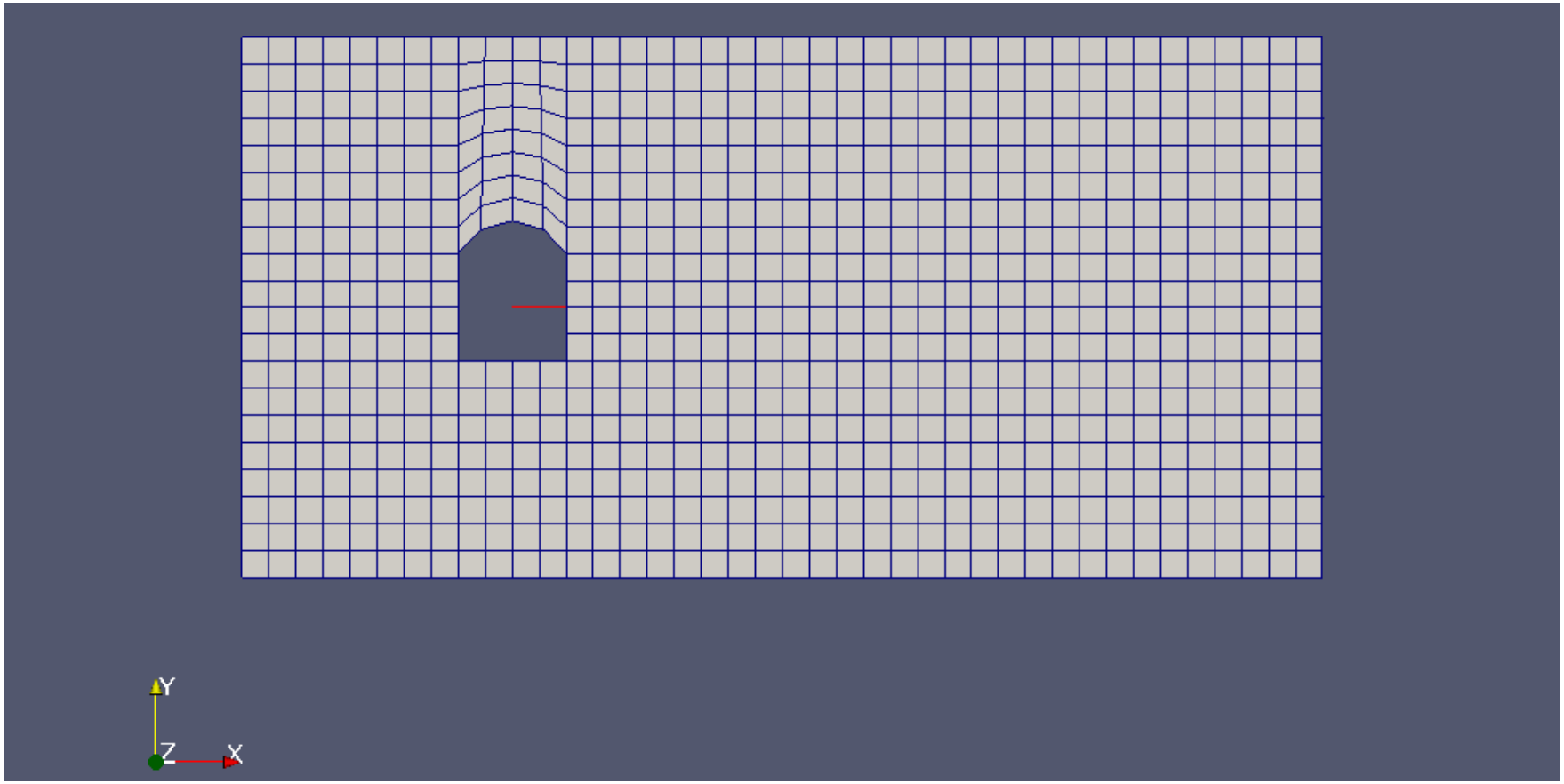
テスト ケース1 : bmTest03



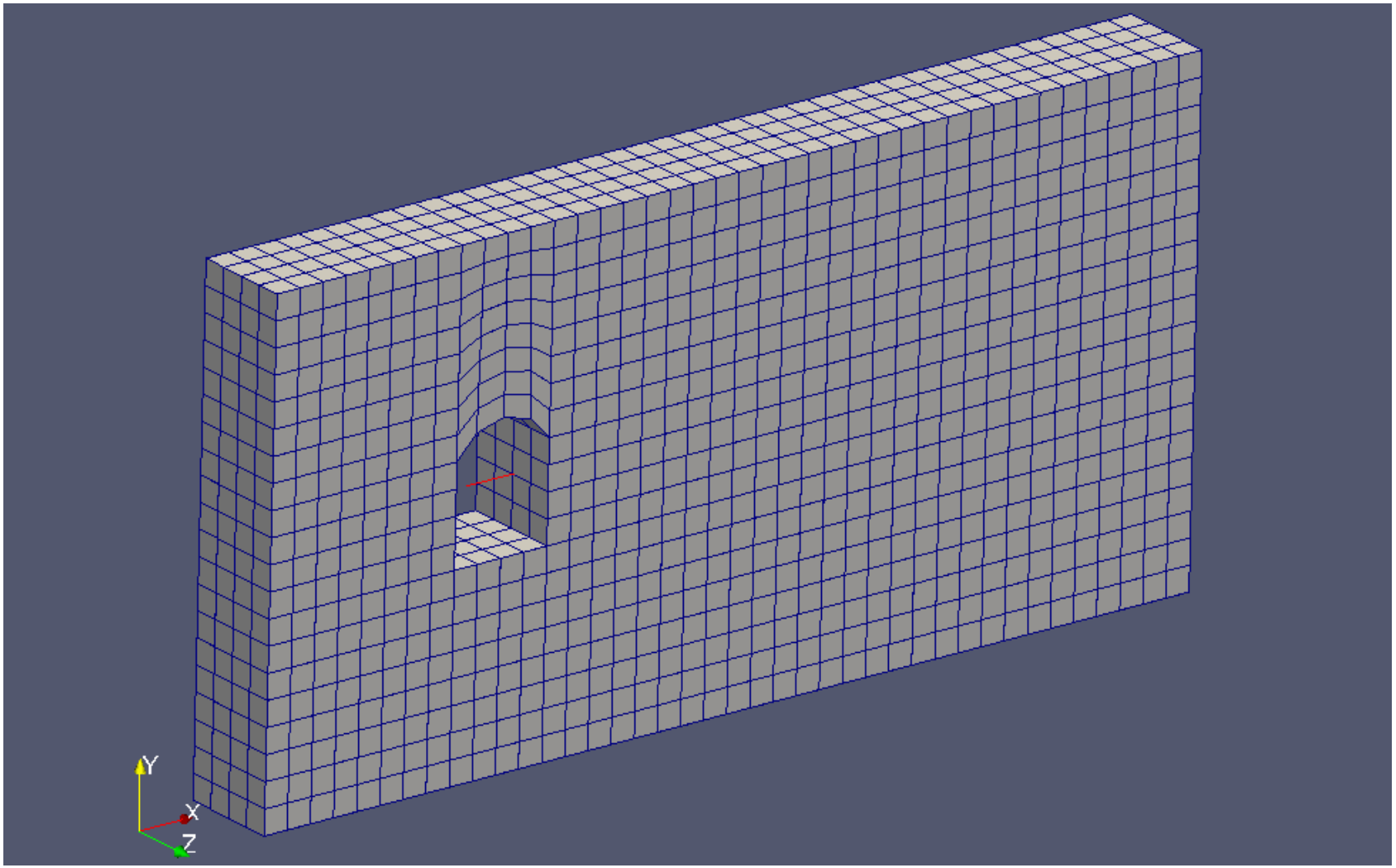
テスト ケース1 : bmTest03



テスト ケース1 : bmTest04



テスト ケース1 : bmTest04



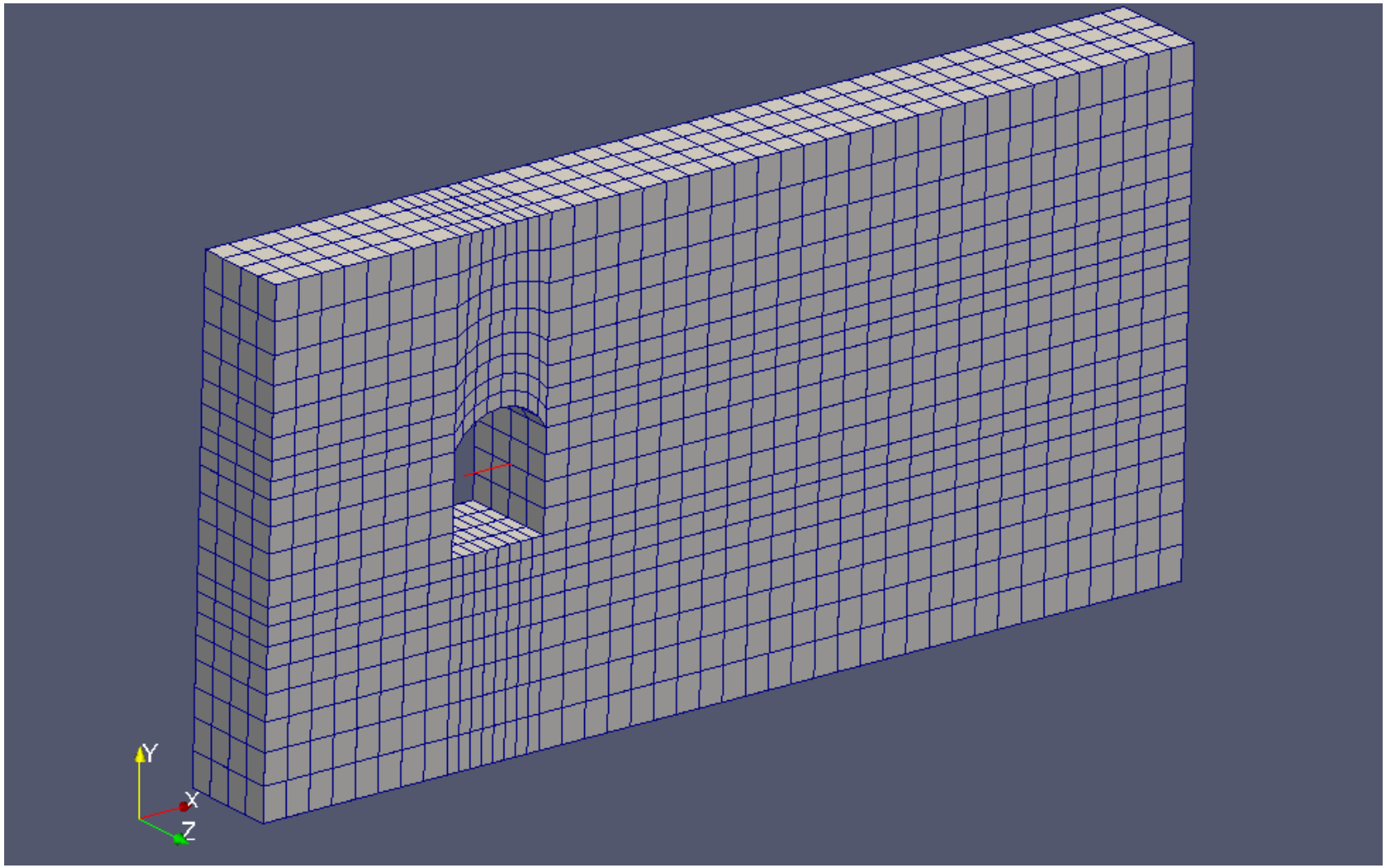
テスト ケース1 : bmTest04

- 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

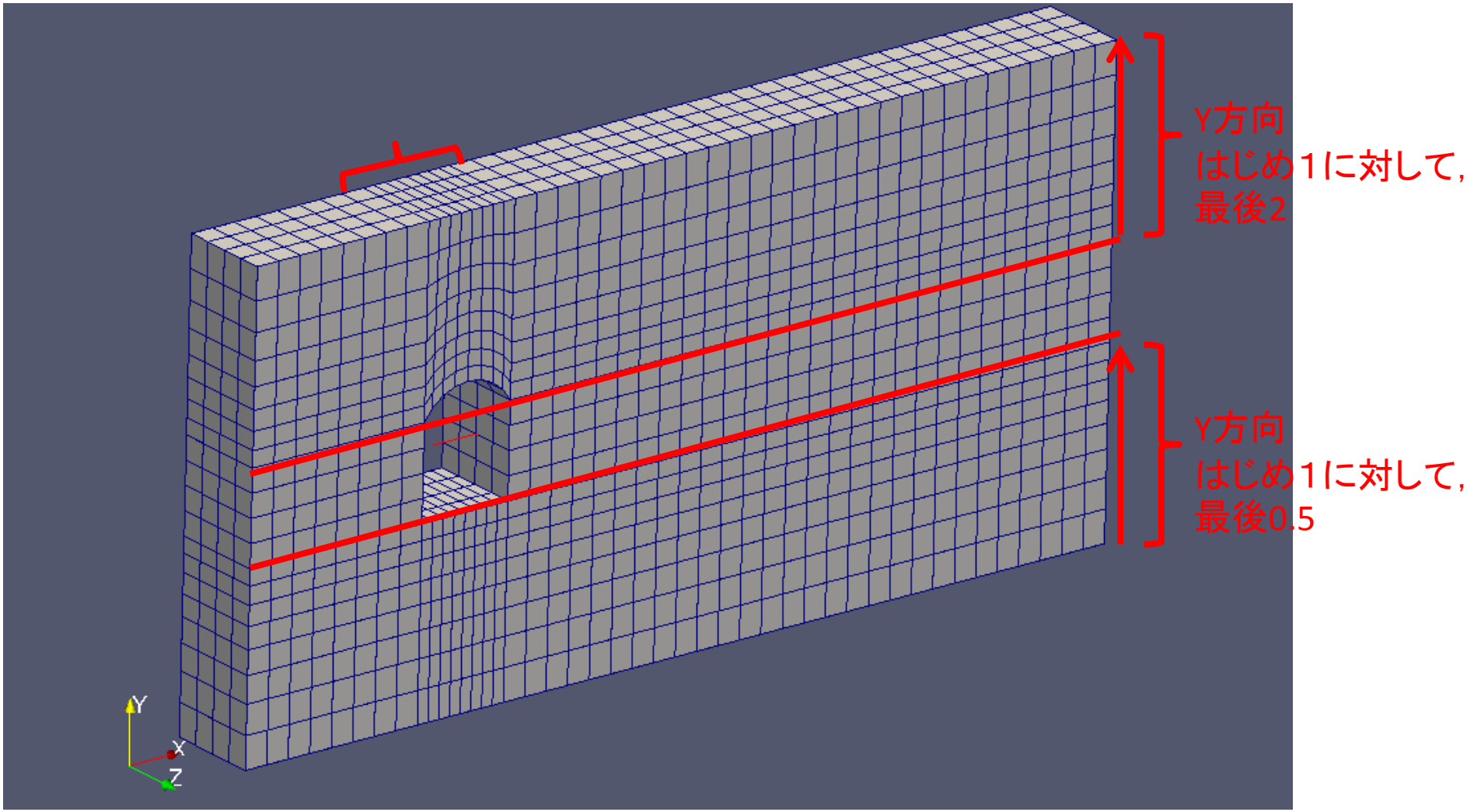
テスト ケース1 : bmTest05



テスト ケース1 : bmTest05

- bmTest04から，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

テスト ケース1 : bmTest05

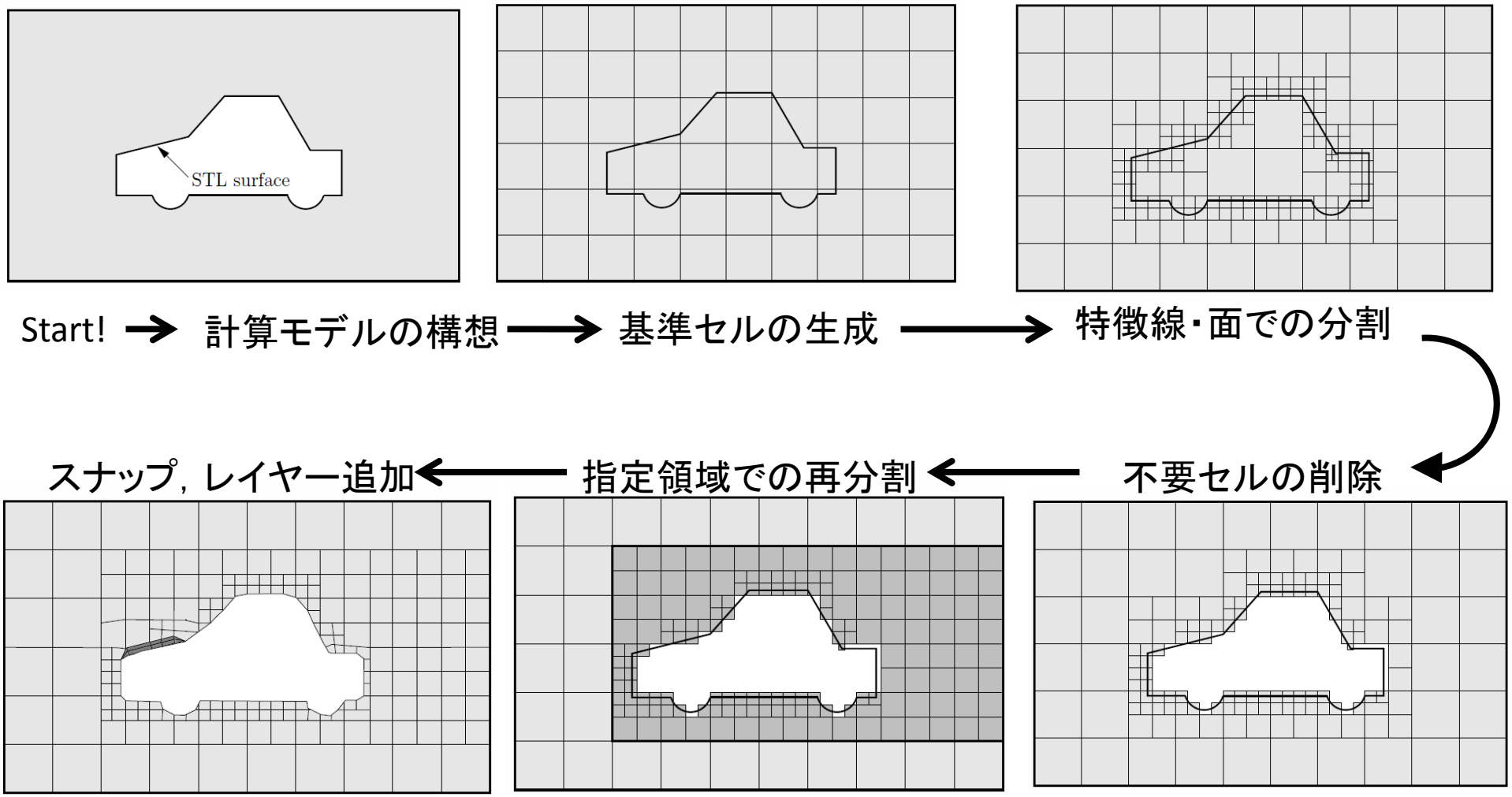


blockMesh演習

おわり

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)

5.4.1 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 基準となるメッシュの大きさ 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	

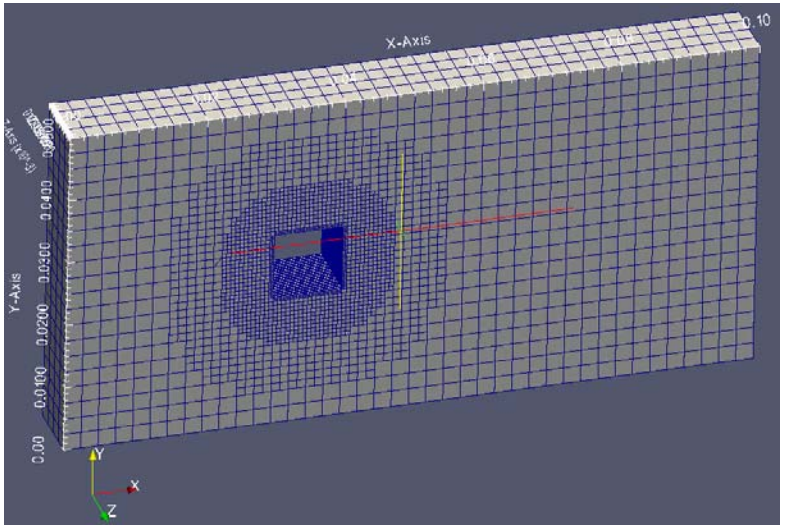
Table 5.7: Keywords at the top level of *snappyHexMeshDict*.

OpenFOAM UserGuide U-146

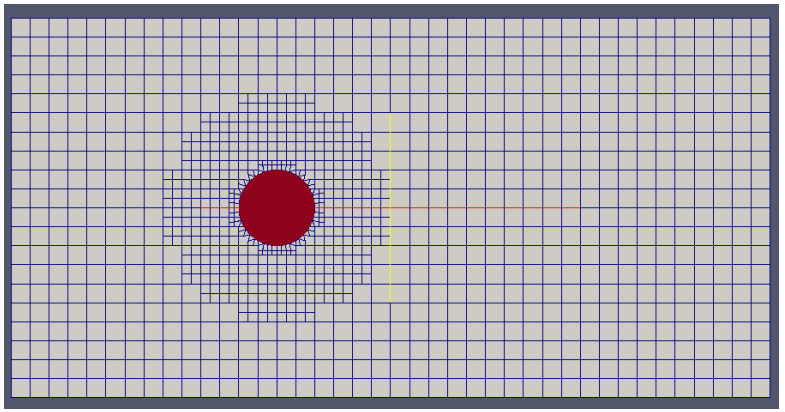
snappyHexMesh 実習

例題

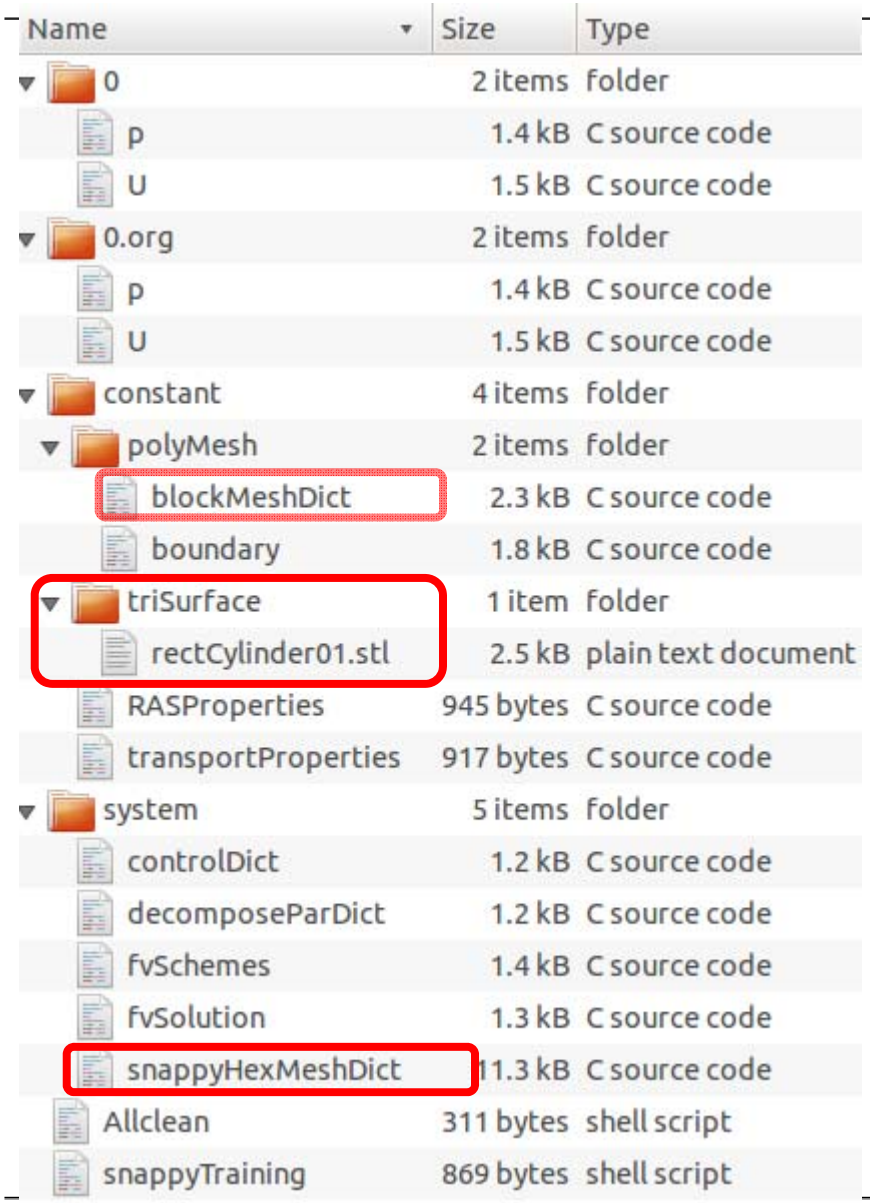
- snappyTestRect
  - 計算領域：直方体
  - 内部に四角柱を設置
  - 四角柱の周りにセルを生成



- snappyTestCyl
  - 計算領域：直方体
  - 内部に円柱を設置
  - 円柱の周りにセルを生成

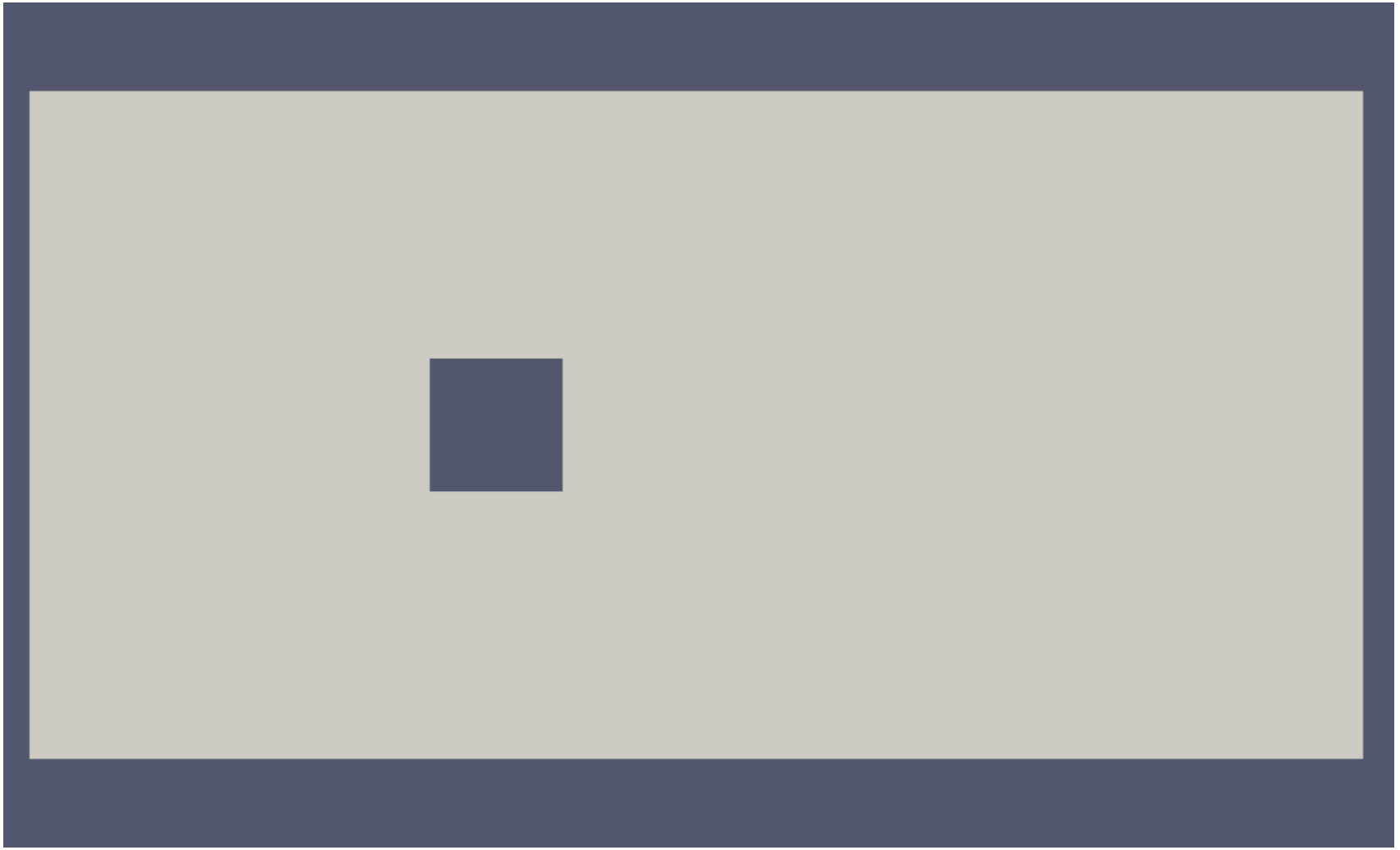


snappyTestRectケース：ファイル構成

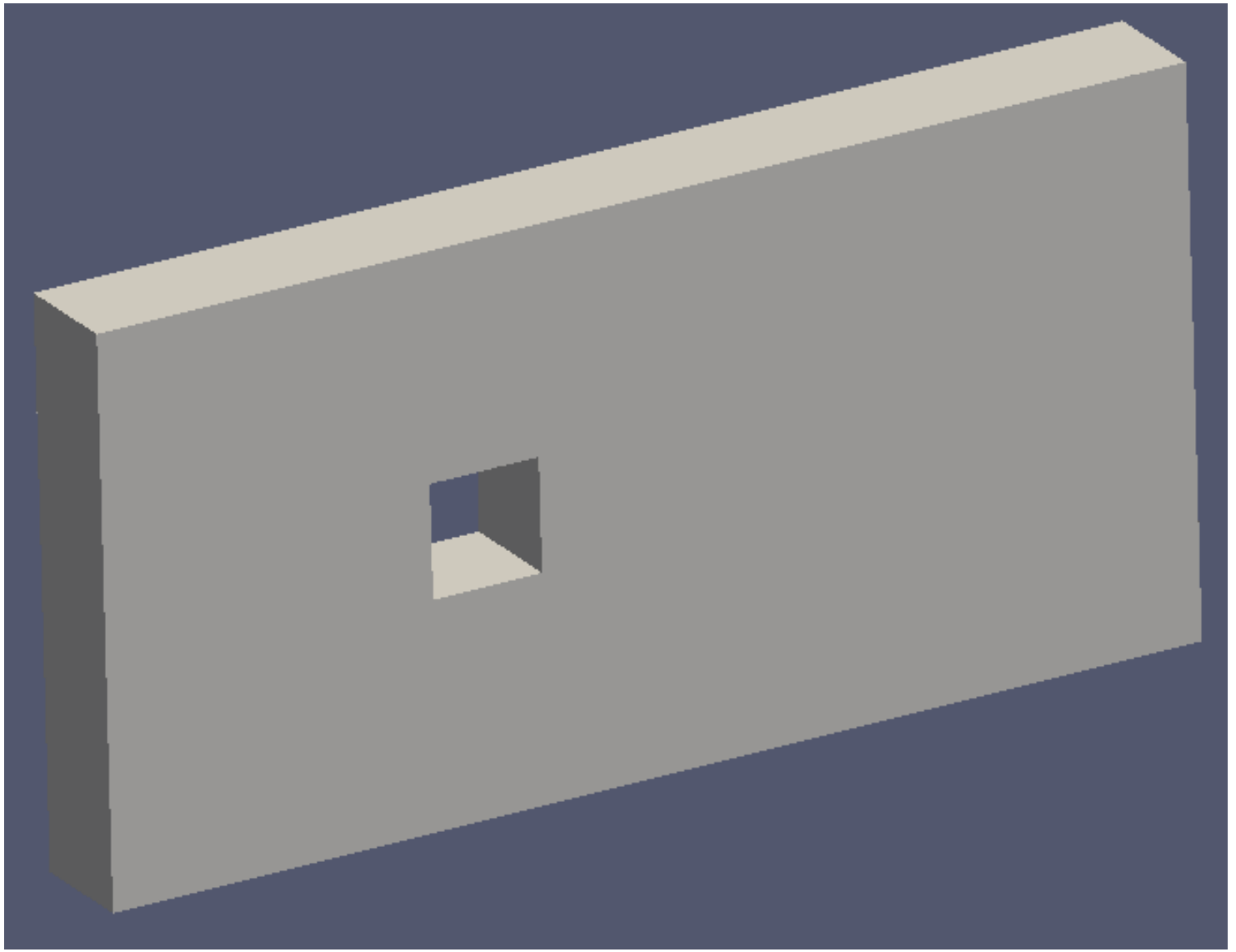


- 形状STLファイル
  - constant/triSurfaceに置く
  - rectCylinder01.stl
- 基準メッシュ設定
  - constant/polyMesh/blockMeshDict
- 詳細メッシュ設定
  - system/snappyHexMeshDict

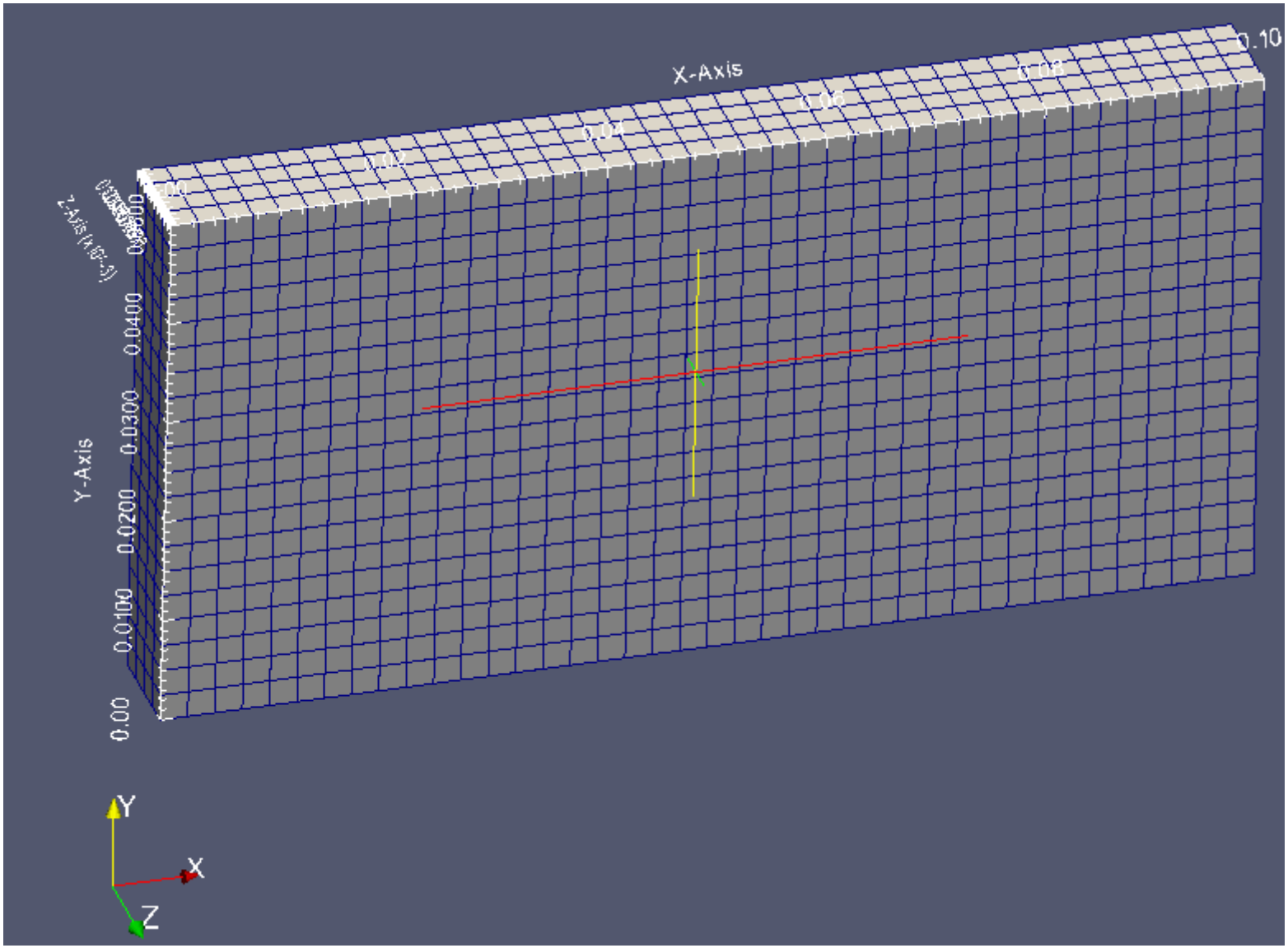
作成したい計算領域



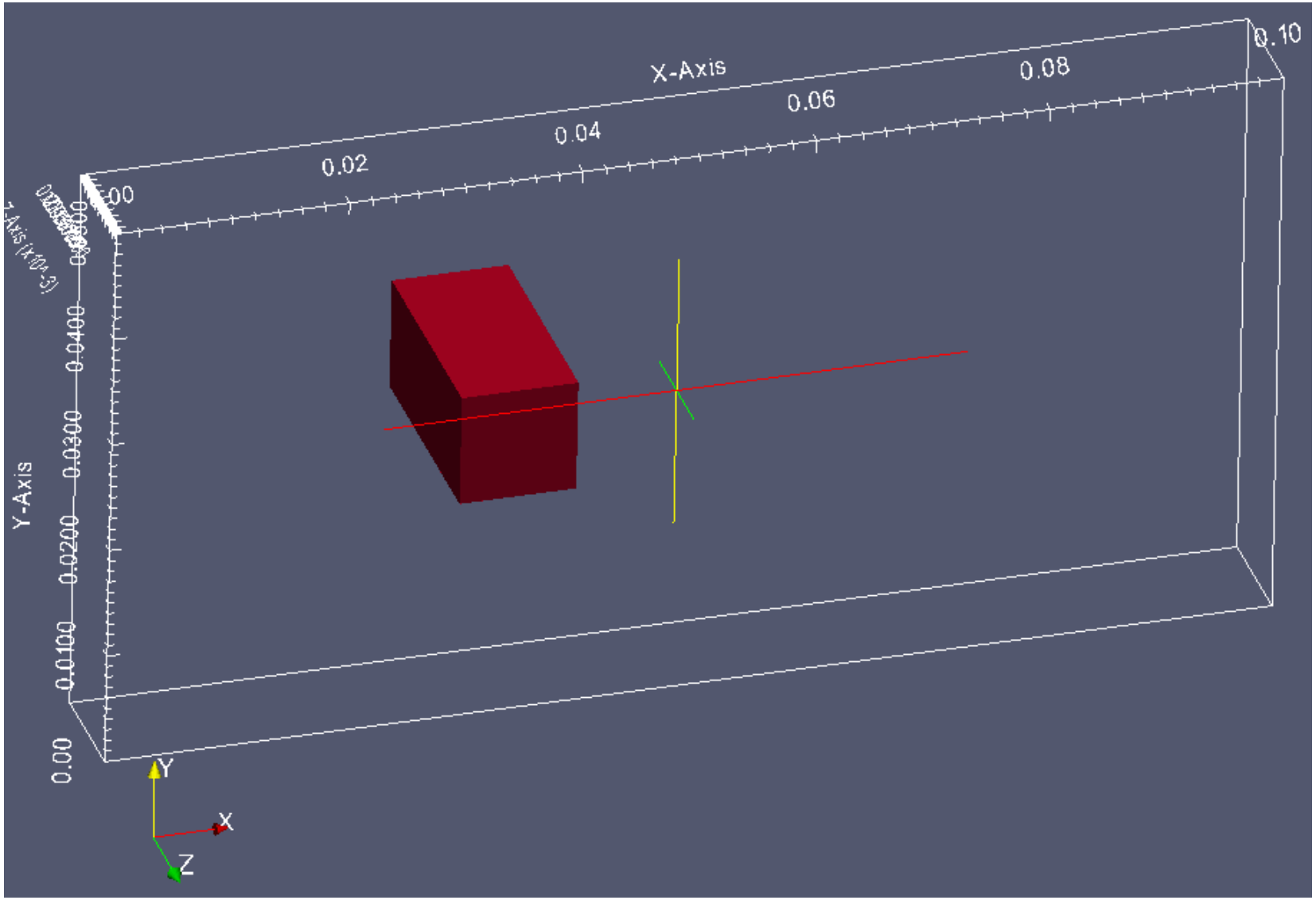
作成したい計算領域



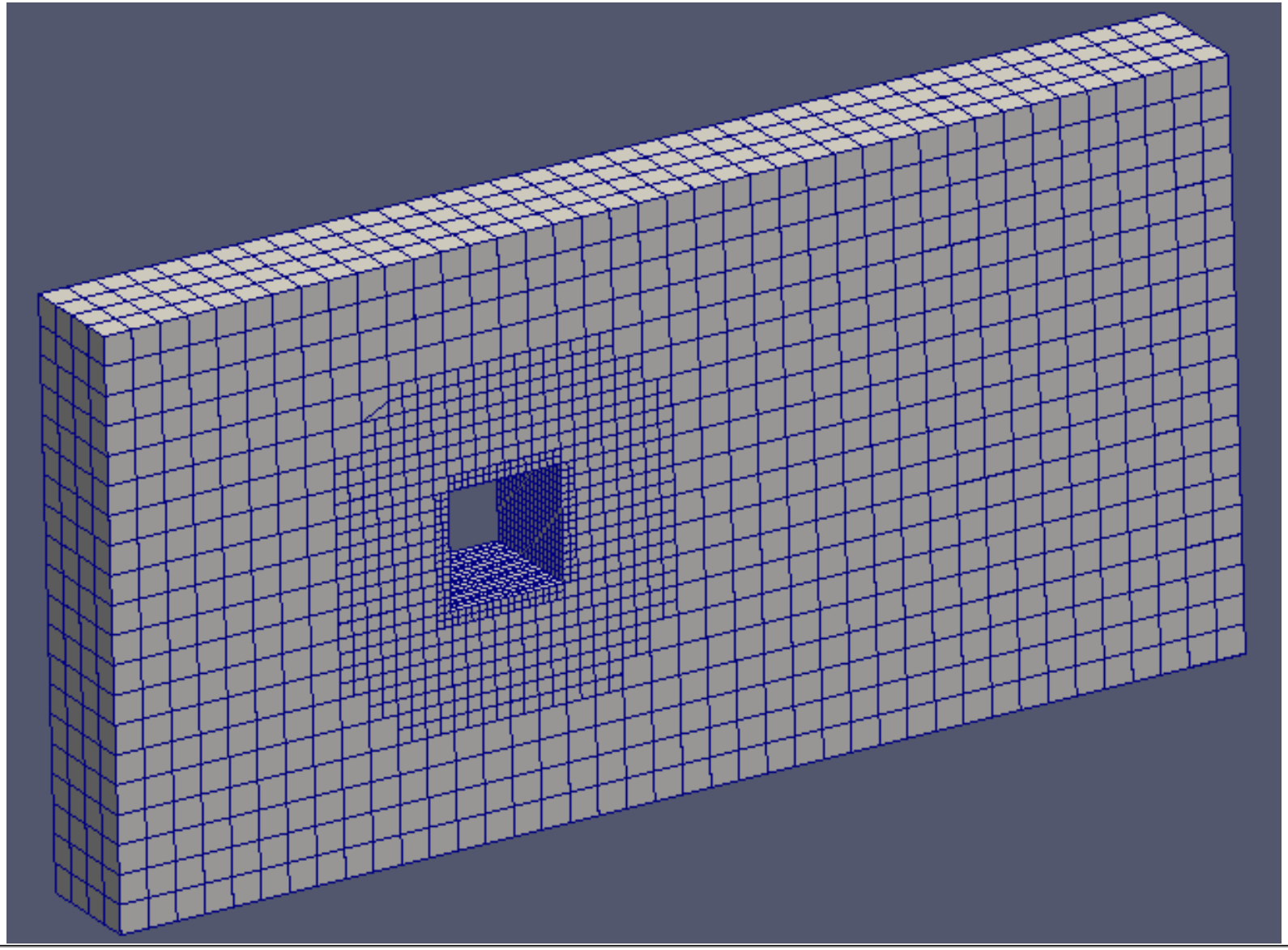
基準となるメッシュ（level 0）



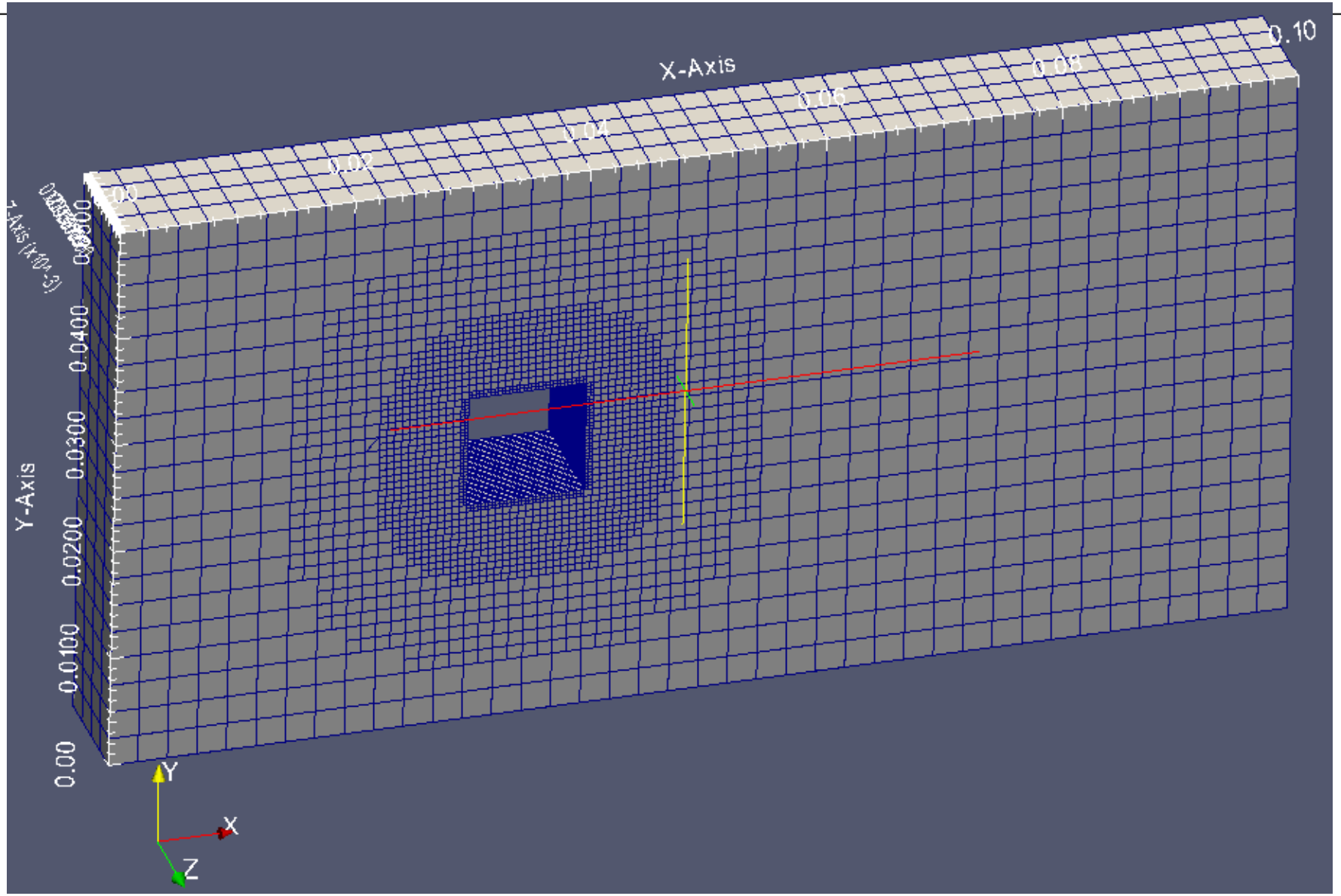
領域内に置いた角柱(STL)



完成図(例1)



完成図（例2）



実行手順

- 端末内で，ケースディレクトリsnappyTestRectに移動する
- ./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

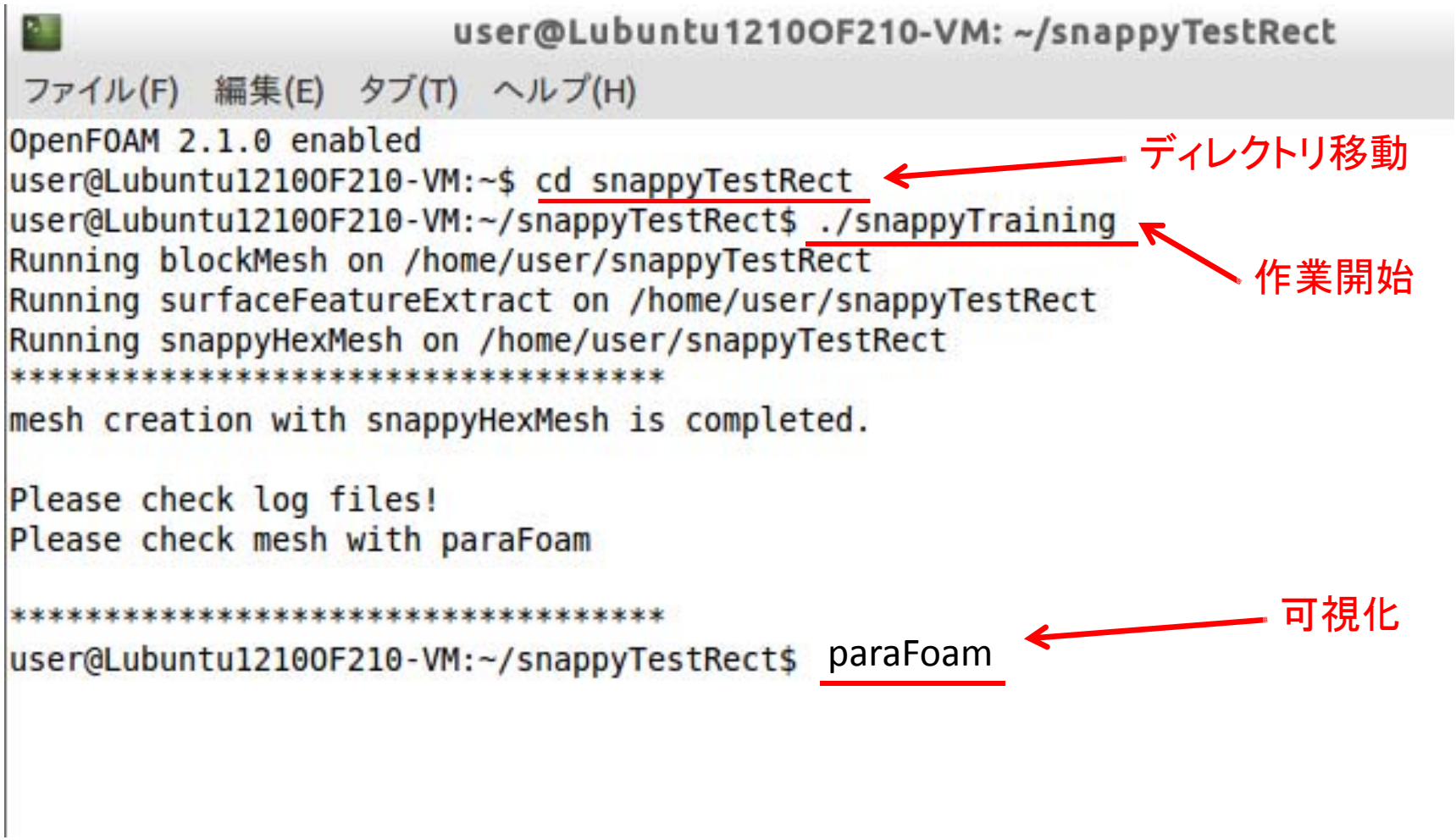
```
#!/bin/sh
cd ${0%/*} || exit 1    # run from this directory

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

runApplication blockMesh
runApplication surfaceFeatureExtract -includedAngle 150 -writeObj
constant/triSurface/rectCylinder01.stl rectCylinder01
mv log.surfaceFeatureExtract surfaceFeatureExtract_rectCylinder01.log

runApplication snappyHexMesh
# runApplication snappyHexMesh -overwrite

# force removal of fields generated by snappy
#¥rm -rf 0

#cp -rf 0.org 0

#runApplication `getApplication`
```

実行手順:詳細

- デスクトップのLX terminalをダブルクリックして，起動する。
- 例題ディレクトリ snappyTestRect に移動する。

  cd 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

作業

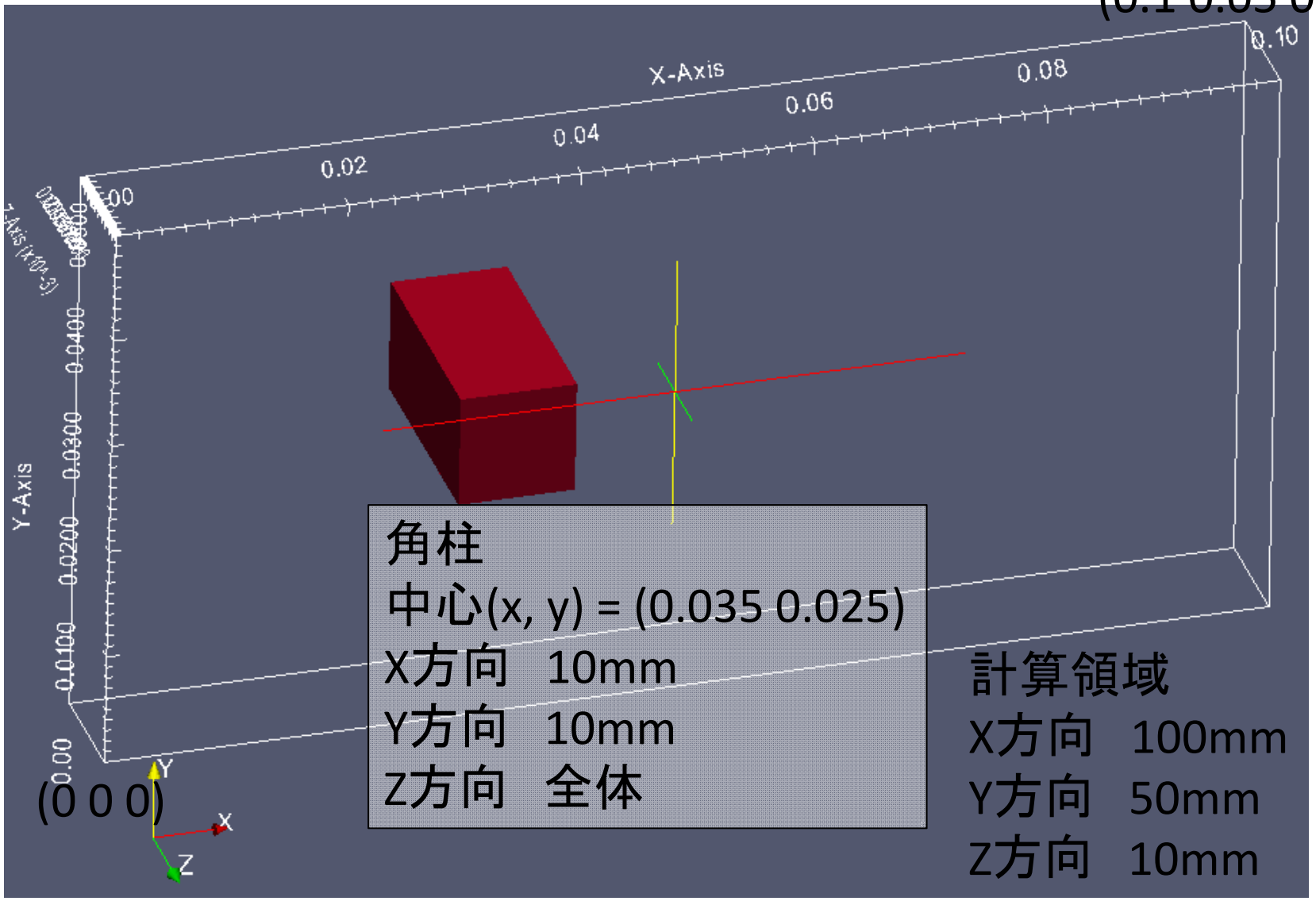

user@Lubuntu1210OF210-VM: ~/snappyTestRect
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
OpenFOAM 2.1.0 enabled
user@Lubuntu1210OF210-VM:~$ cd snappyTestRect
user@Lubuntu1210OF210-VM:~/snappyTestRect$ ./snappyTraining
Running blockMesh on /home/user/snappyTestRect
Running surfaceFeatureExtract on /home/user/snappyTestRect
Running snappyHexMesh on /home/user/snappyTestRect
****
mesh creation with snappyHexMesh is completed.
Please check log files!
Please check mesh with paraFoam
****
user@Lubuntu1210OF210-VM:~/snappyTestRect$ paraFoam
ディレクトリ移動
作業開始
可視化

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。

領域内に置いた角柱(STL)

~~(0.1 0.05 0.01)~~



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　(40 20 4) → 3200
  - 各面ごとを，一つの境界面(patch)とする

blockMeshDict

```
convertToMeters 1;

//- number of mesh
nX 40; nY 20; nZ 4;

//- minimum coordinate [m]
_xMin 0.0; _yMin 0.0; _zMin 0.0;

//- calc domain length: base length [m]
base 0.010;  // 10 mm

//- length of calc domain: coefficient
CLX 10.0;CLY 5.0;CLZ 1.0;

//- maximum coordinate [m]
_xMax #calc "$_xMin + $CLX*$base" ;
_yMax #calc "$_yMin + $CLY*$base" ;
_zMax #calc "$_zMin + $CLZ*$base" ;
```

```
vertices (
   ($_xMin $_yMin $_zMin)  // 0
   ($_xMax $_yMin $_zMin)  // 1
   ($_xMax $_yMax $_zMin)  // 2
   ($_xMin $_yMax $_zMin)  // 3

   ($_xMin $_yMin $_zMax)  // 4
   ($_xMax $_yMin $_zMax)  // 5
   ($_xMax $_yMax $_zMax)  // 6
   ($_xMin $_yMax $_zMax)  // 7
);

blocks(
   hex (0 1 2 3  4 5 6 7) ($nX $nY $nZ)
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

snappyHexMeshの方針

- 角柱の周囲に詳細なメッシュ

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
- 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 よく使う項目

- 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 (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;
}
```

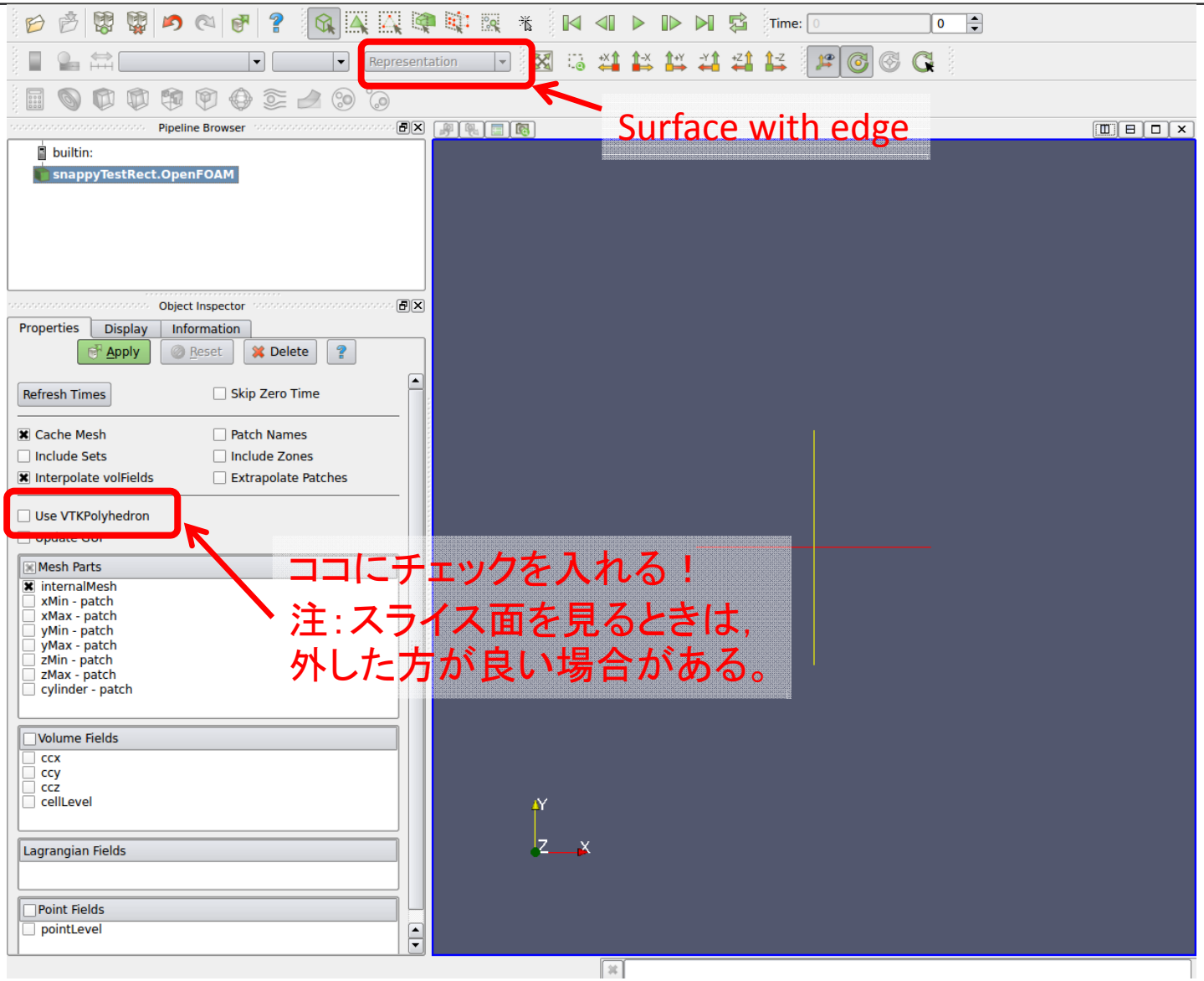
snappyHexMeshDict(meshQual. etc.)

```
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;
```

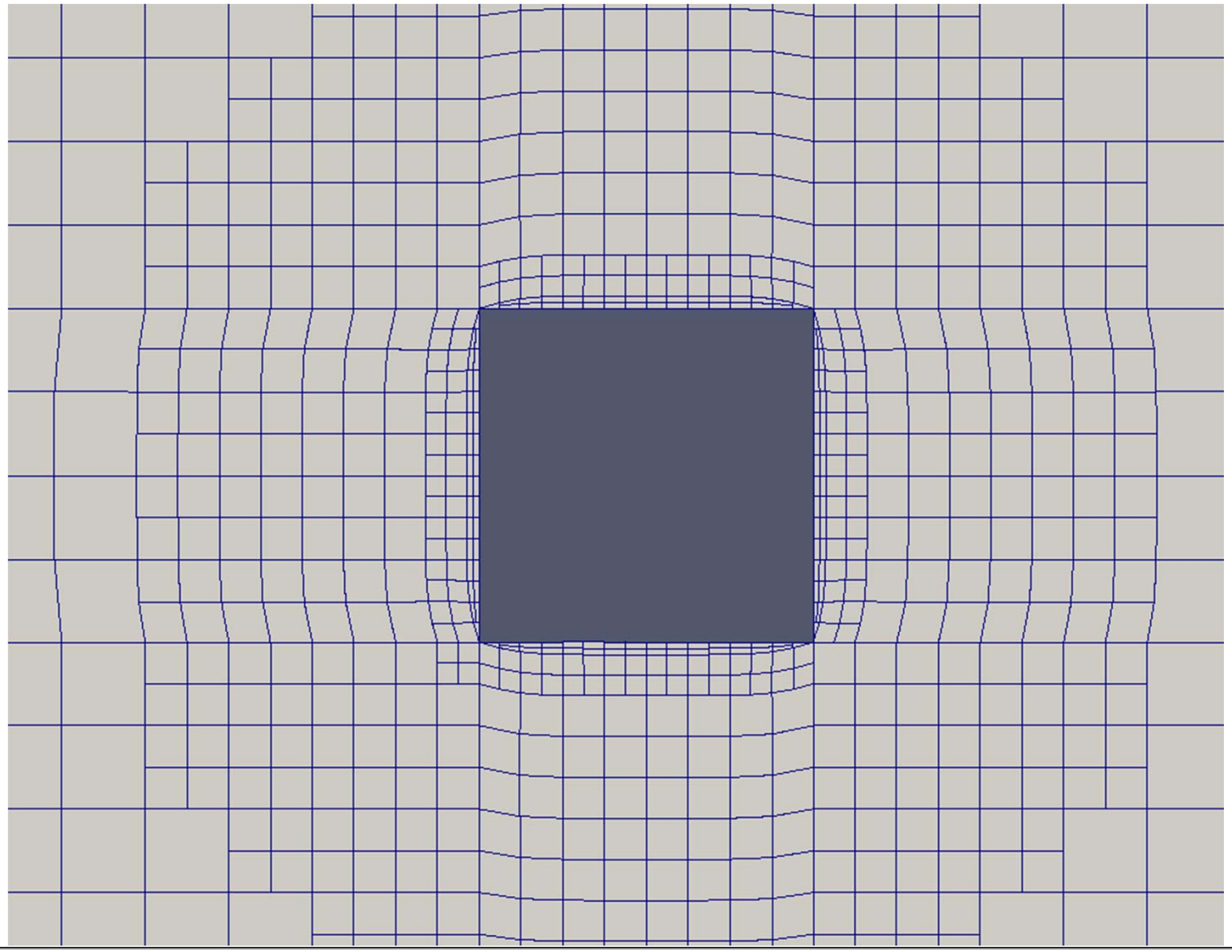
トライ

- 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
- ./Allclean の実行
- ./snappyTraining の実行
- paraFoam で変化の確認

メッシュをキレイに見るために

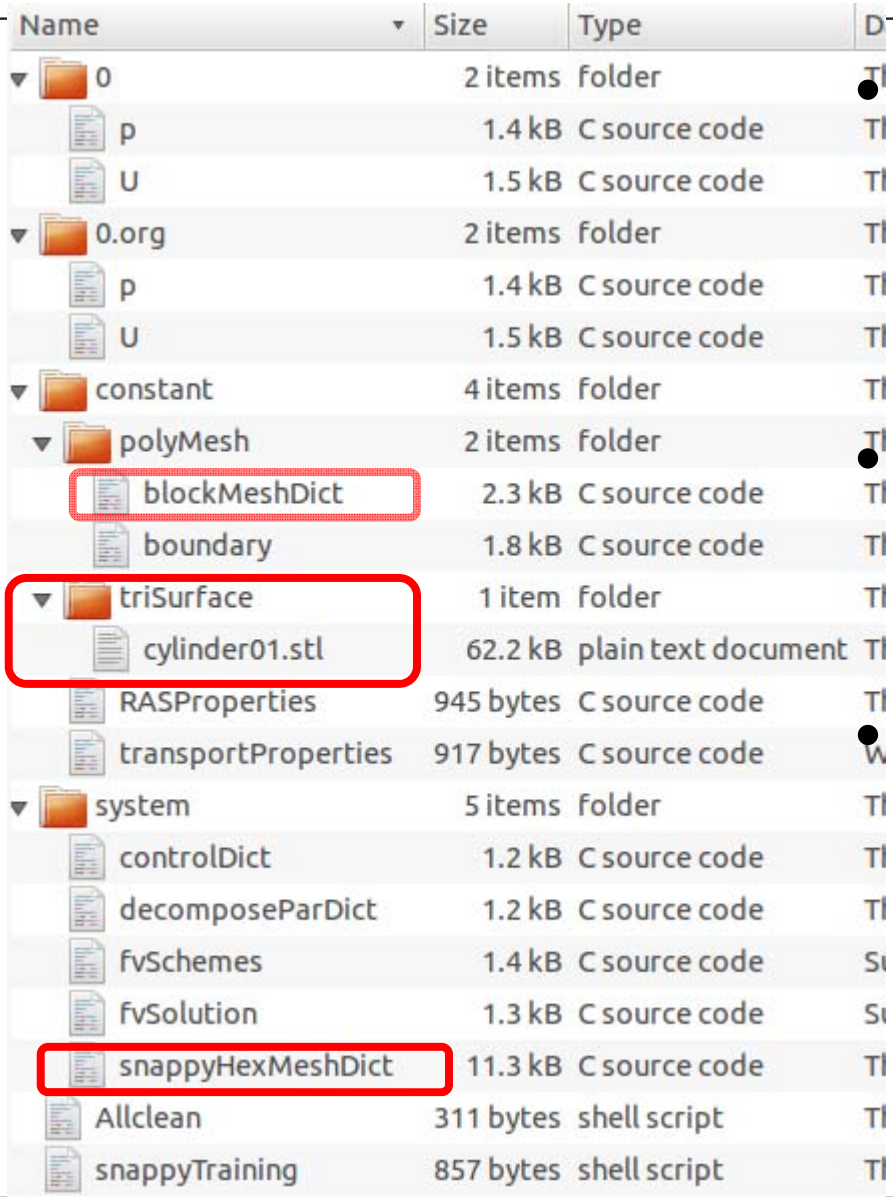


Layer追加例



障害物の形状を変える:円柱

snappyTestCylケース：ファイル構成



- 形状STLファイル
  - constant/triSurfaceに置く
  - cylinder01.stl
    - このファイルだけが異なる
- 基準メッシュ設定
  - constant/polyMesh/blockMeshDict
- 詳細メッシュ設定
  - system/snappyHexMeshDict

snappyTrainingスクリプトでも，STLファイル名が記載されているので，その名前を修正する。

実行手順:詳細

- デスクトップのLX terminalをダブルクリックして，起動する。
- 例題ディレクトリ snappyTestCyl に移動する。

  cd snappyTestCyl
- ./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

作業

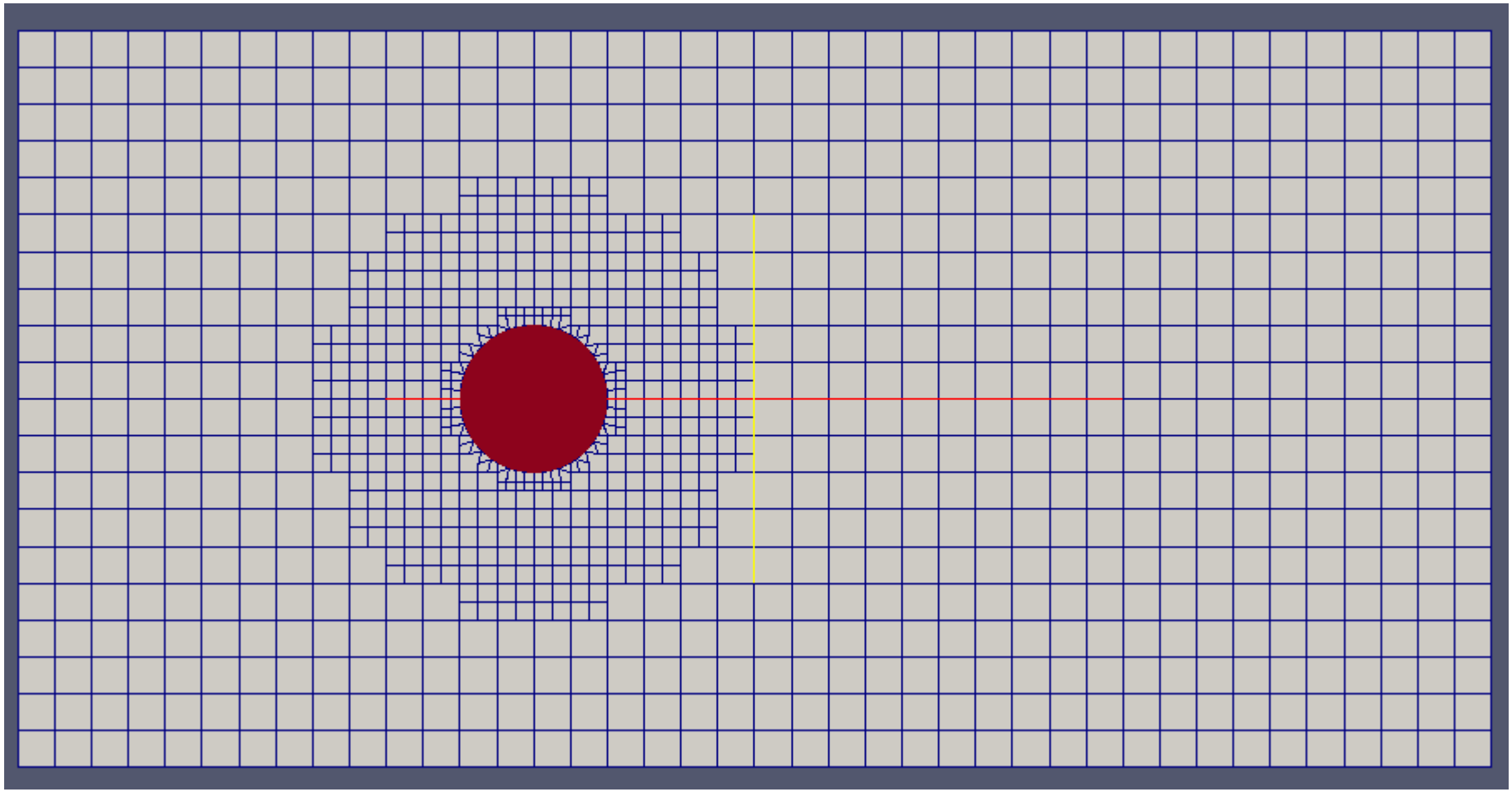
```
user@Lubuntu1210OF210-VM:~/snappyTestRect$
user@Lubuntu1210OF210-VM:~/snappyTestRect$ cd ..
user@Lubuntu1210OF210-VM:~$ cd snappyTestCyl
user@Lubuntu1210OF210-VM:~/snappyTestCyl$ ./snappyTraining
Running blockMesh on /home/user/snappyTestCyl
Running surfaceFeatureExtract on /home/user/snappyTestCyl
Running snappyHexMesh on /home/user/snappyTestCyl
**************************************
mesh creation with snappyHexMesh is completed.

Please check log files!
Please check mesh with paraFoam

**************************************
user@Lubuntu1210OF210-VM:~/snappyTestCyl$ paraFoam
```

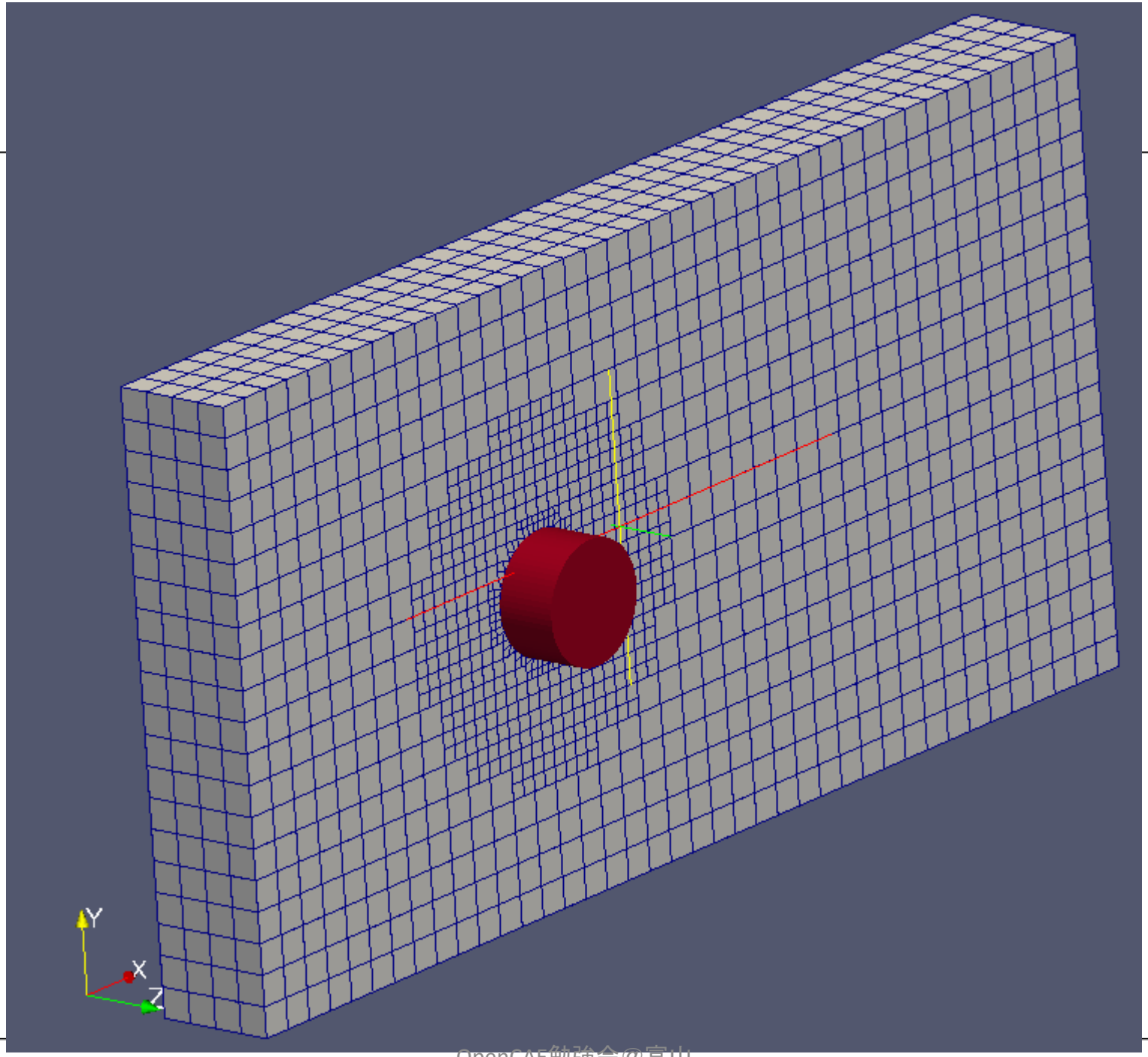
ディレクトリ移動

ディレクトリ移動

作業開始

可視化




Y
X
Z

snappyHexMesh演習

おわり