PimpleDyMFoam::propeller例題 モデル作成

中川慎二

2012/09/29

ソルバ

- PimpleDyMFoam
  - 非定常
  - 非圧縮性流れ
  - ダイナミックメッシュ
  - AMI
  - ニュートン流体
  - 乱流
  - PIMPLE (SIMPLE とPISO の融合) アルゴリズム

例題：

/tutorials/incompressible/pimpleDyMFoam/propeller

- 水中でプロペラを回転させる
- 円柱型の計算領域
- 計算領域内部に，円柱型の回転領域

ファイル構造

フォルダ	ファイル	備考
0	U	境界条件・初期条件（速度）
	p	境界条件・初期条件（圧力）
	k	境界条件・初期条件（乱流エネルギｋ）
	epsilon	境界条件・初期条件（エネルギ散逸率ε）
	nut	境界条件・初期条件（渦動粘度）
0.org		フォルダ0の元
constant	transportProperties	流体物性値設定ファイル
	turbulenceProperties	乱流のモデル化設定（RAS,LESなど）
	RASProperties	RAS乱流モデル設定（k-ε，など）
	dynamicMeshDict	メッシュの動き設定
constant/polyMesh	blockMeshDict	blockMesh用入力ディクショナリ
	boundary	境界条件
cnstant/triSurface	innerCylinder.obj	形状データ：メッシュ細分化領域
	innerCylinderSmall.obj	形状データ：AMI面
	outerCylinder.obj	形状データ：解析領域全体
	propellerStem1.obj	形状データ：軸（プロペラ部）
	propellerStem2.obj	形状データ：軸
	propellerSem3.obj	形状データ：軸
	propellerTip.obj	形状データ：プロペラ
system	controlDict	計算制御設定ファイル
	decomposePar	並列計算用設定ファイル
	fvSchemes	有限体積法での離散化設定ファイル
	fvsolution	行列解析、トレランス、アルゴリズム設定ファイル
	changeDictionaryDict	境界条件修正用ディクショナリ(AMI面の情報)
	createAMIFaces.topoSetDict	topoSet用ディクショナリ(AMI面の生成)
	createInletOutletSets.topoSetDict	topoSet用ディクショナリ(流入・流出境界面の生成)
	removeRedundantZones.topoSetDict	topoSet用ディクショナリ(不要領域の削除)
	createPatchDict	createPatch用ディクショナリ(setからpatchを生成)
	snappyHexMeshDict	snappyHexMesh用ディクショナリ

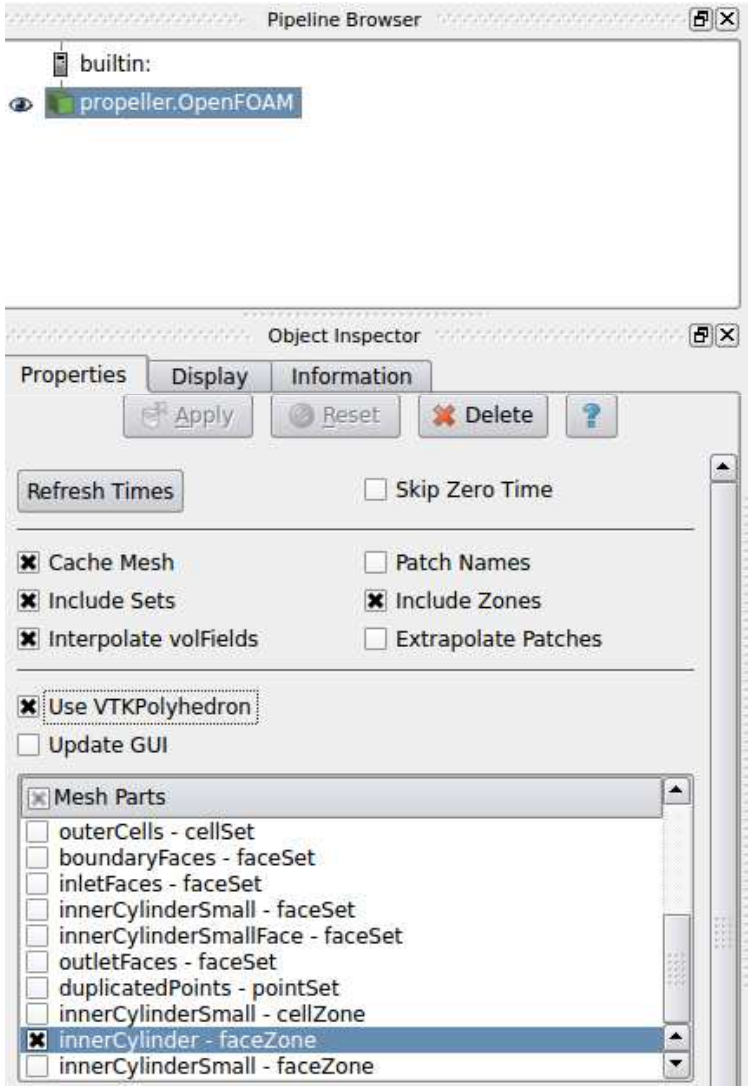
作業手順

- メッシュ生成
  - ブロックメッシュ生成
  - 形状に合わせて修正（snappyHexMesh）
  - 回転領域でのAMI面生成
- 条件設定
  - 初期・境界条件，離散化設定，行列解法
  - 物性，乱流モデル
  - 回転条件

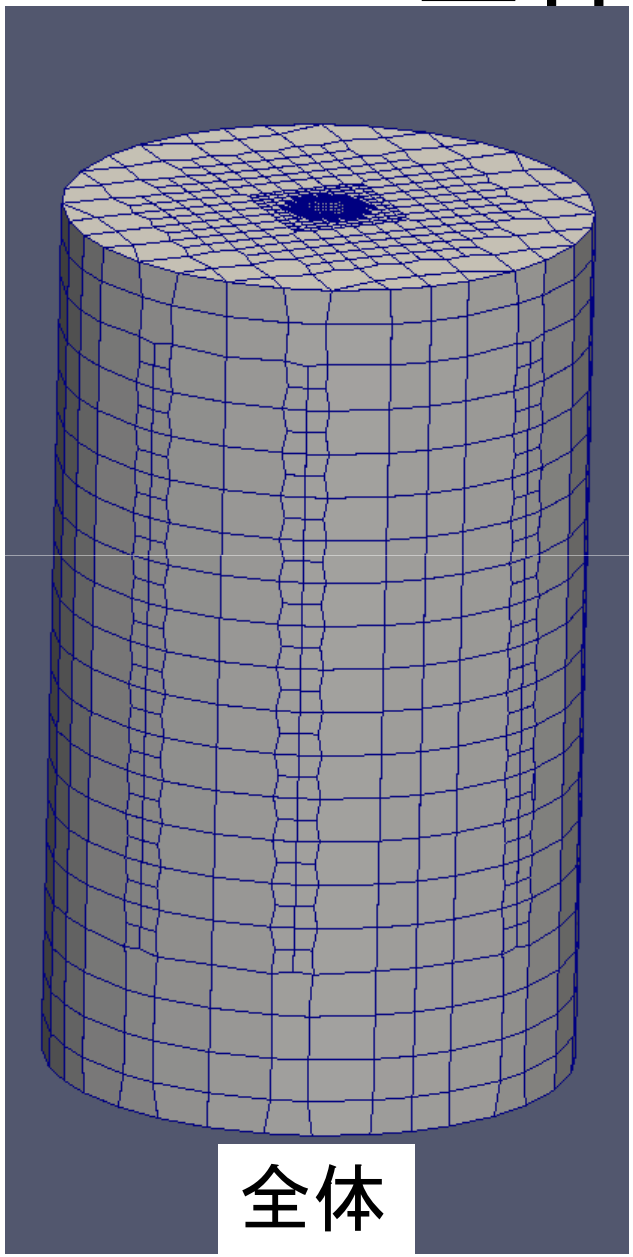
メッシュ確認時のparaview

Object Inspectorで次のチェックボックスをON

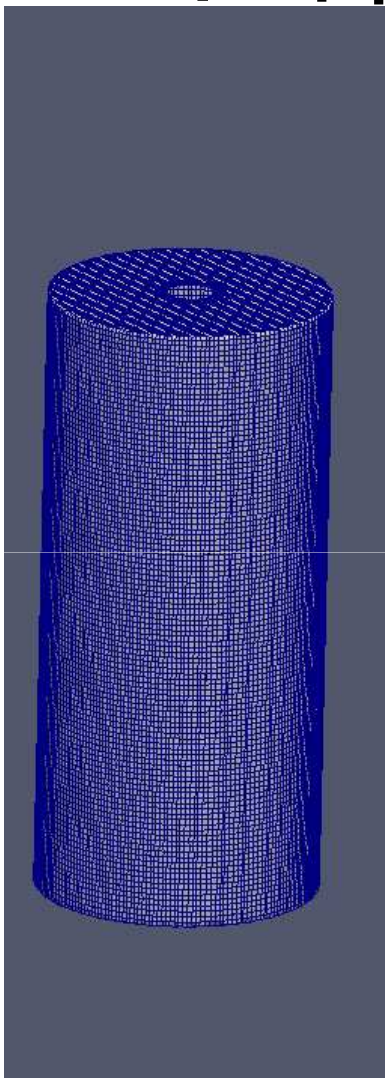
- Include Sets
  - faceSet, cellSet表示
- Include Zones
  - faceZone, cellZone表示
- Use VTKPolyhedron
  - 変形セルを正しく表示



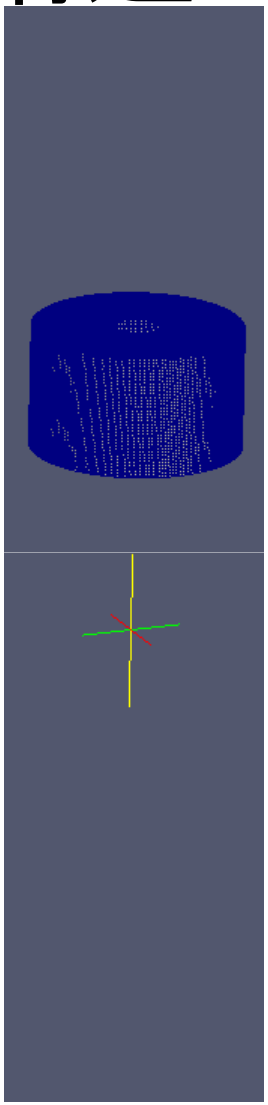
全体 から 内部 構造



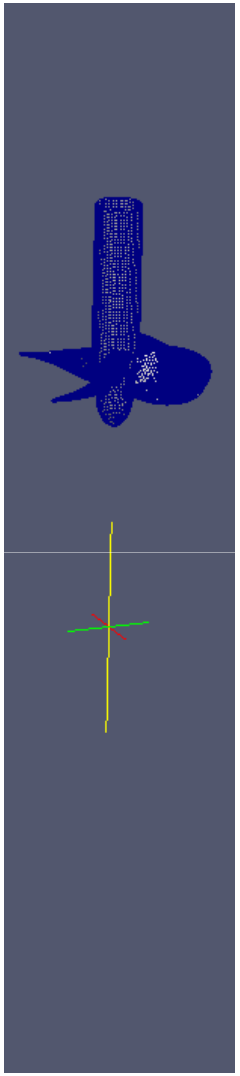
全体



innerCylinder
細分化領域



innerCylinderSmall
回転領域



プロペラ
軸

- # blockMeshの実行　snappyHexMeshに必要な大枠，レベル0のメッシュを生成

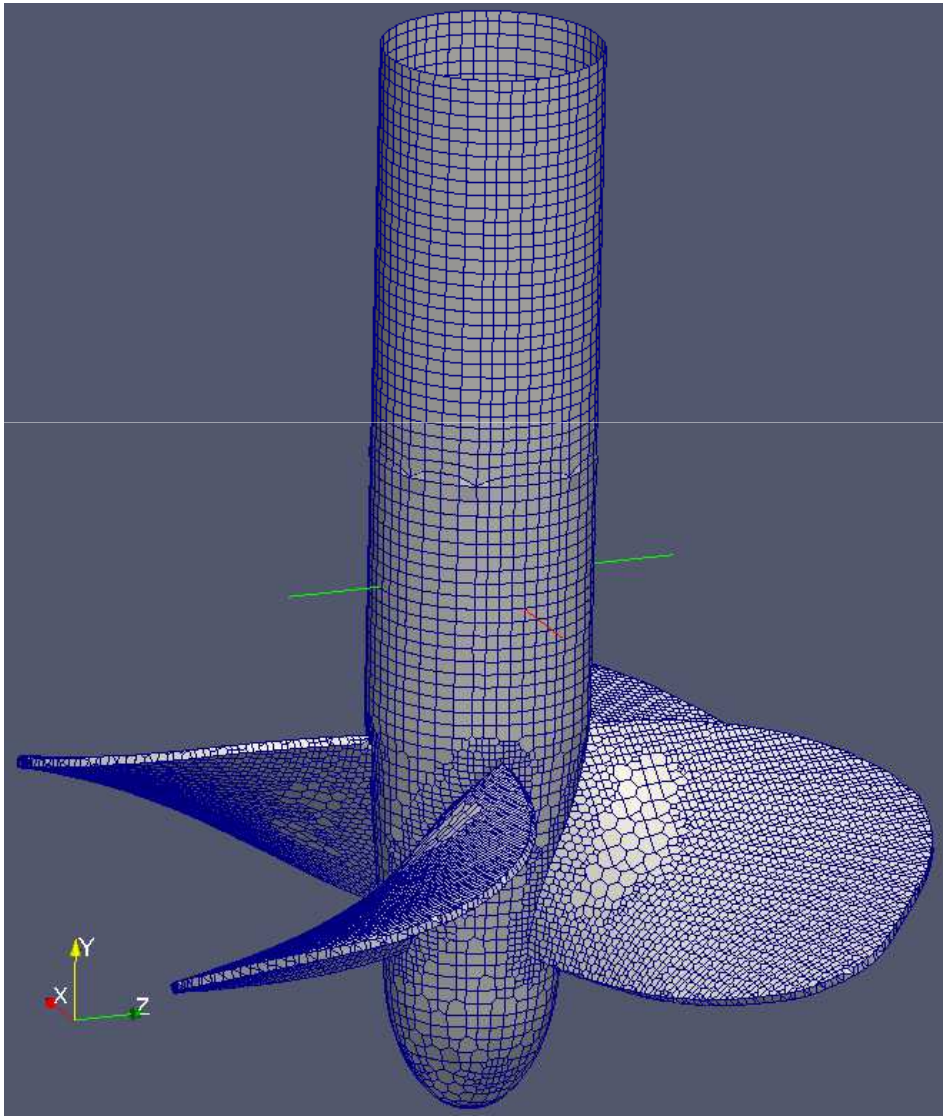
blockMesh

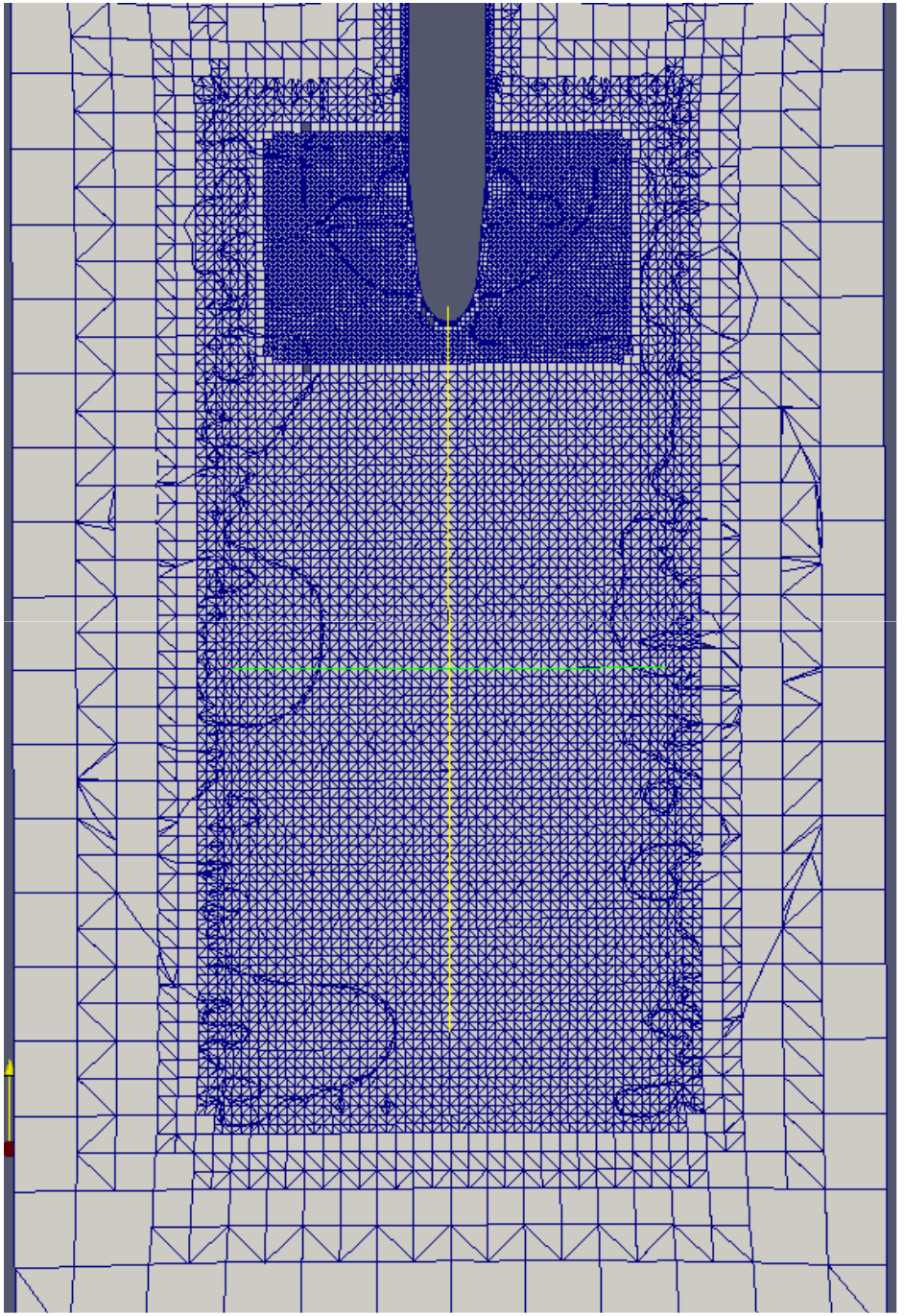
- /constant/triSurfaceディレクトリの3Dモデル（ファイル名を$sとする）から，特徴線の抽出

surfaceFeatureExtract -includedAngle 150 -minElem 10 constant/triSurface/$s.obj $s

- snappyHexMeshの実行

snappyHexMesh -overwrite


Y
X
Z



表面の細分化

```
innerCylinder
    {
      level (2 3); cellZone   innerCylinder;   faceZone    innerCylinder;
       cellZoneInside  inside;
    }
innerCylinderSmall
    {
      level  (4 4); cellZone  innerCylinderSmall;   faceZone innerCylinderSmall;
       cellZoneInside  inside;
    }
outerCylinder   {   level      (0 0);   }
propellerTip   {   level      (4 5);    }
propellerStem1   {   level      (4 4);   }  他も同じ
```

topoSetDictの構造

ヘッダとactions記述で構成される。

ヘッダは，一般的なOpenFOAM関連ファイルと同じ。

```
actions(
        { 一つ目のアクション(処理)の詳細 }
        { 二つ目のアクション(処理)の詳細 }
)
```

actionsの詳細

- name(その動作の名前：任意につけて良い)
- type（動作によってつくったものの種類）
- action（処理(新規作成，削除など)）

removeRedundantZones.topoSetDict

```
actions
(
    {
        name    innerCylinder;
        type    cellZoneSet;
        action  remove;
    }
    {
        name    innerCylinderSmall;
        type    cellZoneSet;
        action  remove;
    }
);
```

createInletOutletSets.topoSetDict

(1)boundaryFacesというfaceSetの新規作成
- boundaryFacesという名前のfaceSetを新規作成(new)します。
- パッチのouterCylinderからfaceを取り出します(patchToFace)。

(2)outletFacesというfaceSetの新規作成
- outletFacesという名前のfaceSetを新規作成(new)します。
- 先に作ったfaceのboundaryFacesからfaceを取り出します(faceToFace)。

(3)inletFacesというfaceSetの新規作成
- inletFacesという名前のfaceSetを新規作成(new)します。
- 先に作ったfaceのboundaryFacesからfaceを取り出します(faceToFace)。

- 以上の作業で，boundaryFaces，outletFaces，inletFacesには，すべて同じ情報（outerCylinderパッチと同じ面が1764個）入っている。

(4)outletFaces というfaceSetの一部抜き出し
- 先に作ったoutletFacesから，faceSetを一部抜き出し(subset)します。
- sourceInfoで記述した情報に垂直な面(normalToFace)だけを，outletFacesに入れます。
- ここでは，-y方向に垂直な面を許容範囲が角度のコサインが0.3として，採用します。

(5)inletFacesというfaceSetの一部抜き出し
- 先に作つinletFacesから，faceSetを一部抜き出し(subset)します。
- sourceInfoで記述した情報に垂直な面(normalToFace)だけを，outletFacesに入れます。
- ここでは，+y方向に垂直な面を許容範囲が角度のコサインが0.3として，採用します。

createAMIFaces.topoSetDict

(1)innerCylinderSmallというcellSetの新規作成
- innerCylinderSmallという名前のcellSetを新規作成(new)します。
- Adding cells with centre within cylinder, with p1 = (0 -0.08 0), p2 = (0 0.06 0) and radius = 0.12。

(2)outerCellsというcellSetの新規作成
- outletCellsという名前のcellSetを新規作成(new)します。
- 先に作ったcellSetのinnnerCylinderSmallから全てのcellを取り出します(cellToCell)。

(3)outerCellsというcellSetの変更
- 先に作ったouterCellsを逆転させ，outerCellsに入っていなかったcellだけを格納します。

(4)innerCylinderSmall というcellZoneSetの新規作成
- 先に作ったinnerCylinderSmallというcellSetから，cellZoneを作成(setToCellZone)します。

createAMIFaces.topoSetDict

(5)innerCylinderSmallFaceというfaceSetの新規作成
- 先に作ったinnerCylinderSmall というcellSetから，全てのfaceSetを入れます。(cellToFace)

(6)innerCylinderSmallFaceというfaceSetの一部抜き出し
- 先に作ったinnerCylinderSmallFace というcellSetから，cellSetのouterCellsにあるfaceSetを抜き出します。
- これで，innerCylinderSmallとouterCellsとの間にあるfaceだけをfaceSetに入れたことになります。

(7)innerCylinderSmall というfaceZoneSetの新規作成
- 先に作ったinnerCylinderSmallFaceというfaceSetと，先に作ったinnerCylinderSmallというcellSet（回転部）から，faceZoneSetを作成(setToFaceZone)します。

- このfaceZoneのinnerCylinderSmallから，後ほど，createBafflesコマンドを使って，AMI1とAMI2を生成することになります。

createPatchDict

```
patches
(
    {
        name inlet;
        patchInfo { type patch; }
        constructFrom set;
        set inletFaces;
    }
    {
        name outlet;
        patchInfo { type patch; }
        constructFrom set;
        set outletFaces;
    }
);
```

changeDictionary

systemの中にある"changeDictionaryDict"の情報を元にして，dictionaryのentriesを変更する。
具体的には，boundaryファイルに，AMI1とAMI2の情報を追加する。

<faceZone>innerCylinderSmallをbaffle boundaryに変換する。

AMI1をmasterPatch，AMI2を slavePatchにする。

baffleは，表裏に条件を設定できる面。

- createBaffles -internalFacesOnly -overwrite innerCylinderSmall '(AMI1 AMI2)'

一つの面であるbaffleを，表と裏で別々のものに分離する。→ AMI1とAMI2が独立した2つの面になる。

- mergeOrSplitBaffles -split -overwrite