OpenFOAM

1 OpenFOAM の概要

OpenFOAM とは GNU のもとで公開されているオープンソースの CFD ツールボックスである。ただし、厳密には有限体積法を中心とする偏微分方程式ソルバー開発のツールボックスと言うべきであり、CFD 以外でも利用することは可能である。計算環境は Linux を想定しており、独自のディレクトリ構造の下で諸計算を実行する。ソースコードのプログラミング言語は C++ であり、もしも機能を拡張したいのであれば、基本的には製作者も C++ で実装しなければならない。しかしながら標準の機能はかなり充実していること、並びに計算条件等の設定は別のテキストファイルで指定することより、解析者がソースコードを読み解くことは滅多にない。OpenFOAM の習得と言えば、ほとんどが計算条件等に関するテキストファイルのフォーマット理解のことを指す。

OpenFOAM は標準でプリプロセスやソルバー、並びにポストプロセスに関する機能を備えている。プリプロセスの中でよく用いるものに CAD の特徴領域把握やメッシュ作成がある。基本的に CAD ファイルには **STL ファイル**を用いる。STL 幾何構造から特徴線を抽出し、メッシュ生成時に役立てる。OpenFOAM のメッシュ生成は構造格子と非構造格子に対応しており、基本的に物体境界付近で非構造格子となる。実は OpenFOAM の計算格子品質は商用のものよりも劣っているが、商用ソフトで造られた計算格子をインポートすることも可能なので、必要となれば別途メッシュ生成ソフトを用意してもよい。メッシュ生成については 2 章で触れる。

OpenFOAM は有名なソルバーを一通り揃えているため、解析者がソルバー開発しなければならないことは稀である。ただし、ソルバー毎に初期条件等の設定ファイルは変わってくることには注意しなければならない。設定ファイルに不足や誤りがあれば、当然ながら解析は実行できない。OpenFOAM はテキストファイルで条件を設定するだけに、このミスに気付きにくいことがある。そのような時は、同様の解析をしているチュートリアルのファイルと見比べるとよい。ソルバーや解析条件の設定については 3 章で述べる。

OpenFOAM では **paraFoam** というコマンドが用意されており、それを実行すると Paraview が起動され、可視化が可能になる。非常に便利な一方で、リモート接続した PC 上で解析を実行し、ターミナルでの操作しかできない場合などでは扱いにくい。また、FieldView や Ensight といった他の可視化ソフトを使いたい場合もある。4 章では可視化ファイルフォーマットの抽出方法や数値データの抽出方法について議論する。

2 メッシュの作成

本章では **snappyHexMesh** という OpenFOAM 標準搭載のメッシュ生成機能を紹介する。OpenFOAM でメッシュ生成するとき、基本的に以下の手順を踏む。

計算格子生成手順

1. STL ファイルの準備。
2. blockMesh。解析領域を包む基本領域の設定。CAD を無視したメッシュの生成。
3. surfaceFeatureExtract。CAD データから特徴線を抽出。
4. snappyHexMesh。2.-3. を加味し、CAD を加味した高度なメッシュの生成。
5. checkMesh。メッシュ品質の診断。
6. createPatch。不要な境界の除去。
7. renumberMesh。各メッシュの番号を再割り振り。

blockMesh からやり直したい場合は、コマンド **foamCleanPolyMesh** を実行する。

2.1 STL ファイルの準備

CAD ファイルフォーマットにはいくつか選択肢があるが、多くの場合 STL ファイルが用いられている。OpenFOAM で利用可能な STL ファイルの特徴を以下に列挙する (なお、以下の項目が利用可能な STL ファイルの必要条件なのかは不明である。あくまで解析に成功したときの特徴だと思ってほしい)。

- ASCII フォーマットで記入する。
- 長さの単位には m を利用する。
- 同一の境界条件やメッシュ条件を定める面毎に solid を定義する。従って 1 つの STL ファイル内に複数の solid が定義されることになる。solid 名は境界定義時などで利用するため、解析条件設定時に把握しやすいものにする。

OpenFOAM は面の分割に **FreeCAD** の利用を紹介している。

FreeCAD による STL ファイルの準備

1. FreeCAD で面分割し、面毎に形状ファイルをエクスポートする (.ast 拡張子を指定)。
2. ast ファイルを開き、solid 名の Mesh を適当な名前に変更する。
3. 各 ast ファイルを一つに統合する。例えば、「cat *.ast > geomety.stl」を実行する。
4. 形状の長さスケールを m に変換する (FreeCAD は基本 mm)。「surfaceConvert -scale 0.001 mm.stl m.stl」を実行する。

作成した CAD ファイルは**./constant/triSurface** に保存する。

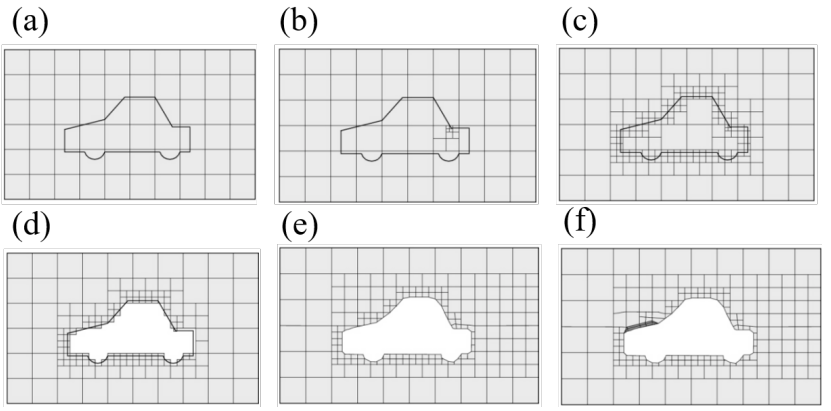

図 2.1　メッシュ生成プロセス。(a)blockMesh による基本メッシュの生成 (b) 特徴線付近の細分化 (c) 境界面の細分化 (d) 領域外の除去 (e) 界面適合 (f) レイヤーメッシュの生成

2.2　メッシュの作成

図 2.1 は OpenFOAM のメッシュ生成手順を示している。OpenFOAM は **blockMesh** というコマンドで、はじめに解析領域全体を包むようにメッシュを生成する (図 2(a))。このメッシュのサイズは大きくてもよく、結果的に最大計算格子サイズに対応する。blockMesh で基本メッシュを作成したのち、特徴線近傍の計算格子を細分化する。細分化には八分木データ構造を利用しており、図 2(b) のように細分化される。特徴線近傍の細分化は **surfaceFeatureExtract** というコマンドで行われる。次に境界面近傍の細分化を行う (図 2(c))。図 2(c) から以後は **snappyHexMesh** が担当する。後述するように、境界面のメッシュ再分割設定や解析領域の指定、界面適合、並びにレイヤーメッシュの設定を行う。図 2(d) では解析領域を指定している。例えば車体の外部流れを解析したい場合、車外の座標を一つ適当に指定する。そうすれば基本メッシュと STL ファイルの面情報より、車内のメッシュを取り除けばよいと分かる。逆に車内の解析をしたい場合は、車内の座標を適当に一つ指定すれば車外のメッシュを取り除く。従って内部流れでも外部流れでも、blockMesh では全体を包み込むような領域を基本的に設定する訳である。

2.2.1　blockMeshDict のフォーマット

blockMeshDict とは blockMesh に関する諸設定を纏めたテキストファイルであり、**./system** に保存する。blockMesh はある程度複雑な形状のメッシュ生成が可能で、複雑な解析領域 (図 2 中グレー領域) も解釈可能である。ただし、複雑形状及び空間は STL ファイルで用意することが一般的で、blockMesh に作ってもらいたい基本メッシュは、図 2 のような矩形であることが多い。本資料でも矩形領域及び立法格子を作成するための方法のみ紹介する。

Listing 2.1　blockMeshDict の基本フォーマット

```
1  FoamFile
2  {
3          version  2.0;
4          format   ascii;
5          class    dictionary;
6          object   blockMeshDict;
7  }
8
9  minx             0;
10 maxx     1;
11 miny             0;
12 maxy     1;
13 minz             0;
14 maxz     1;
15
16 xm               (0 3 4 7);
17 xp               (1 2 6 5);
18 ym               (0 1 4 5);
19 yp               (2 3 6 7);
20 zm               (0 3 2 1);
21 zp               (4 5 6 7);
22
23 scale    1;
24
25 vertices
26 (
27         ($minx $miny $minz)
28         ($maxx $miny $minz)
29         ($maxx $maxy $minz)
30         ($minx $maxy $minz)
31         ($minx $miny $maxz)
32         ($maxx $miny $maxz)
```

```
33          ($maxx $maxy $maxz)
34          ($minx $maxy $maxz)
35
36  );
37
38  blocks
39  (
40          hex (0 1 2 3 4 5 6 7) (400 200 1) simpleGrading (1 1 1)
41  );
42
43
44  boundary
45  (
46          inlet
47          {
48                  type    patch;
49                  faces
50                  (
51                          $xm
52                  );
53          }
54          outlet
55          {
56                  type    patch;
57                  faces
58                  (
59                          $xp
60                  );
61          }
62          wall
63          {
64                  type    wall;
65                  faces
66                  (
67                          $ym
68                          $yp
69                  );
70          }
71
72          no_dim
73          {
74                  type empty;
75                  faces
76                  (
77                          $zm
78                          $zp
79                  );
80          }
81  );
82
83  mergePatchPairs
84  ();
```

Listing2.1 は基本的な blockMeshDict のフォーマットである。

- 7 行目まではヘッダーと呼ばれるものであり、修正を加えることはない。
- 9 行目から 21 行目までのように、OpenFOAM は変数定義が可能である。
- 24 行目の scale はメートルへの単位変換時に乗じる数値を定義している。例えば後に登場する数値をミリメートルで記載する場合は、ここで 0.001 を設定する。
- vertices では矩形領域の頂点座標を指定する。各点の定義の順番は重要で、ここを間違えると正しく矩形領域が定義できない。vertices で座標を直接入力するよりも、7-14 行目のように各軸の定義域を指定して、良しなに座標定義の順番が定義されるようにした方が安全である。
- blocks では矩形領域と各軸の離散数、並びに計算格子の大きさを指定する。hex(0 1 2 3 4 5 6 7) は矩形領域を表しており、各数値は vertices 中の頂点番号を意味している。従って (0 1 2 3 4 5 6 7) という定義と vertices 中頂点の定義の順番は連携している訳であり、hex(0 1 2 3 4 5 6 7) のところは基本的に修正してはならない。(400 200 1) は x 軸、y 軸、z 軸の離散数を表している。この場合、z 軸方向の離散数が多く、z 軸方向は離散化しない。このようにある軸を離散化せず、かつそれに直交する面で如何なる流束も生じないように設定すれば、OpenFOAM で 2 次元解析ができるようになる。simpleGrading (1 1 1) は各軸の計算格子幅の成長率を表す。図 2.1 のように blockMesh の機能で不均質格子を生成することは稀なので、(1 1 1) の設定から変える必要はほぼ無い。
- boundary では各面の名称と特性を定義する。Listing2.1 中の「inlet」などは面の定義であり、{} 内で諸設定を行う。type では壁面など面の特性を指定する。表 2.1 は type で指定できる特性の一覧である。faces では対象の面を指定する。例えば inlet の faces に (0 3 4 7) を記入した場合、inlet は x 軸負側の面を指していることになる。63 行目の wall のように、複数の面を faces で指定することもできる。なお表 2.1 で説明している通り、type で empty を指定すればその面で流束は生じない。
- mergePatchPairs では、いわゆる界面境界を定義する。基本的な解析で使用することは少ないので、Listing2.1 のように省略することが多い。

blockMeshDict を編集後、解析ディレクトリで「blockMesh」と実行する。ただし、このときに controlDict ファイルも保存していなければならない。controlDict については 3 章で触れる。また、blockMesh の結果を foamToVTK などで出力

表 2.1　境界のタイプ

patch	パッチ。流入出境界条件で利用する。
wall	壁面境界。
symmetryPlane	対称境界面。
cyclic	周期境界条件。
cyclicAMI	不整合周期境界条件。
wedge	2 次元軸対称。
empty	2 次元。

する場合、fvSolution と fvSchemes のファイルも必要になる。foamToVTK については 4 章で、fvSolution と fvSchemes については 3 章で触れる。

2.2.2　surfaceFeatureExtractDict のフォーマット

surfaceFeatureExtractDict とは surfaceFeatureExtract に関する諸設定を纏めたテキストファイルであり、**./system** に保存する。前述の通り、surfaceFeatureExtract は特徴線を抽出する。このような処理は他の商用ソフトでも見られるが、OpenFOAM に限らず多くのソフトにおいて特徴線抽出のためのパラメータを調整することは稀である。従って特にこだわりがなければ下記の Listing2.2 をそのまま使うとよい。なお、9 行目の geo.stl は用意した CAD ファイルのことであり、9 行目のみ STL ファイル名に合わせて変更しなければならない。

Listing 2.2　surfaceFeatureExtractDict の基本フォーマット

```
1  FoamFile
2  {
3          version  2.0;
4          format   ascii;
5          class    dictionary;
6          object   surfaceFeatureExtractDict;
7  }
8
9  geo.stl
10 {
11         extractionMethod         extractFromSurface;
12         extractFromSurfaceCoeffs
13         {
14                 includedAngle    150;
15         }
16
17         writeObj         no;
18 }
```

この状態で surfaceFeatureExtract を実行すると、**./constant/triSurface** に geo.**eMesh** というファイルが作成される。ちなみに特徴線がない非常に単純な形状であったとしても、surfaceFeatureExtract を実行しなければならない。この.eMesh ファイルがないと snappyHexMesh が実行できないためである。

2.2.3　snappyHexMeshDict のフォーマット

snappyHexMeshDict とは snappyHexMesh に関する諸設定を纏めたテキストファイルであり、**./system** に保存する。前述の通り snappyHexMesh は図 2.1(c)-(f) までの実行を担当するが、図 2.1(c)-(e) のように途中で止めることも可能である。例えば直交格子のみで構成したいならば、図 2.1(d) の完了時点で終了させればよい。

Listing 2.3　snappyHexMeshDict の基本フォーマット

```
1  FoamFile
2  {
3      version      2.0;
4      format       ascii;
5      class        dictionary;
6      object       snappyHexMeshDict;
7  }
8
9  castellatedMesh  true;
10 snap             true;
11 addLayers        true;
12
13 geometry
14 {
15         geo.stl
16         {
17                 type     triSurfaceMesh;
18                 name     geo;
19         }
20 }
21
22 castellatedMeshControls
23 {
24         maxLocalCells 100000;
```

```
        maxGlobalCells 2000000;
        minRefinementCells 0;
        maxLoadUnbalance 0.10;
        nCellsBetweenLevels 1;
        features
        (
                {
                        file "geo.eMesh";
                        level 0;
                }
        );

        refinementSurfaces
        {
                geo
                {
                        level (0 0);
                }
                inlet
                {
                        level (0 0);
                }
                outlet
                {
                        level (0 0);
                }
                wall
                {
                        level (0 0);
                }
                no_dim
                {
                        level (0 0);
                }
        }

        resolveFeatureAngle     30;
        refinementRegions
        {}

        locationInMesh (2.003 0.003 0.0);
        allowFreeStandingZoneFaces true;

}

snapControls
{
        nSmoothPatch                    3;
        tolerance                               2.0;
        nSolveIter                              30;
        nRelaxIter                              5;
        nFeatureSnapIter                10;
        implicitFeatureSnape    false;
        explicitFeatureSnape    true;
        multiRegionFeatureSnape false;
}
addLayersControls
{
    relativeSizes   true;
    expansionRatio  1.0;
    finalLayerThickness 0.3;
    minThickness    0.25;
    layers
    {
        "geo_side"
        {
            nSurfaceLayers  3;
        }
    }
    nGrow   0;
    featureAngle    130;
    maxFaceThicknessRatio   0.5;
    nSmoothSurfaceNormals   1;
    nSmoothThickness    10;
    minMedialAxisAngle  90;
    maxThicknessToMedialRatio    0.3;
    nSmoothNormals  3;
    slipFeatureAngle    30;
    nRelaxIter  5;
    nBufferCellsNoExtrude   0;
    nLayerIter  50;
    nRelaxedIter    20;
}
meshQualityControls
{
        maxNonOrtho                     65;
        maxBoundarySkewness 20;
        maxInternalSkewness 4;
```

```
113         maxConcave                      80;
114         minVol                          1e-13;
115         minTetQuality           1e-15;
116         minArea                         -1;
117         minTwist                        0.02;
118         minDeterminant          0.001;
119         minFaceWeight           0.05;
120         minVolRatio             0.01;
121         minTriangleTwist        -1;
122     errorReduction  0.75;
123     nSmoothScale    4;
124 }
125 mergeTolerance 1e-6;
```

- snappyHexMesh で何をするかは 9-11 行目で指定する。castellatedMesh は図 2.1(c)-(d) の処理、snap は図 2.1(e) の処理、並びに addlayers は図 2.1(f) の処理のことを指す。
- geometry では読み込む CAD ファイルを定義する。Listing2.3 のように、CAD ファイル名を記入し、trisurfaceMesh を指定する。name は geo.stl に変わる変数名であり、この後の処理で使うことができる。
- castellatedMeshControls では、castellatedMesh の設定をする。
  - maxLocalCells は 1 コア当たりの許容できる最大計算格子数を意味する。
  - maxGlobalCells は許容できる総計算格子数を意味する。
  - minRefinementCells は図 2.1(c) の再分割の繰り返しを制御する。繰り返し制御の途中、再分割が必要と判断されたメッシュの数が minRefinementCells 以下になったとき、繰り返し計算は終了する。
  - maxLoadUnbalance は並列計算時のコア毎の処理量のアンバランス性を設定するためにある。maxLoadUnbalance が高いほど各コアの計算量を均等にしようとし、逆に低い場合は計算の配分を柔軟に割り当てることができる。
  - nCellsBetweenLevels は少なくとも同じサイズのメッシュが並ぶ数を制御するためにある。この数が大きいほどメッシュサイズの急激な変化を緩和する。
  - features では特徴線近傍のメッシュについて設定する。file 部分で./constant/triSurfaceMesh にある eMesh ファイルを指定する (./constant/triSurface/*.eMesh のように相対パスで指定する必要はなく、ファイル名を記載するだけでよい)。また、レベルではメッシュのサイズを整数値で指定する (次項目参照)。
  - 前述の通り、snappyHexMesh はメッシュを 8 分木で管理する。メッシュを再分割するとき、元のメッシュの半分の長さ (体積は 3 次元空間の場合 1/8) のメッシュを作成する。このとき、最も大きいメッシュ (木構造の根に相当) は blockMesh で生成された基本メッシュである。snappyHexMesh は、基本メッシュのサイズを「level 0」と呼び、そこから 1 段階ずつ小さくになるにつれ level を 1 ずつ上げて呼ぶ。例えば features で level2 と指定した場合、基本的に level 0 のメッシュの 1/64 のサイズのメッシュが特徴線付近で生成される。なお、features が snappyHexMeshDict で定義されていないと snappyHexMesh 実行時にエラーとなる。それゆえ features の定義及び surfaceFeatureExtract の実行は必須となる。
  - refinementSurface では各面の分割レベルを設定できる。併せて STL データの面の定義も兼ねているので、たとえ基本メッシュからの再分割が不要であっても、refinementSurface には全ての面について記載しなければならない。定義の仕方は blockMesh 及び STL の solid で設定した名前を指定し、それぞれに分割レベルを定義する。分割レベルは level (x y) のように設定される。ここで $x < y$ であり、面上のメッシュは level x から y のサイズとなるように自動調整される。再分割不要である場合は Listing2.3 のように level (0 0) とすればよい。
  - 2 つの境界面の成す角度が小さいとき、その間のメッシュは細かくしなければならない。角度の大小の閾値は resolveFeatureAngle で設定する。
  - refinementRegions では、領域に対するメッシュ細分化を指定できる。例えばカルマン渦における後流など、細かいメッシュが必要な領域が明らかな場合に指定する。
  - locationInMesh では任意の解析領域内の座標を指定する。locationInMesh の設定によって、解析が内部流れか外部流れかが決まる。
  - allowFreeStandingZoneFaces については現状よく分かっていない。ただし、true にするのが一般的な模様。
- snapControls では snap 処理について設定する。なお、たとえ snap を false にしていたとしても、諸々のパラメータを入力しなければならない仕様になっている。snap を false にしているならば、これらパラメータが影響を及ぼすことはない。また、各パラメータの最適化は複雑なので、Listing2.3 記載のデフォルト値を使うことが多い。
  - nSmoothPatch は境界適合における平滑化反復回数である。
  - tolerance で境界適合の許容誤差を設定する。
  - nSolveIter でスナッピングの反復回数を指定する。
  - nRelaxIter でスナッピングのリラックス回数を指定する。
  - nFeatureSnapIter で特徴領域の反復回数を指定する。
  - implicitFeatureSnape が true の場合は特徴領域のスナッピングを暗黙的に行い、explicitFeature が true の場合は明示的に行う。
  - multiRegionFeatureSnape で複数の領域のスナッピングを行うか指定する。
- addLayersControls でレイヤーに関する設定を行う。重要な設定だが、本例は立方格子を生成するためのサンプルも兼ねているので、ここでは紹介しない。ただし、addLayers が false でも「addLayersControls{}」は明記しなければならない。
  - relativeSizes を true にした場合、レイヤー外メッシュサイズを相対値としてレイヤーメッシュサイズを指定できるようになる。
  - expansionRatio でレイヤーメッシュ成長率を指定する。

  - finalLayerThickness は最外殻メッシュのサイズを指す。
  - minThickness で下限サイズを指定する。
  - layers のディクショナリでレイヤーメッシュを作成する境界面を指定する。本例のように境界名とレイヤー数 (nSurfaceLayers) を指定する。なお、境界はジオメトリの名前と境界面の名前で名づける。Listing2.3 はジオメトリ geo(つまり geo.stl) に含まれる境界名 side(solid side で定義されている面) の例であり、g̈eo_side のようにアンダーバーでつなげる。
- meshQualityControls でメッシュ品質の許容閾値を指定する。Listing2.3 はデフォルト値であり、多くの場合このまま使う。
- mergeTolerance の意味は現状分かっていない。ただし、1e-6 よりも大きな数値を入れなければならないらしい。

snappyHexMeshDict を設定したら、snappyHexMesh を実行してメッシュを生成する。ただし、このとき fvSolution と fvSchemes のファイルも用意されていなければエラーとなる。

2.2.4 その他処理と全体の流れ

snappyHexMesh 実行後、次にメッシュ品質のチェックを勧める。OpenFOAM はチェックのため処理を用意しており、**checkMesh -constant** のコマンドを実行するだけでよい。なお、snappyHexMesh の結果は./constant に保存されており、checkMesh の後に続くオプション引数はそのことを意味している。

次に不要な境界を削除していく。snappyHexMesh を実行したとき、結果的に境界にメッシュが生成されないことがある (例えば内部流れにおける blockMesh のいくつかの境界)。このような境界は不要なので、削除することが望ましい。OpenFOAM はそのためのコマンドとして、**createPatch -overwrite** を用意している。なお、-overwrite は snappyHexMesh の結果に上書きするためのもので、多くの場合指定される。また、createPatch の実行には下記の **createPatchDict** が必要であり、実行時には createPatchDict を**./system** フォルダに保存する。

最後に、連立一次方程式の係数行列のバンド幅を小さくするために、メッシュの定義の順序を並び変える。そのためには、**renumberMesh -overwrite** を実行すればよい。

Listing 2.4　createPatchDict の基本フォーマット

```
1  FoamFile
2  {
3          version 2.0;
4          format  ascii;
5          class           dictionary;
6          object          createPatchDict;
7  }
8
9  pointSync       false;
10 patches
11 (
12 );
```

最後に blockMesh から renumberMesh までの一連のコマンドを記す。

Listing 2.5　メッシュ作成手順

```
1 blockMesh
2 surfaceFeatureExtract
3 snappyHexMesh -overwrite
4 checkMesh -constant
5 createPatch -overwrite
6 renumberMesh -overwrite
```

3　解析条件の設定

解析条件の設定は controlDict や fvSchemes などのファイルで設定するが、ファイルに記述すべき内容は解法 (simple 法など) によって大きく異なり、しかもややこしい。例えば解法の simpleFOAM は定常乱流解析をデフォルトとして考えているため、たとえ層流解析であっても乱流応力の収束などについてわざわざ指定しなければならない。そのため、解析をするときは一から解析条件設定のためのファイルを用意するのではなく、解法毎に用意されたファイルを引用することが多い。

とはいえ各ファイルの意味を理解することは重要なので、本章の初めに各ファイルの紹介をする。解法毎に用意された設定ファイルはその後に回す。

3.1　解析条件設定ファイルの紹介

3.1.1 controlDict

controlDict とは解析の時間刻み幅や出力トリガー等を設定するファイルであり、**./system** に保存する。controlDict の各設定値の解釈は定常計算と非定常計算で異なる。

まず初めに定常計算における controlDict のフォーマットを以下に示す。

Listing 3.1　定常計算における controlDict の基本フォーマット

```
1 FoamFile
2 {
```

```
3           version 2.0;
4           format  ascii;
5           class   dictionary;
6           object  controlDict;
7   }
8   application     simpleFoam;
9   startFrom       startTime;
10  startTime       0;
11  stopAt  endTime;
12  endTime 1000;
13  deltaT  1;
14  writeControl    timeStep;
15  writeInterval   100;
16  purgeWrite      0;
17  writeFormat     ascii;
18  writePrecision  6;
19  writeCompression        off;
20  timeFormat      general;
21  timePrecision   6;
22  runTimeModifiable       true;
```

- application では、利用するソルバー名 (後述) を指定する。
- controlDict では、連立一次方程式の反復計算回数のことを時刻と呼んでいる。
- startFrom では下記の何れかを選択し、開始時刻を指定する。
  - latestTime：既に存在するデータの中で、最も遅い時刻 (つまり最新時刻) から開始。これを選択すれば中断していた解析を再開することができる。
  - firstTime：既に存在する時刻データの中で、最も早い時刻から開始。
  - "startTime"で設定した時刻から開始。これを選択した場合は 12 行目のように数値を入力しなければならない。
- stopAt では下記の何れかを選択し、解析終了時刻を指定する。
  - writeNow：現時刻の計算を終えてから、計算結果を書き出して停止。
  - noWriteNow：現時刻の計算を終えてから、計算結果を書き出さずに停止。
  - nextTime：次の計算結果の書き出し時刻 (後述) で停止。
  - endTime："endTime"で設定した時刻で終了。これを選択した場合は 20 行目のように数値を入力しなければならない。
- 定常解析の場合、deltaT、つまり時間刻み幅は必ず 1 とする。
- writeControl 及び writeInterval で計算結果出力のタイミングを設定する。一般的には writeControl で timeStep を選択し、writeInterval でその間隔を設定する。
- 出力結果の総メモリ制御のために、purgeWrite で出力の最大保存数を指定する。例えば数値 N と指定した場合、時系列出力結果のうち最新の N 個のみ保存する。ただし、purgeWrite をゼロとした場合は全出力結果を保存する。
- 計算結果ファイルフォーマットは writeFormat で指定する (基本的に ascii でよい)。
- writeFormat が ascii の場合、計算結果の数値の桁数は writePrecision で指定できる。
- writeCompression で、出力結果を都度圧縮するか否かを指定できる。指定には on/off を使う。
- 時刻の書式 (例えば 12.34 なのか 1.234e+1 なのか) は timeFormat で指定できる。general としておけば OpenFOAM がよしなに書式を選択してくれる。
- 時刻数値の桁数は timePrecision で指定する。
- 解析再開時に controlDict ファイルを再読み込みして欲しい場合は、runTimeModifiable を true とする。基本的に true でよい。

非定常解析の場合、時間を秒で指定する。ただし writeInterval のみはタイムステップ数である。

Listing 3.2　非定常計算における controlDict の基本フォーマット

```
1   FoamFile
2   {
3       version     2.0;
4       format      ascii;
5       class       dictionary;
6       object      controlDict;
7   }
8   application     icoFoam;
9   startFrom       latestTime;
10  startTime       0;
11  stopAt          endTime;
12  endTime         10;
13  deltaT          0.05;
14  writeControl    timeStep;
15  writeInterval   20;
16  purgeWrite      0;
17  writeFormat     ascii;
18  writePrecision  6;
19  writeCompression off;
20  timeFormat      general;
21  timePrecision   6;
22  runTimeModifiable true;
```

3.1.2 transportProperties

物性値の設定は **transportProperties** ファイルで行う。このファイルは**./constant** に保存する。解析に必要な物性値は解析条件毎に異なる。例えば層流解析の場合は以下のように動粘性係数のみ指定すればよい。

Listing 3.3 transportProperties の例

```
FoamFile
{
    version 2.0;
    format  ascii;
    class   dictionary;
    object  transportProperties;
}

nu  0.01;
```

3.1.3 fvSchemes

有限体積法において必要な離散化の設定は **fvSchemes** ファイルで行う。このファイルは**./system** に保存する。
fvSchemes では以下の離散化を設定する (フォーマット例は別の節で紹介する)。

- ddtSchemes：時間微分に関する離散化。以下のいずれかを設定。
  - steadyState：定常解析の場合。
  - Euler：Euler 法 (1 次精度) の場合。
  - CrankNicolson：クランク-ニコルソン法の場合。例えば「default CrankNikolson x」のように記載する。ここで x は [0,1] のパラメータであり、0 ならば Euler 法と同じ処理になる。
  - backward：後退差分 (2 次精度) の場合。
- gradSchemes：セル界面上の勾配を計算するための補間方法。基本的に Gauss linear にする。
- divSchemes：発散項に関する離散化。
- laplacianSchemes：ラプラシアンに関する離散化。基本的に Gauss linear corrected とする。
- interpolationSchemes：セル界面上の物理量を計算するための補間方法。基本的に linear とする。
- snGradSchemes：非直交補正に関する設定。基本的に corrected とする。

離散化方法は物理量毎に指定できる。また、デフォルトの指定も可能であり、明示的に指定されなかった物理量はデフォルトの方法で離散化される。

上記離散化の中で最も重要なものは divSchemes である。物理量 X に関する発散項は div(X) と書かれる。一方で div(phi, X) は X による phi の移流を意味する。流体解析において対流項の取り扱いは重要であるため、div(phi, X) の設定には特に注意しなければならない。以下に divSchemes で設定できる方法を示す。

- linear：線形補間 (中心差分、2 次精度)。
- upwind：1 次風上差分。
- linearUpwind：線形風上差分 (2 次精度)。
- QUICK：QUICK スキーム (2 次精度)。
- Minmod：minmod 制限関数 (2 次精度 TVD)。
- SuperBee：superbee 制限関数 (2 次精度 TVD)。
- vanLeer：van Leer 制限関数 (2 次精度 TVD)。
- vanAlbada：van Albada 制限関数 (2 次精度 TVD)。
- UMIST：UMIST 制限関数 (2 次精度 TVD)。
- MUSCL：Monotonized central difference 制限関数 (2 次精度 TVD)。
- limitedLinear：線形補間に TVD 制限をつけたもの。安定性を調整するための [0, 1] のパラメータを設定する必要がある。例えば「default limitedLinear 1」のように指定する。このパラメータが大きいほど安定性は高い。ベクトル用の離散化では limitedLinearV を用いる。

なお、計算を安定化するために bounded という指定ができ、例えば「bounded Gauss upwind」のように設定する。定常解析時にこれを利用することが推奨されている。bounded 以外にも Gauss というものを指定することが多いが、私はまだこの意味を理解していない。

3.1.4 fvSolution

fvSolution では線形代数ソルバーの解法を設定する。解法は物理量毎に設定することができる。いくつか解法は用意されているが、圧力に対しては PCG(前処理付き CG 法)、それ以外は PBiCG(前処理付き BiCG 法) を使えばよい。以下は simple 法における例である。

Listing 3.4 fvSolution の例

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
```

```
solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0.01;
    }
    U
    {
        solver          PBiCG;
        smoother        DILU;
        tolerance       1e-05;
        relTol          0.1;
    }
}
SIMPLE
{
    nNonOrthogonalCorrectors 2;
    pRefPoint    (40 0 0);
    pRefValue    0;
    residualControl
    {
        p    1e-3;
        U    1e-3;
    }
}
relaxationFactors
{
    fields
    {
        p    0.3;
    }
    equations
    {
        U    0.7;
    }
}
```

- solvers で連立一次方程式の解法を設定する。基本的に前処理は上記のものを使うとよい。tolerance は残差の閾値、refTol は残差比 (初期残差に対する比率) であり、これらよりも低くなれば収束したと判定する。
- 本例では SIMPLE 法を用いている。
  - nNonOrthogonalCorrectors は格子の非直交性に関わる数値であり、基本的に 2 とする。
  - pRefPoint は基準圧力とする位置、pRefValue は基準圧力値である。圧力境界条件がある場合は参照されない。
  - residualControl は収束判定値である。
- relaxationFactors は連立一次方程式ソルバーの緩和係数である。fields と equations の違いは分からない。

3.1.5 初期条件と境界条件の設定

B.C.(境界条件) と I.C.(初期条件) の設定は**./0** で行う。