ま え が き

このドキュメントは，2016 年にオープン CAE 勉強会@関西で実施された”講師のきまぐれ OpenFOAM 講習会”用のテキストです．内容はかの有名な”PENGUINITIS”サイトの OpenFOAM 計算例 (キャビティ流れ，円柱まわり流れ，曲がり管内の流れ) をトレースしたものとなっています．

なお，計算結果の妥当性については一応検証していますが，全てを保証するものではないので業務や研究に活用される場合は各自の責任でお願いいたします．

2016 年 6 月

yotakagi77

目　　次

1

キャビティ流れ

1.1 解析モデル

CFD のベンチマーク問題としてよく取り上げられるキャビティ流れは図 1.1 に示すような二次元矩形キャビティの中の流れであり (三次元の立方体でもキャビティであるが流れのパターンが異なる), 3 つの境界が静止壁面, 残りの 1 つが一様な速度でスライドする壁面となっている．キャビティ流れは OpenFOAM のユーザーガイドに載っている基本的なチュートリアルであり，OpenFOAM ユーザーであれば一度は実行したことがある解析である．今回の講習では，有名な論文である Ghia *et al.*[2] の結果との比較を行う．なお，Ghia *et al.* の解析は圧力のポアソン方程式を解く通常の流体解析ではなく，流関数-渦度法を用いていることに注意されたい．流関数-渦度法について勉強したい方は，例えば河村[3] のテキストなどを参照するとよい．

非圧縮性流体の解析を実施するとき，現象を特徴づける大事なパラメータ (無次元数) はレイノルズ数 Re であり，以下のように定義される．

$$Re = \frac{UL}{\nu} \tag{1.1}$$

ここで，U は代表速度，L は代表長さ，ν は動粘性係数である．キャビティ流れでは，U は移動壁面速度，L は領域一辺の長さにとる．レイノルズ数を変更したいとき，実験を想定すると U を変化させる方法が考えられるが，ν を変更した方が後の計算結果の比較が行いやすい．すなわち，有次元の解析であっても，$U = 1$ m/s，$L = 1$ m とすれば，出力結果は単位を無視すればそのまま無

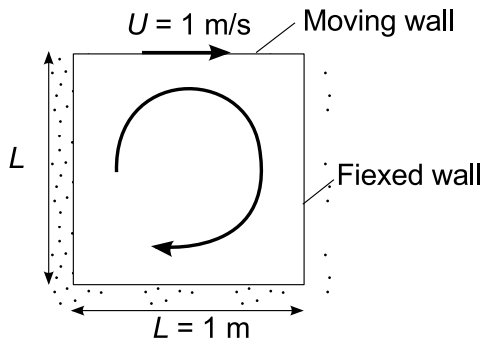

図 1.1 キャビティ流れ.

次元した値となる．ちなみに，実験において動粘性係数を段階的に変更することは容易ではないが，作動流体を変更する (水を空気にする)，水をグリセリン水溶液にする，などといった方法で少しは変更できる．

1.2 計 算 条 件

今回の解析では，レイノルズ数を 100 から 10000 まで変化させる ($Re = 100$, 1000, 5000, 10000)．レイノルズ数を変化させるためには上述したように動粘性係数の値を変えることによって行うため，今回の解析では $U = 1.0$ m/s, $L = 1.0$ m と固定し，各レイノルズ数における ν の値は式 1.1 から逆算して表 1.1 のように設定する．

表 1.1 各レイノルズ数における動粘性係数の設定

Re	ν
100	0.01
1000	0.001
5000	0.0002
10000	0.0001

解析領域は 2 次元とし，キャビティの寸法は 1 m × 1 m とする．計算格子は等間隔直交格子を用い，セル分割数は 200 × 200 として各レイノルズ数で共通とする．このとき格子幅は $\Delta x = 1/200 = 5 \times 10^{-3}$ m となる．一般にレイノルズ数が高くなると速度境界層が薄くなりより高解像度な計算格子が必要と

なるが，今回は共通の計算格子を用いる．なお，比較する Ghia *et al.*[2] ではセル分割数を 128×128 または 256×256 としている (論文中では格子点数なので 129×129 または 257×257 と記載)．

境界条件はオリジナルのチュートリアルと同じとし，速度に関してはディリクレ条件，圧力に関してはノイマン条件とする．なお，非圧縮性流体解析ではすべての境界で圧力に対してノイマン条件を用いると一般に収束が遅いことに注意されたい．

使用ソルバーはオリジナルチュートリアルと同じように `icoFoam` とし，離散化スキームの設定も同じものを用いる．

時間刻み幅 Δt は最大速度 u_{max}，Δt，Δx によって定義されるクーラン数，

$$Co = \frac{u_{max}\Delta t}{\Delta x} \tag{1.2}$$

が 1 以下になるように決めると，動く壁面速度を u_{max} にして，

$$\Delta t \leq \frac{Co \cdot \Delta x}{u_{max}} = \frac{1.0 \cdot 5 \times 10^{-3}}{1.0} = 5 \times 10^{-3}\mathrm{s} \tag{1.3}$$

となる．計算時間 (積分時間) t_{end} は各条件共通で 100 s として解析してみる (条件によってはさらに長い時間の計算が必要な場合もある)．

以上の計算条件を表にまとめると表 1.2 のようになる．なお，OpenFOAM では二次元解析であっても奥行方向 (z 方向) の長さを指定する必要があるため，適当な長さ ($L_z = 0.1$ m) 及びセル分割数 ($N_z = 1$) を設定している．

表 1.2 キャビティ流れ解析の計算条件.

Property	Value
L_x	1.0 m
L_y	1.0 m
L_z	0.1 m
N_x	200
N_y	200
N_z	1
Δx, Δy	5.0×10^{-3} m
Δt	5.0×10^{-3} s
t_{end}	100 s
Solver	`icoFoam`
Schemes	i.q. original tutorial

1.3 解 析 手 順

1.3.1 オリジナルチュートリアルの実行

はじめに OpenFOAM の環境設定確認も兼ねてオリジナルチュートリアルを実行してみる．なお本ドキュメントでは，端末上でタイプするコマンドやテキストファイルの編集部分を丸い角の枠表示で記述することとする．

作業 1

```
$ run
$ cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity ./
$ cd cavity
$ foamRunTutorials
```

計算が正常に終了したことをログファイル (log.blockMesh，log.icoFoam) によって確認し，ParaView によって可視化もしてみる．

作業 2

```
$ more log.*
$ paraFoam
```

ParaView による可視化の手順は OpenFOAM ユーザーガイドを参照されたい．なお，オリジナルチュートリアルの応用チュートリアルとして cavityClipped, cavityGrade が同じく $FOAM_TUTORIALS/incompressible/icoFoam/ にあるので，ユーザーガイドを参考に内容を理解されることを推奨する．

いま実行したオリジナルチュートリアルケースの中には自動実行スクリプト Allrun がないので (1 つ上のディレクトリにあるが他の icoFoam チュートリアルも実行するため複雑)，自分で簡単なものをコピーしてきて用意する．

作業 3

```
$ foamCleanTutorials
$ cp -r $FOAM_TUTORIALS/basic/potentialFoam/pitzDaily/Allrun ./
$ vi Allrun
```

Allrun の編集 (修正後)

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

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

application=‘getApplication‘

runApplication blockMesh
runApplication $application
```

必要であれば `Allclean` も用意する．

作業 4

```
$ cp -r $FOAM_TUTORIALS/basic/potentialFoam/pitzDaily/Allclean ./
$ vi Allclean
```

Allclean の編集 (修正後)

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

# Source tutorial clean functions
. $WM_PROJECT_DIR/bin/tools/CleanFunctions

cleanCase
```

用意した `Allrun`，`Allclean` が正常に機能するか確認する．

作業 5

```
$ ./Allclean
$ ls
$ ./Allrun
```

1.3.2 $Re = 100$ ケースの作成・計算実行

オリジナルのチュートリアルは $Re = 1.0 \times 0.1/0.01 = 10$ であり，領域の大きさ等も変更する必要がある．まずはじめに，一番レイノルズ数の低い $Re = 100$ のケースを準備して計算を実行する．先ほど実行したオリジナルチュートリアルのケースをコピーする．

作業 6

```
$ run
$ foamCloneCase cavity cavityRe00100
$ cp cavity/All* cavityRe00100/
$ cd cavityRe00100/
```

foamCloneCase は OpenFOAM 用コマンドであり，計算結果を除くファイルをコピーできる．ただし Allrun，Allclean はコピーされないため cp でコピーする．新しく作成したケースディレクトリは条件 (レイノルズ数) がわかるように cavityRe00100 としている．

編集すべきファイルは blockMeshDict，controlDict であり，はじめに blockMeshDict を書き換える．ここでオリジナルの blockMeshDict では convertToMeters を 0.1 としているので書き換える必要がある．

作業 7

```
$ vi constant/polyMesh/blockMeshDict
```

blockMeshDict の変更箇所

```
convertToMeters 1;
 ...
blocks
(
    hex (0 1 2 3 4 5 6 7) (200 200 1) simpleGrading (1 1 1)
);
```

次に controlDict を書き換える．

作業 8

```
$ vi system/controlDict
```

controlDict の変更箇所

```
endTime         100;
 ...
writeInterval   200;
```

今回の解析ではオリジナルに比べて計算時間が長くなるため，データの出力間隔 writeInterval も変更している．

表 1.1 に示したとおり動粘性係数は $\nu = 0.01$ とするが，この値はオリジナル

と同じである．念のため確認する．

作業 9

```
$ more constant/transportProperties
```

U，p，fvSchems，fvSolution も変更する必要はないが，気になる場合は確認する．

最終的な確認が終わったら Allrun を用いて計算を実行する．

作業 10

```
$ ./Allrun &
```

マシンスペックにも依るが，計算時間が長いので最終時刻 100 s になるまで時間がかかる．この待ち時間の間に，Ghia *et al.*[2] とのデータとの比較のために必要な sample ユーティリティの準備をする．このユーティリティは sampleDict を system 以下に用意して実行するため，ひな型ファイルをコピーしてきて編集する．

作業 11

```
$ cp $FOAM_UTILITIES/postProcessing/sampling/sample/sampleDict system/
$ vi system/sampleDict
```

sampleDict の変更箇所

```
sets
(
    lineX1
    {
        type        uniform;
        axis        x;
        start       (0.0 0.5 0.05);
        end         (1.0 0.5 0.05);
        nPoints     200;
    }
    lineY1
    {
        type        uniform;
        axis        y;
        start       (0.5 0.0 0.05);
        end         (0.5 1.0 0.05);
        nPoints     200;
```

sampleDict の変更箇所 (続き)

```
    }
);
 ...
surfaces
(
);
```

今回の解析結果比較では，キャビティ中心線上 ($x = 0.5$ 及び $y = 0.5$) での速度分布を比較するため，sets セクションで直線を 2 本指定する．また面サンプルはしないので，surfaces セクションはすべて中身を削除する (コメントアウトでも良い)．

sampleDict の編集が終わり，$Re = 100$ の計算が終了していたら，sample コマンドを実行する．サンプルするのは最終時刻だけで良いので以下のようにオプションをつける．

作業 12

```
$ sample -latestTime
$ ls postProcessing/sets/100/
```

sample 実行後に最終時刻のフィールドデータからサンプルした速度及び圧力のデータが出力されていることが確認できる．

1.4 Ghia *et al.* の結果との比較

Ghia *et al.*[2] の論文には，キャビティ中心線上での速度，流線 (流関数の等高線)，渦度等高線，移動壁面上での渦度，渦中心 (primary vortex, seconday vortex) の位置が掲載されている．オリジナル論文から抜き出した結果を表 1.3(速度 u)，表 1.4(速度 v)，表 1.5(流関数・渦度の等高線の値) に示す．その他の結果については論文[2] を参照されたい．

1.4.1 gnuplot による中心線上速度分布の比較

まずはじめに，中心線上の速度分布を比較してみる．sample コマンドで出力した結果と論文の値 (表 1.3・表 1.4) をグラフソフトでプロットしてみる．こ

表 1.3 垂直中心線 ($x = 0.5$ に沿った速度の水平方向成分 (u) の分布[2)]

		Re						
Pt. No.	y	100	400	1000	3200	5000	7500	10000
129	1.0000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
126	0.9766	0.84123	0.75837	0.65928	0.53236	0.48223	0.47244	0.47221
125	0.9688	0.78871	0.68439	0.57492	0.48296	0.46120	0.47048	0.47783
124	0.9609	0.73722	0.61756	0.51117	0.46547	0.45992	0.47323	0.48070
123	0.9531	0.68717	0.55892	0.46604	0.46101	0.46036	0.47167	0.47804
110	0.8516	0.23151	0.29093	0.33304	0.34682	0.33556	0.34228	0.34635
95	0.7344	0.00332	0.16256	0.18719	0.19791	0.20087	0.20591	0.20673
80	0.6172	-0.13641	0.02135	0.05702	0.07156	0.08183	0.08342	0.08344
65	0.5000	-0.20581	-0.11477	-0.06080	-0.04272	-0.03039	-0.03800	0.03111
59	0.4531	-0.21090	-0.17119	-0.10648	-0.08664	-0.07404	-0.07503	-0.07540
37	0.2813	-0.15662	-0.32726	-0.27805	-0.24427	-0.22855	-0.23176	-0.23186
23	0.1719	-0.10150	-0.24299	-0.38289	-0.34323	-0.33050	-0.32393	-0.32709
14	0.1016	-0.06434	-0.14612	-0.29730	-0.41933	-0.40435	-0.38324	-0.38000
10	0.0703	-0.04775	-0.10338	-0.22220	-0.37827	-0.43643	-0.43025	-0.41657
9	0.0625	-0.04192	-0.09266	-0.20196	-0.35344	-0.42901	-0.43590	-0.42537
8	0.0547	-0.03717	-0.08186	-0.18109	-0.32407	-0.41165	-0.43154	-0.42735
1	0.0000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

表 1.4 水平中心線 ($y = 0.5$ に沿った速度の垂直方向成分 (v) の分布[2)]

		Re						
Pt. No.	x	100	400	1000	3200	5000	7500	10000
129	1.0000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
125	0.9688	-0.05906	-0.12146	-0.21388	-0.39017	-0.49774	-0.53858	-0.54302
124	0.9609	-0.07391	-0.15663	-0.27669	-0.47425	-0.55069	-0.55216	-0.52987
123	0.9531	-0.08864	-0.19254	-0.33714	-0.52357	-0.55408	-0.52347	-0.49099
122	0.9453	-0.10313	-0.22847	-0.39188	-0.54053	-0.52876	-0.48590	-0.45863
117	0.9063	-0.16914	-0.23827	-0.51550	-0.44307	-0.41442	-0.41050	-0.41496
111	0.8594	-0.22445	-0.44993	-0.42665	-0.37401	-0.36214	-0.36213	-0.36737
104	0.8047	-0.24533	-0.38598	-0.31966	-0.31184	-0.30018	-0.30448	-0.30719
65	0.5000	0.05454	0.05186	0.02526	0.00999	0.00945	0.00824	0.00831
31	0.2344	0.17527	0.30174	0.32235	0.28188	0.27280	0.27348	0.27224
30	0.2266	0.17507	0.30203	0.33075	0.29030	0.28066	0.28117	0.28003
21	0.1563	0.16077	0.28124	0.37095	0.37119	0.35368	0.35060	0.35070
13	0.0938	0.12317	0.22965	0.32627	0.42768	0.42951	0.41824	0.41487
11	0.0781	0.10890	0.20920	0.30353	0.41906	0.43648	0.43564	0.43124
10	0.0703	0.10091	0.19713	0.29012	0.40917	0.43329	0.44030	0.43733
9	0.0625	0.09233	0.18360	0.27485	0.39560	0.42447	0.43979	0.43983
1	0.0000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

表 1.5 流関数 (ψ) 及び渦度 (ω) の等高線の値[2)]

Stream function				Vorticity	
Contour letter	Value of ψ	Contour number	Value of ψ	Contour number	Value of ω
a	-1.0×10^{-10}	0	1.0×10^{-8}	0	0.0
b	-1.0×10^{-7}	1	1.0×10^{-7}	± 1	± 0.5
c	-1.0×10^{-5}	2	1.0×10^{-6}	± 2	± 1.0
d	-1.0×10^{-4}	3	1.0×10^{-5}	± 3	± 2.0
e	-0.0100	4	5.0×10^{-5}	± 4	± 3.0
f	-0.0300	5	1.0×10^{-4}	5	4.0
g	-0.0500	6	2.5×10^{-4}	6	5.0
h	-0.0700	7	5.0×10^{-4}		
i	-0.0900	8	1.0×10^{-3}		
j	-0.1000	9	1.5×10^{-3}		
k	-0.1100	10	3.0×10^{-3}		
l	-0.1150				
m	-0.1175				

こではOpenFOAMの計算を実行しているLinux環境を想定し，フリーのグラフソフトであるgnuplotでのプロットを行う．gnuplotはスペース区切りの数値データファイルを読み込むので，表1.3・表1.4の値をテキストファイルに変換するなどして準備する．gnuplotは1列目に”#”があるとその行はコメント行とみなすので，適宜レイノルズ数など記入しておく．

作業 13

```
$ run
$ mkdir Ghia
$ cd Ghia
$ vi u-vel.dat
$ vi v-vel.dat
```

u-vel.dat の中身

```
#Results for u-velocity
#No.   y    100       400       1000      3200      5000      ...
129 1.0000  1.00000  1.00000  1.00000  1.00000  1.00000  ...
126 0.9766  0.84123  0.75837  0.65928  0.53236  0.48223  ...
 ...
```

テキストファイルが用意できたらgnuplotを起動してOpenFOAMの結果と文献値を比較してみる．

作業 14

```
$ run
$ gnuplot
gnuplot> plot 'cavityRe00100/postProcessing/sets/100/lineY1_U.xy' \
usi 1:2 w l, 'Ghia/u-vel.dat' usi 2:3 w p
gnuplot> plot 'cavityRe00100/postProcessing/sets/100/lineX1_U.xy' \
usi 1:3 w l, 'Ghia/v-vel.dat' usi 2:3 w p
gnuplot> quit
```

ここで gnuplot の `plot` で指定するところが長くなるため折り返し"\"を記述しているが，実際の端末では 1 行に入力する．デフォルトの gnuplot の設定では X ウィンドウの画面にグラフがプロットされ，概ね文献値と一致する結果が得られていることがわかる．ここで作業したように gnuplot を起動してインタラクティブに描画しても良いが細かい設定をタイプするのが面倒なので，通常はスクリプト (テンプレート) を用意しておいて自動的に描画する方が便利である．筆者が利用しているテンプレートを以下に示すので，必要であれば利用されたい．他にも Web を検索すればいろいろなデモ・テンプレートが入手できる．

作業 15

```
$ vi uRe00100.plt
$ gnuplot uRe00100.plt
```

uRe00100.plt の中身

```
#
#   Template for gnuplot     2016.06.04
#
#---- Title ----
set title 'Re = 100' font 'Arial, 24'
#
#---- Axis caption ----
#
set xlabel 'y' font 'Arial, 20'
set ylabel 'u-velocity' font 'Arial, 20'
#---- Axis format ----
set format x "%3.1f"
set format y "%3.1f"
#
#---- Length & scale ---
set size 0.7,0.7
#
```

uRe00100.plt の中身 (続き)

```
#---- Aspect ratio ----
set size ratio 0.77
#
#---- Range of x & y ----
set xrange [ 0 : 1 ]
set yrange [ -0.4 : 1 ]
#
set xtics 0.2
set ytics 0.2
#
#---- Start position of legend ----
set key reverse
set key left top
#
#---- Insert text ----
#set label "abc"  at  2,3 center
#set label "123"  at  2,4 center
#
#---- Arrow ----
#set arrow  from  0,0 to 0.5,1.0 nohead
#
#---- Grid ----
#set grid
#
#---- Line style ----
set style line 1 lt 1 lw 2 pt 1
set style line 2 lt 2 lw 2 pt 2
set style line 3 lt 3 lw 2 pt 3
set style line 4 lt 4 lw 2 pt 4
set style line 5 lt 5 lw 2 pt 5
#
#---- Plot graph on terminal ----
#
set terminal X11
plot 'cavityRe00100/postProcessing/sets/100/lineY1_U.xy' \
     usi 1:2 ti 'icoFoam' ls 1 w l, \
     'Ghia/u-vel.dat' usi 2:3 ti 'Ghia et al.' ls 2 w p

pause -1 "Hit return key  "

#---- Output eps file ----
set terminal postscript eps enhanced
set output "uRe00100.eps"
replot

#---- Output SVG file ----
#set terminal svg
#set output "uRe00100.svg"
#replot
```

このスクリプトを用いて gnuplot を実行すると X ウィンドウにグラフが表示され，元の端末画面には `Hit return key` と表示される．リターンキーを押すと gnuplot が終了するとともにグラフが eps ファイルとして出力される．eps ファイルはベクトル形式のフォーマットなので拡大・縮小しても解像度が変わらず文書に挿入するときに便利である．Linux 端末で簡易に eps ファイルをみたい場合には `evince` を使うと便利である．

作業 16

```
$ evince uRe00100.eps
```

同様にしてこのスクリプトファイルをコピー・編集して鉛直方向速度 v の分布も再度出力してみる．

作業 17

```
$ cp uRe00100.plt vRe00100.plt
$ vi vRe00100.plt
$ gnuplot vRe00100.plt
$ evince vRe00100.eps
```

スクリプトファイルの変更箇所については自分で確かめながら修正されたい．

1.4.2 ParaView による流線・渦度の可視化

次に流関数及び渦度の等高線を可視化してみる．流関数及び渦度の計算は OpenFOAM の後処理ユーティリティ `streamFunction` 及び `vorticity` を使って行うことができる．時間発展をみたいのならばオプション無しでコマンドを実行すれば良いが，今回は十分に流れが発達した最終時刻のみに興味があるので`-latestTime` オプションを付ける．後の等高線による可視化で渦度の z 方向成分のみが必要となるので，ベクトルの成分を書き出す `foamCalc` コマンドも一緒に実行する．

作業 18

```
$ cd cavityRe00100
$ streamFunction -latestTime
$ vorticity -latestTime
$ foamCalc components vorticity
```

正常に流関数と渦度が計算できたら，ParaView で可視化してみる．

作業 19

```
$ paraFoam
```

各フィールド変数の等高線を描画するが，ParaView が起動したら GUI での操作となるので，手順を以下に示す．

1）'Last Frame' (▷ |) をクリックして最終時刻 (100 s) にし，'Apply' をクリックする．
2）左側の'Properties' から，'Volume Fields' 内の'magVorticity' と'vorticity'，'Point Fields' 内の'streamFunction' をチェックして'Apply' をクリックする．
3）'Contour' をクリックし，'Properties'-'Contour By' を'streamFuncton' に変更する．'Isosurfaces'-'Value Range' に表 1.5 の流関数の値を入力する．終了したら'Apply' をクリックする．
4）奥行き方向も見える三次元表示になっているので，二次元断面にする．'Slice' をクリックし，'Properties'-'Z Normal' にクリックして変更する．'Show Plane' をアンチェックする．
5）等高線が細く感じる場合は, 表示方法を'Wireframe' に変更し, 'Properties'-'Styling'-'Line Width' の数値を大きくする．等高線の色を変更したい場合は，'Coloring'-'Edit' をクリックして変更できる．等高線の色を等高線の値に対応して変更したい場合は，'Pipeline Browser' から'Contour1' を選択し, 'Compute Scalars' をチェックする．その後'Coloring' を'streamFunction' に変更する．同じ操作を'Slice1' に対しても行い，色調を変更したい場合は'Edit Color Map' をクリックして'Color Map Editor' で好みのパターンに変更する．
6）背景色を変更したい場合は，'Edit'-'View Settings...' をクリックして調整する．座標軸の設定も同じパネルで変更できる．
7）キャビティの枠線を表示するには, 'Pipeline Browser' の'cavityRe00100.OpenFOAM' を非表示から表示に変更し，表示方法を'Outline' に変更する．
8）図が完成したらスナップショットを保存する．'File'-'Save Screenshot...'

でファイル名を付けて png 形式などで保存できる．ParaView のバージョンが 4.x であれば'File'-'Export Scene...' で eps 形式などでも出力できる．

9）上記の手順では'Contour' から'Slice' をパイプラインでつなげているが，'Contour' のみでも二次元的に表示できる．三次元のコンターを表示した後に，'Edit'-'View Settings...'-'General' から，'Use Parallel Projection' をチェックして'Apply' をクリックすれば平行投影で二次元的に表示される．また，'Slice'-'Contour' の順番でパイプラインをつなげても特に問題はない．

10）描画するフィールド変数を渦度の z 方向成分 (vorticityz) に変更して上記の作業を繰り返す (magVorticity では渦度の絶対値なので向きがないことが注意)．

作成したコンター図が文献の図と同じかどうかを比べる．厳密には文献のコンター図から値を何らかの方法で抜き出して重ねて比較しなければいけないが，文献のコンター図をスキャンしてその画像に上記の出力を重ねて比較してもよい．

1.5　高レイノルズ数ケース ($Re = 1000, 5000, 10000$) の解析

今回のように解析すべきケースの数がそれほど多くない場合は，1.3.2 節で作成した $Re = 100$ のケースディレクトリをコピーして動粘性係数 ν を修正して計算を繰り返せばよい．

ただし，パラメータを変更するだけのパラメトリックスタディの場合，ケース数が多くなると条件の修正作業が面倒であり，また計算ジョブを同時に投入できない環境では計算状況を適宜確認しなければいけない．

ここではレイノルズ数を変化させた解析を半自動化して実行するスクリプトを作成して解析を行う．まず `run` 直下に実施すべき解析条件リストを記述したテキストファイルを用意する．ファイルには動粘性係数とレイノルズ数の対応を記入しておく．

作業 20

```
$ run
$ vi condition.dat
```

condition.dat の中身

```
# nu Re
#0.01 100
0.001 1000
0.0002 5000
0.0001 10000
```

ここで#はコメント行であることを示しており，$Re = 100(\nu = 0.01)$ の解析は終了しているので#を挿入している．

自動化の方針としては, $Re = 100$ のケースをコピーして transportProperties の nu の値を置換して解析を順次行うシェルスクリプトを用意する．

作業 21

```
$ vi cavityHighRe.sh
```

cavityHighRe.sh の中身

```
#!/bin/bash

. /opt/OpenFOAM/OpenFOAM-2.4.x/etc/bashrc

while read LINE
do
  if [[ ! "$LINE" =~ ^# ]]; then
    NU=‘echo ${LINE} |awk ’{print $1;}’‘
    RE=‘echo ${LINE} |awk ’{print $2;}’‘
    CASE="Re‘printf %05d $RE‘"
    NEWCASE="cavity$CASE"
    foamCloneCase cavityRe00100 $NEWCASE
    cp cavityRe00100/All* $NEWCASE/
    cd $NEWCASE
    sed -i constant/transportProperties -e "s/0.01/$NU/"
    ./Allrun
    cd ..
  fi
done < condition.dat
```

このシェルスクリプトの詳細については，一般的なシェルスクリプトに関する参考書[4, 5] を参照されたい．OpenFOAM の環境変数の設定読み込みは大概

のシステムでは必要ない．シェルスクリプトが完成したら，シェルスクリプトが正常に動作するか確認してみる．各ケースで `Allrun` が実行されると計算が終了するまでシェルが実行されるので，まずは`./Allrun` の行にコメント (`#`) を挿入してシェルを実行してみる．

作業 22

```
$ vi cavityHighRe.sh  # insert comment(#)
$ chmod u+x cavityHighRe.sh
$ ./cavityHighRe.sh
```

正常にシェルが終了すれば，各レイノルズ数のケースディレクトリが自動的に作成され，`transportProperties` の `nu` の値も書き換えられているはずである．シェルの動作が確認できたら，作業 20 で挿入したコメントを外す．計算が行われていないケースディレクトリを削除し，$Re = 100$ のケースディレクトリも念のためバックアップを取っておく．そしてシェルを実行する．

作業 23

```
$ vi cavityHighRe.sh # delete comment(#)
$ cp -r cavityRe00100 cavityRe00100.bak
$ rm -rf cavityRe??000
$ ./cavityHighRe.sh &
```

後は計算が終了するのをひたすら待つ．解析が終了したケースディレクトリが出てきたら，各ディレクトリの中で `sample` を実行し，gnuplot による速度分布の比較，及び ParaView によるコンター図作成を繰り返す．`sample` の実行をシェルスクリプトに含めてもよいが，計算が正常に終了しない場合も考えられるのでここでは含めていない．作業の一例を以下に示す．

作業 24

```
$ run
$ cd cavityRe01000
$ sample -latestTime
$ cd ..
$ cp uRe00100.plt uRe01000.plt
$ vi uRe01000.plt
$ gnuplot uRe01000.plt
$ cp vRe00100.plt vRe01000.plt
$ vi vRe01000.plt
$ gnuplot vRe01000.plt
$ cd cavityRe01000
$ paraFoam
```

1.6 まとめと宿題

最終時刻 $t = 100$ s まで計算したときの速度分布を Ghia *et al.* の結果を比較すると，PENGUINITIS のページで指摘されているようにレイノルズ数が高くなると文献値からのずれが大きくなることがわかる．格子解像度が十分ではない，計算時間が十分ではない，等の理由が考えられる．結果の改善方法として以下の方法を提案しておくので，各自の宿題として (余裕があれば) チャレンジされたい．

1）計算格子を `cavityGrade` チュートリアルを参考にして不等間隔格子にしてみる．セル数は 200×200 のままにし，壁方向に格子を寄せることで改善できるか検討してみる．

2）計算時間を 200 s, 300 s, ... と増加させてみる．100 s までは解析しているので，`controlDict` を修正して継続計算を行う．

3）セル数を 300×300 や 400×400 などに増加させて計算してみる．低いレイノルズ数では粗い格子 (128×128) でもよく一致するかもしれない．

4）`icoFoam` は PISO 法による非定常解析である．十分に時間が経過したときの定常状態での比較ならば，SIMPLE 法による定常解析でも結果が得られそうに思える．`cavity` チュートリアルケースと `$FOAM_TUTORIALS/incompressible/simpleFoam/pitzDaily` などのチュートリアルケースをミックスさせて定常解析のケースを作成して解析

してみる．ただし `simpleFoam` のチュートリアルはどれも乱流解析であるので層流 (`laminar`) モデルを設定して残差の収束判定にも気を使わないといけない．

文　　献

1) PENGUINITIS OpenFOAM, http://www.geocities.jp/penguinitis2002/study/OpenFOAM/index.html, (accessed 2016).
2) U. Ghia, K. N. Ghia, and C. T. Shin, "High-Re solutions for incompressible using the Navier-Stokes equations and a multigrid method", J. Comp. Phys., 48, pp.387-411 (1982).
3) 河村哲也, 流体解析 I(応用数値計算ライブラリ), 朝倉書店 (1996).
4) ブルース・ブリン, 入門 UNIX シェルプログラミング―シェルの基礎から学ぶ UNIX の世界, ソフトバンククリエイティブ (2003).
5) 山森丈範, シェルスクリプト基本リファレンス―#!/bin/sh で，ここまでできる, 技術評論社 (2011).