


DEXCS OpenFOAM メモ

2023/5/16©岩本幸治

※ この文書では、↵は Enter またはリターンキーを表しています。

端末  で使用する最低限の UNIX コマンド

- **cd** フォルダまでのパス ↵→作業フォルダを指定するフォルダに変える. **change directory** の略. (かつてフォルダはディレクトリと呼ばれていました.) 作業フォルダは端末画面で\$マークの手前に~/Desktop のように表示されます.
 - ☆ 特別なパスの別表現
 - ✓ .→現在のフォルダ
 - ✓ ..→一つ上のフォルダ
 - ✓ ~→ホームフォルダ
 - ✓ フォルダのパスを入力するのが面倒だな, というとき **cd** (**cd** の後ろに半角スペース) と入力し, フォルダをドラッグすればそのフォルダ名がフルパスで入力される.
 - **ls**→現在のフォルダの内容を表示する. **list** の略.
 - プログラム実行中に **Ctrl + C**→プログラムを強制終了
 - プログラム実行中に **Ctrl + S**→プログラムを一時停止. 再開には **Ctrl + Q**
 - **grep -r** 検索したい文字列 . ↵→現在のフォルダ (.) の中にあるファイルで, 検索したい文字列を含むファイルを表示
-


スクリプトファイル (拡張子が **sh** (シェルスクリプト) または **py** (パイソンスクリプト) のファイル) を実行できない時の対処法

- ① 問題のファイルがあるフォルダを開いて空白領域で右クリック→「端末で開く」を選ぶ.
- ② 端末に以下のコマンドを入力する.




chmod +x *.sh ↵ または **chmod +x *.py** ↵

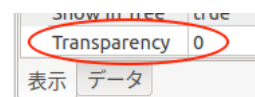
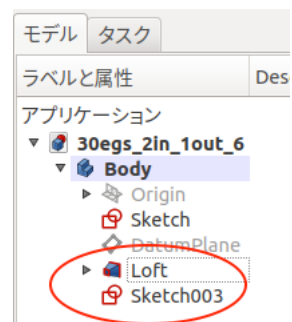
※ 「変える (**change**), モード (**mode**) を, 追加する (**±**) ように, 実行 (**execute**) 権限を」の意味







テキストエディター (gedit) 覚書

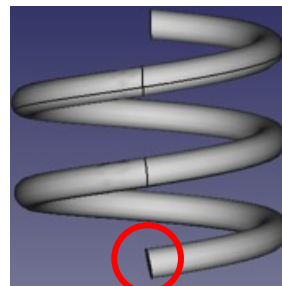
- 空白を表示する方法
 - ① 端末で `sudo apt install -y gedit-plugins` と入力し、`gedit-plugins` をインストールする。
 - ② テキストエディターを開いてメニューバー  **テキストエディター** ▾ → 設定 → プラグイン → 空白の表示をチェックする。
- 別名で保存したいときは **Shift + Ctrl + S** キーを押す。
- 字下げ(インデント)したいときは字下げしたい領域を選択後に **Tab** キーを押す。
- 字下げを外し(逆インデント)したいときは外したい領域を選択後に **Shift + Tab** キーを押す。

FreeCAD 覚書

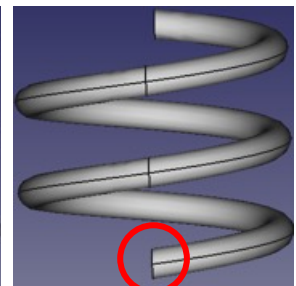
- マウスの操作 (ナビゲーションスタイルが **CAD** の場合)
 - ◇ 選択 → 左ボタン
 - ◇ 移動 → 中ボタン, あるいは **Ctrl + 右ボタン**
 - ◇ ズーム → スクロールホイール
 - ◇ 回転 → 中ボタンを押し続けてから右または左ボタン, あるいは **Shift + 右ボタン**
- 要素の表示, 非表示 ⇄ 要素を選んでスペースキー
- 新規に部品を作る時は, Part Design ワークベンチ → ボディーを作成  **ボディーを作成** から始めるよと良さげ。
- Part Design ワークベンチでは, 要素やスケッチを選択すると, 「タスク」タブで次にしそうなことを案内してくれる。
- ソリッドの面上にスケッチ ⇄ ソリッドの面を選んで Part Design ワークベンチ → 新規スケッチを作成 
 - ◇ 外部形状にリンクするエッジを作成  (SolidWorks で言う「エンティティ変換」) を使うためには, モデルビューでスケッチがエッジを作りたいソリッドよりも下にないとけない
- 断面表示 ⇄ 「表示」メニュー → 断面
- 透明度 ⇄ 要素を選んで「表示」タブ → Transparency



- ソリッドを面に分割☞ダウングレード
 - 面をまとめる☞Part ワークベンチ→複数の形状の和集合を作成
 - SolidWorks の「参照ジオメトリ」の「平面」☞Part Design ワークベンチ→新しいデータム平面を作成
- ☆ 「アクティブなボディーがありません。」というエラーメッセージが出る場合
☞参照ジオメトリを作りたい物体を選んでから、新しいボディーを作成してそれをアクティブ化
- ☆ “You selected geometries which are not part of the active body...”というメッセージが出る場合、たいていの場合「相互参照を作成」を選べば良さげ。
- スケッチを面にしたい☞スケッチを選択後、Part ワークベンチ→「パート」メニュー→ワイヤーから面を作成
 - 新しいパーツを作成しアクティブ化で作成した Part にドラッグ&ドロップすれば、body や他の物体を含めることができる。
 - Part または Part Design ワークベンチのスイープ系のユーティリティにおいて、スイープする断面が傾くのが嫌な場合は「フレネ」を利用すると良い。
 - 参考になるページ：XSim FreeCAD 使い方メモ <https://www.xsim.info/articles/FreeCAD/How-to-use-FreeCAD.html>



フレネなし



あり

変数ファイルの上の方に書かれている dimensions の[1 -1 -2 0 0 0 0]の意味→[kg m s K mol A cd]の累乗を表している。

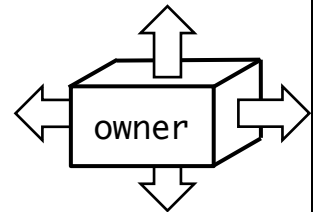
※ 上の例だったら $\text{kg}^1\text{m}^{-1}\text{s}^{-2}\text{mol}^0\text{A}^0\text{cd}^0 = (\text{kg m/s}^2)/\text{m}^2 = \text{N/m}^2 = \text{Pa}$ 。

フィールド変数の定義

名前	単位	定義
alpha.*	無次元	*に書いている名前（解析フォルダ/constant/transportProperties で決める）の流体の相の体積分率

名前	単位	定義
		<p>water と air で構成されている場合、密度や粘性係数は次式で計算される：</p> $\rho = \alpha_{\text{water}}\rho_{\text{water}} + (1 - \alpha_{\text{water}})\rho_{\text{air}},$ $\mu = \alpha_{\text{water}}\mu_{\text{water}} + (1 - \alpha_{\text{water}})\mu_{\text{air}}$
alphan	m^2/s または $\text{kg}/(\text{m}\cdot\text{s}) = \text{Pa}\cdot\text{s}$	<p>乱流温度拡散率 $\alpha_t = \frac{\kappa_t}{\rho c_p} = \frac{\nu_t}{Pr_t}$ または $\rho\alpha_t = \frac{\kappa_t}{c_p} = \frac{\mu_t}{Pr_t}$, κ_t は乱流熱伝導率, ρ は密度, c_p は定圧比熱, Pr_t は乱流プラントル数 壁面では compressible::alphanWallFunction, それ以外では calculated 境界条件を使うと良い.</p>
epsilon	m^2/s^3	<p>乱流エネルギー散逸率 $\varepsilon = \nu \frac{\partial u'_i}{\partial x_j} \frac{\partial u'_i}{\partial x_j}$ 標準 $k - \varepsilon$ モデルだと乱流動粘性係数は $\nu_t = C_\mu \frac{k^2}{\varepsilon}$, $C_\mu = 0.09$ で与えられる. 管内流の場合, $\varepsilon \equiv \frac{C_\mu^{0.75} k^{1.5}}{\text{混合距離}}$, $C_\mu = 0.09$, 混合距離 = 0.07 × 管内径ぐらいの大きさになる. #calc ディレクティブ (6 ページ) も参照のこと. 入口境界では turbulentMixingLengthDissipationRateInlet, 壁面では epsilonWallFunction 境界条件を使うと良い. http://en.wikipedia.org/wiki/Turbulence_kinetic_energy 局所平衡を仮定すると $\varepsilon = -\overline{u'_i u'_j} \frac{\partial \overline{u}_i}{\partial x_j} = \nu_t \left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i} \right) \frac{\partial \overline{u}_i}{\partial x_j}$ になる.</p>
href	m	p_rgh を計算する際の $z = 0$ に対応する基準水平面高さ
k	m^2/s^2	<p>乱流エネルギー $k = \frac{1}{2}(\overline{u'^2} + \overline{v'^2} + \overline{w'^2})$ 管内流の場合, $k \equiv \frac{3}{2} \left(0.16 \times Re^{-1/8} \times \text{平均速度} \right)^2$ ぐらいの大きさになる. #calc ディレクティブ (6 ページ) も参照のこと. 入口境界では turbulentIntensityKineticEnergyInlet, 壁面では kqRWallFunction 境界条件を使うと良い. http://en.wikipedia.org/wiki/Turbulence_kinetic_energy</p>
Ma	無次元	マッハ数
nut	m^2/s	<p>乱流動粘性係数 壁面では nutkWallFunction, それ以外では calculated 境界条件を使うと良い.</p>
nuTilda	m^2/s	<p>有効動粘性係数 (分子粘性係数 + 乱流粘性係数) Spalart-Allmaras 乱流モデルだけが使用する.</p>
omega	1/s	<p>乱流エネルギー比散逸率 ω 乱流渦の渦度に比例する量で, ε/k にも比例する. 標準 $k - \omega$ モデルだと乱流動粘性係数は $\nu_t = k/\omega$ で与えられる. $k - \varepsilon$ モデルよりも $k - \omega$ モデルの方が安定に計算できると言われている. 管内流の場合, $\omega \equiv \frac{k^{0.5}}{C_\mu^{0.25} \times \text{混合距離}}$, $C_\mu = 0.09$, 混合距離 = 0.07 × 管内径ぐらいの大きさになる. #calc ディレクティブ (6 ページ) も参照のこと.</p>

名前	単位	定義
		ージ) も参照のこと. 入口境界では turbulentMixingLengthFrequencyInlet , 壁面では omegaWallFunction 境界条件を使うと良い.
p	$\text{kg}/(\text{m}\cdot\text{s}^2) = \text{Pa}$ または $\text{m}^2/\text{s}^2 = \text{Pa}/(\text{kg}/\text{m}^3)$	圧力 (圧縮性流体や混相流のときに多い定義) または 圧力/密度 (非圧縮性流体のときに多い定義) どっちかを知りたいときは, p ファイルの dimensions を見れば良い.
p_rgh	$\text{kg}/(\text{m}\cdot\text{s}^2) = \text{Pa}$	上向きをz軸をとり, z方向に加速度 $-g < 0$ で重力がかかる (下向きに $g > 0$ で重力がかかる) とすると $p + \rho g z$. 無重力環境下で同じ流れが作り出されるとききの圧力 ※ http://www.cfd-online.com/Forums/openfoam/80454-p_rgh-1-7-a.html
phi	非圧縮性流では m^3/s 圧縮性流では kg/s	セル境界面での体積流量または質量流量 セルの owner から見て, セルから流れ出す (右図の⇒) 方向を正と定義している. この定義により, 境界では計算領域外に流れ出す方向が phi > 0 になる. http://openfoamwiki.net/index.php/OpenFOAM_guide/Table_of_common_fields
Phi	m^2/s	速度ポテンシャル Φ ($\partial\Phi/\partial x = u, \partial\Phi/\partial y = v, \partial\Phi/\partial z = w$) potentialFoam で解かれる.
pointDisplacement	m	格子点の変位
T	K	温度
Qdot	$\text{kg}/(\text{m}\cdot\text{s}^3) = \text{W}/\text{m}^3$	熱発生率
qr	$\text{kg}/\text{s}^3 = \text{W}/\text{m}^2$	ふく射による熱流束 正の方向はおそらく phi と同じ.
rho	kg/m^3	密度
rhok	無次元	$1 - \text{beta}*(T - T_{\text{ref}})$. Boussinesq 近似した時の密度を, 温度 T_{ref} における密度で除したもの. 熱膨張率 beta [1/K]や T_{ref} は constant/transportProperties で設定する.
sigma	$\text{kg}/\text{s}^2 = \text{N}/\text{m}$	表面張力係数
U	m/s	速度ベクトル



ファイル内で変数を定義できる.

例) 解析フォルダ/**0/U** ファイルの先頭で

Uinit 0.01; // 初期速度 [m/s] ←C 言語のコメントも使えます.

と書いておけば

```
inlet
{
    type fixedValue;
    value uniform ($Uinit 0 0);
}
```

の\$Uinit には 0.01 が入る. こうしておけば複数の場所で 0.01 を使う場合でも上の Uinit 0.01; の 0.01 だけを修正すればよくなる.

境界条件の値をわざわざ電卓で計算するのはメンドクサイ→#calc ディレクティブ

例) 解析フォルダ/0 の中にある U, k, epsilon, omega ファイルの上の方で

```
_Re 1.0e+04; // レイノルズ数
_D 100.0e-03; // 管内径 [m]
_nu 1.0e-06; // 動粘性係数 [m^2/s]
_U #calc "$_Re*$_nu/$_D"; // 流速 [m/s]
_L_mix #calc "0.07*$_D"; // 混合距離 [m]
_k_init #calc "1.5*pow(0.16*pow($_Re,-0.125)*$_U,2); // k の推奨
初期値 [m^2/s^2]
_epsilon_init #calc "pow(0.09,0.75)*pow($_k_init,1.5)/$_L_mix
"; // epsilon の推奨初期値 [m^2/s^3]
_omega_init #calc "sqrt($_k_init)/(pow(0.09,0.25)*$_L_mix)"; /
/ omega の推奨初期値 [1/s]
```

と書いて、その後ろで

```
internalField uniform ($_U 0 0); // 0/U ファイルで使う例
internalField uniform $_k_init; // 0/k ファイルで使う例
internalField uniform $_epsilon_init; // 0/epsilon ファイルで使う
例
```

```
internalField uniform $_omega_init; // 0/omega ファイルで使う例
```

と書けば、それぞれの値をわざわざ計算しなくてもよくなる.

※ #calc ディレクティブで書かれた場所は、計算した数値にファイルが書き換えられてしまう. 計算.py では 0_bak フォルダに元のファイルを残しておいて、最後に 0 フォルダを 0_bak フォルダで置き換えている. ただし発散などで不慮に終了した場合、置き換えられないことがある. その場合は手動で置き換える必要がある.

境界条件あれこれ

- ※ 境界条件の雛形をコピー.py を利用すると写し間違わない.
- ※ 端末を起動, cd コマンドで解析フォルダに移動した後で, 例えば速度 U に関しては
foamHelp boundary -field U ↵
と入力すれば使用可能な境界条件の一覧を知ることができる.
- ※ #includeEtc "caseDicts/setConstraintTypes"と書き込んでおけば, パッチ名と type が以下の名前のどれかで一致しているパッチに対しては, 何も書かなくて良くなる: cyclic, cyclicAMI, cyclicSlip, empty, nonuniformTransformCyclic, processor, processorCyclic, symmetryPlane, symmetry, wedge

// C で始まるもの

```
{
    type calculated;
    // 他の変数から計算可能であることを表す.
    // 壁面でない境界における nut の境界条件としてよく使われる.
    value $internalField; // 実際には使わないけど必要
}
{
    type compressible::alphatWallFunction;
    // alphas = mut/Prt (mut: 乱流粘性係数) から壁面乱流温度拡散率を計算する.
    Prt 0.85; // 乱流プラントル数
    value $internalField; // 実際には使わないけど必要
}
{
    type compressible::turbulentTemperatureCoupledBaffleMixed;
    // 境界の両側で熱流束が一致するように温度 T を決める.
    // -kappa*(T_boundary - T)/delta =
    //     -kappa_nbr*(T_nbr - T_boundary)/delta_nbr
    Tnbr T; // 隣接する場の名前. 普通は T.
    kappaMethod fluidThermo; // 境界内側の熱伝導率を指定
    // 流体側の境界なら fluidThermo, 個体側の境界なら solidThermo
    value $internalField; // 実際には使わないけど必要
}
{
```

```

type compressible::turbulentTemperatureRadCoupledMixed;
// 熱ふく射も含めて、境界の両側で熱流束が一致するように温度 T を決める.
//  $-\kappa(T_{\text{boundary}} - T)/\Delta + q_r =$ 
//  $-\kappa_{\text{nbr}}(T_{\text{nbr}} - T_{\text{boundary}})/\Delta_{\text{nbr}} - q_{r,\text{nbr}} +$ 
//  $(\Delta \rho C_p + \Delta_{\text{nbr}} \rho_{\text{nbr}} C_{p,\text{nbr}}) \cdot$ 
//  $(T_{\text{boundary}} - T_{\text{old}})/\Delta t$ 
kappaMethod fluidThermo; // 境界内側の熱伝導率を指定
// 流体側の境界なら fluidThermo, 個体側の境界なら solidThermo
thermalInertia false; // 境界付近の温度の時間的変化を考慮に入れるか
// false だと  $\Delta t = \infty$  (定常) または  $C_p = 0$  に相当する.
}
{
    type cyclic;
    // 周期境界
    // constant/polyMesh/boundary で
    // neighbourPatch を指定しないとイケない.
    // http://penguinitis.g1.xrea.com/study/OpenFOAM/cyclic/cyclic.h
    tml
}
// E で始まるもの
{
    type empty;
    // 1次元または2次元解析の時に、計算しない方向に垂直な面であることを示す.
}
{
    type epsilonWallFunction;
    // epsilon の壁面境界条件
    value $internalField; // 実際には使わないけど必要
}
{
    type externalWallHeatFluxTemperature;
    // 壁面からの熱伝達
    mode flux;
    // flux→熱流束  $q$  | power→熱量  $Q$  | coefficient→熱伝達率  $h = q/(T - T_a)$ 
    q uniform 100; // flux の時に使用
    // Q 100; // power の時に使用
}

```



```

// h 10; // coefficient の時に使用
// Ta 500; // 外部温度, coefficient の時に使用
kappaMethod fluidThermo; // 境界内側の熱伝導率を指定
// 流体側の境界なら fluidThermo, 個体側の境界なら solidThermo
value $internalField; // 実際には使わないけど必要
}
// F で始まるもの
{
    type fixedFluxPressure;
    // 速度境界条件を満足するように p_rgh を設定
}
{
    type fixedGradient;
    // こう配を gradient で決めた値にする.
    gradient uniform 1.2; // ベクトルの場合は(1.2 3.4 5.6)のように書く.
}
{
    type fixedValue;
    // value で決めた値に固定
    value uniform 1.2; // ベクトルの場合は(1.2 3.4 5.6)のように書く.
}
{
    type flowRateInletVelocity;
    // 体積流量または質量流量で流入速度を設定し,
    // 境界に平行な方向の速度は 0 にする.
    volumetricFlowRate 0.1; // 体積流量, massFlowRate とは併用できない.
    // massFlowRate 0.1; // 質量流量, volumetricFlowRate とは併用できない.
    // rhoInlet 1; // 密度, massFlowRate の場合に必要
    extrapolateProfile false;
    // true→内側と相似な速度分布で流入 | false→一様流入
    value $internalField; // 実際には使わないけど必要
}
{
    type freestreamPressure;
    // p に対する自由流入出条件. freestreamVelocity と併用する.
    // 境界垂直方向と流速方向が完全に同じ向きで

```

```

// 流入する時は zeroGradien に規定し,
// 流出する時は freestreamValue にする.
// 完全に同じでないときは, これらの間を連続的に変化させたものを使う.
freestreamValue uniform 1.0e+05;
}
{
type freestreamVelocity;
// U に対する自由流入出条件. freestreamPressure と併用する.
// 境界垂直方向と流速方向が完全に同じ向きで
// 流入する時は freestreamValue に規定し,
// 流出する時は zeroGradient にする.
// 完全に同じでないときは, これらの間を連続的に変化させたものを使う.
freestreamValue uniform (100 0 0);
}
// G で始まるもの
{
type greyDiffusiveRadiationViewFactor;
// 形態係数を利用してふく射による熱流速を決定する.
// 壁面は灰色体（ふく射率には constant/radiationProperties の中にある
// emissivity を利用?）とする.
gro uniform 0; // 外部から入るふく射による熱流束
}
// I で始まるもの
{
type inletOutlet;
// 計算領域内に流入する場合→inletValue で決めた値に設定
// 計算領域外に流出する場合→zeroGradient
inletValue uniform 0; // ベクトルの場合は(1.2 3.4 5.6)のように書く.
}
// K で始まるもの
{
type kqRWallFunction;
// 高レイノルズ数型乱流モデルにおける k, q, R の壁面境界条件
// zeroGrdient のラッパー
value $internalField; // 実際には使わないけど必要
}

```

```

// Mで始まるもの
{
    type movingWallVelocity;
    // Uに使用
    // 移動壁面の場合の noSlip 条件, 壁面が移動しなければ noSlip と同じ
    value uniform (0 0 0); // 初期速度
}

// Nで始まるもの
{
    type nutkRoughWallFunction;
    // 荒い壁面に対する nut の境界条件
    // nutkWallFunction で使っている式に,
    // 壁面荒さ (凹凸の高さ)  $K_s$ , 定数  $C_s$  による補正を加えている.
    // https://dergipark.org.tr/en/download/article-file/202910
     $K_s$  uniform 0; // 単位は m, 0 だと滑面
     $C_s$  uniform 0.5; // 0.5 - 1.0, 大きいほど荒さの影響大
    value $internalField; // 実際には使わないけど必要
}

{
    type nutkWallFunction;
    // 壁面に対する nut の境界条件, 標準的
    //  $yPlus = C_{\mu}^{0.25} \sqrt{k} y / \nu$  から格子中心の  $yPlus$  を求め,
    // 対数則領域内に格子中心があるかどうかを判断する.
    // ある場合, 対数速度分布から得られる壁面せん断応力
    //  $\tau_w = \mu \kappa yPlus / \log(E yPlus) * (u/y)$ 
    // になるように乱流粘性係数を設定する.
    // https://www.slideshare.net/fumiyanozaki96/openfoam-36426892
    value $internalField; // 実際には使わないけど必要
}

{
    type nutUWallFunction;
    // 壁面に対する nut の境界条件
    // 格子中心での速度  $u$ , 壁からの距離  $y$ , 対数速度分布から得られる関係
    //  $yPlus \log(E yPlus) = \kappa u y / \nu$  から格子中心の  $yPlus$  を求め,
    // 対数則領域内に格子中心があるかどうかを判断する.
    // ある場合, 対数速度分布から得られる壁面せん断応力

```

```

// tau_w = mu*kappa*yPlus/log(E*yPlus)*(u/y)
// になるように乱流粘性係数を設定する.
// https://www.slideshare.net/fumiyanozaki96/openfoam-36426892
value $internalField; // 実際には使わないけど必要
}
{
    type noSlip;
    // U = (0 0 0)に規定
}
// 0 で始まるもの
{
    type omegaWallFunction;
    // 壁面に対する omega の境界条件
    value $internalField; // 実際には使わないけど必要
}
{
    type outletInlet;
    // 計算領域外に流出する場合→outletValue で決めた値に設定
    // 計算領域内に流入する場合→zeroGradient
    outletValue uniform 0; // ベクトルの場合は(1.2 3.4 5.6)のように書く.
}
{
    type outletPhaseMeanVelocity;
    // alpha で示す相の平均流出流速が Umean になるように規定する.
    // 典型的な例としては, 曳航水槽による船周りの流れシミュレーションで,
    // 入口と出口の水位が同じになるようにする場合に使う.
    alpha alpha.water;
    Umean 1.2;
}
// P で始まるもの
{
    type pressureInletOutletVelocity;
    // U に使用
    // 計算領域内に流入する場合→垂直方向成分はこう配が 0,
    // 接線方向成分は tangentialVelocity のうちの接線方向成分のみ
    // 計算領域外に流出する場合→全成分でこう配が 0

```

```

    tangentialVelocity uniform (0 0 0);
    value $internalField; // 実際には使わないけど必要
}
{
    type prghPressure;
    // p_rgh に使用
    // 設定したい p の値から p_rgh を計算して設定する.
    // 対応するパッチの p には calculated を使う.
    rho rhok; // 計算で用いる密度の変数名, rho または rhok
    // p の次元が Pa の場合→rho, m^2/s^2 にの場合→rhok
    p uniform 0; // 設定したい p の値
}
// R で始まるもの
{
    type rotatingWallVelocity;
    // 速度を回転物体表面の速度と一致させる
    origin (0 0 0); // 回転中心
    axis (0 0 1); // 回転軸
    omega (0 0 1); // 右ねじが進む向きを正とした角速度 [rps]
}
// S で始まるもの
{
    type slip;
    // 非粘性流れの壁面境界条件
}
{
    type surfaceNormalFixedValue;
    // 境界に垂直な方向の速度を外向きを正として設定し, 平行方向の速度は 0 にする.
    refValue uniform -10; // 垂直方向速度
}
{
    type symmetry;
    // 対称境界, 境界が曲がっていても使える
}
{
    type symmetryPlane;

```

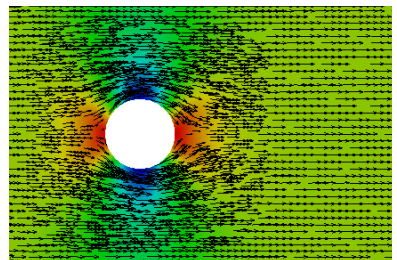
```

    // 対称境界, 完全な平面にしか使えない
}
// Tで始まるもの
{
    type totalPressure;
    // p または p_rgh に使用
    // p0 で決めた値が全圧 (p_rgh の場合は全圧 + rho*g*z) になるように設定
    p0 uniform 0;
}
{
    type turbulentIntensityKineticEnergyInlet;
    // 計算領域内に流入する場合→k = 1.5*(intensity*局所速度)^2 に規定
    // 計算領域外に流出する場合→zeroGradient
    intensity 0.05;
    value $internalField; // 実際には使わないけど必要
}
{
    type turbulentMixingLengthDissipationRateInlet;
    // 計算領域内に流入する場合→epsilon = C_mu^0.75*k^1.5/混合距離,
    // C_mu = 0.09 に規定
    // 計算領域外に流出する場合→zeroGradient
    mixingLength 0.001; // 混合距離, ふつうは 0.07*管内径ぐらいの大きさ
    value $internalField; // 実際には使わないけど必要
}
{
    type turbulentMixingLengthFrequencyInlet;
    // 計算領域内に流入する場合→omega = k^0.5/(C_mu^0.25*混合距離),
    // C_mu = 0.09 に規定
    // 計算領域外に流出する場合→zeroGradient
    mixingLength 0.001; // 混合距離, ふつうは 0.07*管内径ぐらいの大きさ
    value $internalField; // 実際には使わないけど必要
}
// Vで始まるもの
{
    type variableHeightFlowRate;
    // alpha に使用

```

```
// alpha < lowerBound の時→alpha = lowerBound,
// lowerBound < alpha < upperBound の時→zeroGradient,
// upperBound < alpha の時→alpha = upperBound
lowerBound 0.0;
upperBound 1.0;
}
// Z で始まるもの
{
    type zeroGradient;
    // こう配が 0, 境界での値 = セル中心での値にする.
}
```


potentialFoam (右図のようなポテンシャル流れを計算する) を行う際に、圧力 p の境界条件は以下の条件を満たさないとうまくいかない.




- `fixedValue`, `empty`, `symmetry`, `symmetryPlane` 以外の条件は使えない.
 - 使えない条件に対しては、速度の値を決めている境界の p には `zeroGradient`, 速度のこう配を決めている境界の p には `fixedValue` を使う.
-

与えた境界条件の要約が知りたい→`patchSummary`



※ 使い方

- ① 端末を起動して、`cd` コマンドで現在の解析フォルダに移動する.
 - ② 起動した端末で
`patchSummary` 
と入力する.
 - ③ 結果が端末内に表示される.
-

✓

ファイルを GUI 画面で比較したい→KDiff3 

※ 使い方

- ① 画面左下の  → KDiff3  をクリックする。
- ② 比較したいファイル／フォルダを A(Base)と B に指定して、OK を押す。

計算実行画面または logfile に表示される time step continuity errors は以下を表している (<https://groups.google.com/forum/#!msg/openfoam/hXeRveNQEbM/CTCM2lCifXwJ>) :

- sum local → 時間ステップ×セル体積で重みづけ平均した流量誤差の絶対値
- global → 時間ステップ×セル体積で重みづけ平均した流量誤差 (絶対値じゃない)
- cumulative → global を時間ステップごとに足し合わせていったもの

定常解収束判定は解析フォルダ/system/fvSolution の residualControl で行っており、残差 (residual) が設定値以下になったら収束と見なしている。なお、残差の定義は

$$\text{残差} = \frac{\sum_{\text{全格子点}} \left| \text{解くべき連立方程式の右辺} - \text{解くべき連立方程式の左辺} \right|}{\sum_{\text{全格子点}} \left| \text{解くべき連立方程式の右辺} \right|}$$

である。

(https://groups.google.com/forum/#!msg/openfoam/eluWBvEdnDw/RqiN1_nrXiYJ, <https://groups.google.com/forum/#!msg/openfoam/cA-xHaBZV8E/PcOdCs8H8doJ>)

発散する場合の対処法

- system/fvSolution の書き換えで出来ること
 - ☆ $k-\varepsilon$ モデルや $k-\omega$ モデルで発散する場合、system/fvSolution で計算の緩和係数を小さくする。経験的に、画面に表示される各タイムステップでの繰り返し計算の回数 (Solving for ... No Iterations ○の○の部分) が 10 より小さいと発散しにくい。

relaxationFactors // 緩和係数の設定

{

// 書いていない量の緩和係数は 1

// 時間精度を保証するために、Final が付く量の緩和係数は 1 にすべき

fields

```

{
    "plp_rgh" 0.8; // 緩和係数, 0 より大きく 1 以下にする
}
equations
{
    U 0.8; // 緩和係数, 0 より大きく 1 以下にする
    "klepsilonlomega" 0.3; // 緩和係数, 0 より大きく 1 以下にする
}
}

```

- ☆ system/fvSolution のソルバーを安定なものにする。以下が良いという報告があった。

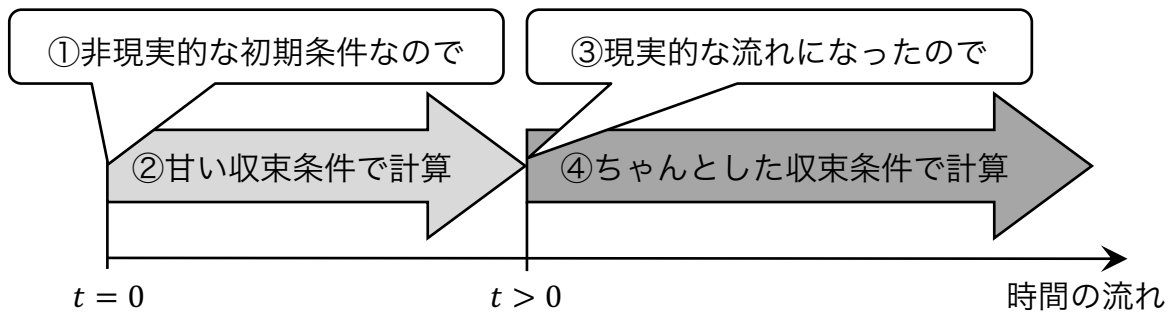
```

"plp_rgh"
{
    solver PCG;
    preconditioner DIC;
    tolerance 1e-06;
    relTol 0.01;
}
Phi // potentialFoam で求める量
{
    $p; // p と同じにしている
}
U
{
    solver PBiCG;
    preconditioner DILU;
    tolerance 1e-06;
    relTol 0.1;
}
"klepsilonlomega"
{
    solver PBiCG;
    preconditioner DILU;
    tolerance 1e-05;
    relTol 0.1;
}

```

✧ 初期に発散する場合，（止まっている流れを無限小時間で加速させる等の）実際にはありえない流れを計算していることによる，非現実的な誤差が生じてることが多い．よって，非現実的な誤差が取り除かれるまでは甘い収束条件で計算を進行する．具体的には `system/fvSolution` で `tolerance` を `0.01` や `0.001`，`relTol` を `0.1` 程度にする．

- ✓ 甘い収束条件の結果は最終結果にできない．甘い収束条件での計算はいわゆる「ならし運転」であり，その後で甘くない，ちゃんとした収束条件を使って「本計算」を行った結果のみが最終結果にできる．








- ✧ `simpleFoam`, `pisoFoam`, `pimpleFoam` で発散する場合，`system/fvSolution` で `momentumPredictor` を `no` から `yes` にする．
- ✧ `interFoam` で発散する場合，`system/fvSolution` で `nAlphaSubCycles` を増やす．
- ✧ `system/fvSolution` で `nNonOrthogonalCorrectors` を 1 以上にする．
- `system/fvSchemes` の書き換えで出来ること
 - ✧ 【空間差分の精度が悪くなるので推奨できないが，最終手段として】`system/fvSchemes` で $k-\varepsilon$ モデルや $k-\omega$ モデルの対流項の差分スキームを `bounded Gauss Upwind` にする．

```
divSchemes    // 対流項の差分スキーム
{
    ...(略)...
    div(phi,k) bounded Gauss upwind;
    div(phi,epsilon) bounded Gauss upwind;
    div(phi,omega) bounded Gauss upwind;
}
```

 - ✓ 乱流に関する方程式はモデルに過ぎないので，高い空間精度で解いても本当に正しいかどうかは分からない，という考えが背景にある．
- 他の発散回避法は「計算の安定化のための指針」http://penguinitis.g1.xrea.com/study/OpenFOAM/stabilize_calculation.html に詳しい．

ParaView覚書




- マウスの操作
 - 3次元回転→左ボタン+ドラッグ
 - 並行移動→スクロールホイールボタン+ドラッグ
 - ズーム→スクロールホイール
 - 2次元回転→Shift+左ボタン+ドラッグ
- Properties タブ→Lighting→Ambient を 1, Diffuse を 0 にするとカラーバーの色のくすみがなくなる。 (<https://www.rccm.co.jp/icem/pukiwiki/index.php?色がくすんで見える>)
 - メッシュを表示させるときは Ambient を 0, Diffuse を 1 にする。
- Properties タブ→Parts で表示させたい部分 (internalMesh だとパッチを含む全て, group/wall だと壁面パッチ全部, 他だとそれぞれの名前のパッチ) を選ぶ事ができる。
- プレゼンなどに使う資料には背景が白の方が良いが, マウスで操作する点や線や面が見えにくい時には, Properties タブ→Background で背景色を変える事ができる。
- Tools メニュー→Lock View Size Custom... で画面の大きさをいつも同じにできる。
- Add current viewpoint as custom viewpoint アイコン  で今のカメラアングルを保存することができる。保存したアングルは, アイコン  の右側にカメラアイコン  として現れる。Configure custom viewpoints アイコン  で保存したアングルを消すことができる。
- 一度作り込んだ設定を保存するためには, File→Save State..., Load State... を使うと良い。Load State Data File Options で Choose File Names を使うと, 違う OpenFOAM (または .foam) ファイルに対しても, 作り込んだ設定を適用できる。
- SpreadSheet View で  を押すと, 数値データを csv ファイルで取り出す事ができる。
- 表示する文字に上付き/下付き添字, 平方根などを使いたい場合は文字列を \$ と \$ で囲んで TeX 表記を使う。


☆ 例) $U_{in} \rightarrow U_{in}$ | $a^2 \rightarrow a^2$ | $\sqrt{3} \rightarrow \sqrt{3}$ | $\frac{1}{2} \rightarrow \frac{1}{2}$ |
 $\mathrm{U}_{in} \rightarrow U_{in}$

ParaViewでよく使うフィルタ





- ※ フィルタ内の項目が見つからない場合は, 歯車アイコン  をクリックして項目を全て展開するか, Search... ウィンドウ  を使って検索する。
-  Filters→Common→Calculator  パラメータを使って別の量を計算する。無次元化

するときなどに便利.

-  Filters→Common→Contour等値面
-  Filters→Common→Clip切断
 - ✧ Clip Type には Plane (平面), Box (直方体), Sphere (球), Cylinder (無限長の円柱), Scalar (値のしきい値) を選ぶことができる.
 - ✧ Box の Scale は計算領域全体を覆った時に になる.
 - ✧ Show Plane 等のチェックを外すと切断面を示す境界線が消える.
 - ✧ Invert のチェックで切断後に残る部分を指定できる.
-  Filters→Common→Sliceある面だけを切り出す.
 - ✧ Show Plane のチェックを外すと切り出す面を示す境界線が消える.
 - ✧ 平行な面であれば, Value Range の所で複数の面を切り出すこともできる.
-  Filters→Common→Glyphベクトル
 - ✧ Orientation→Orientation Array と Scale→Scale Array に速度ベクトルを選ぶ
 - ✧ Scale→Vector Scale Mode を Scale by Magnitude にする.
 - ✧ Scale→Scale Factor でベクトルの長さを変えられる.
 - ✧ Masking→Glyph Mode でベクトルを描く点を決める方法が変わる.
 - ✧ Glyph Source→矢印の種類を決める.
 - ✧ Glyph Type で Arrow にすれば立体の矢印, 2D Glyph にすれば2次元の矢印.
 - ✧ 2D Glyph の場合, Glyph Source→Center の1つ目のボックスの値を0.5にする.
 - ✧ 2D Glyph が見えにくい場合, Glyph Transform→Rotate の1つ目のボックスの値を90などにして, 軸周りに何度か回転させる.
-  Filters→Common→Stream Tracer流線. 詳しく言えば瞬間流線のことで, 流脈や流跡線ではない!
 - ✧ Line width で線を太くできる.
 - ✧ Seed で指定した点の座標から初めて, 新しい座標 = 古い座標 + 速度×微小時間で座標を更新させることを繰り返す.
 - ✧ Seeds→Seed Type で流線を開始する点の座標を決める.
 - ✧ High Resolution Line Source だと Line Parameters で決めた線上の点から開始する.
 - ✧ Point Source だと Sphere Parameters で決めた球内の点から開始する.
-  Filters→Common→Group Datasets複数の入力ファイルをグループにまとめる. 出来上がったグループにフィルタを作用させれば, 一回の操作でグループ内の複数のファイルにフィルタを作用させることができる.
- Filters→Temporal→Annotate Time Filter時間表示
 - ✧ Format には%fなどのC言語の書式変換指定子が使える.
 - ✧ 表示時間 = 元の時間×Scale + Shift. 無次元化に使える.


- Filters→Search...→Surface Vectors とキーボードで入力→Surface Vectors を選択☞面に平行または垂直なベクトルを作る。
 - ✧ Constraint Mode☞Parallel→平行, Perpendicular→垂直, PerpendicularScale→垂直成分のスカラー値
 - ✧ Parallel で2次流れベクトルを作ることが多い。
- Filters→Data Analysis→Integrate Variables☞積分
 - ✧ Spreadsheet View の Attribute を Cell Data にすれば, Volume (体積積分) または Area (面積積分) が表示されるので, 値を Volume または Area で割れば, 体積平均または面積平均が得られる。割り算が面倒なら, Divide Cell Data By Volume のチェックを入れれば, 表示される値が体積平均または面積平均になる。
 - ✧ Attribute を Point Data にした時の数値が何を意味するかは不明。
- Filters→Search...→Transform とキーボードで入力→Transform を選択☞変形
 - ✧ Transform に Transform を選ぶと, 拡大 (Scale) →原点まわりの回転 (Rotate, 単位は度) →移動 (Translate) の順番で変形する。Scale は座標の無次元化に使える。
 - ✧ Transform に RotateAroundOriginTransform を選ぶと, 任意の点まわりに回転させることができる。
- Filters→Search...→Gradient とキーボードで入力→Gradient を選択☞こう配
 - ✧ Scalar Array にベクトルを選び, 歯車アイコンをクリックして項目を全て展開, Compute Vorticity や Compute Q Criterion にチェックをつけると, 渦度やこう配テンソルの第2不変量 (Q 値) が計算できる。Q 値が正, 負の領域は, それぞれ渦, せん断が支配的である。

ParaViewで複数の結果を同時に表示させる方法

- ① 右上の Split Vertical  または Split Horizontal  →Render View を選ぶ。
- ② (異なる解析フォルダの結果を表示させたい場合のみ) Open  メニューを選択し, 表示させたい解析フォルダ内にある拡張子.OpenFOAM のファイルを選ぶ。
 - ✓ 拡張子.OpenFOAM のファイルがない場合は, 端末  で解析フォルダに移動し, paraFoam -touch を実行する。
- ③ ①で作ったそれぞれの View に結果を表示させる。青枠で囲まれた View に表示されることに注意。
- ④ 片方の View を右クリック→Link Cameraを選択→もう片方の View をクリックする。
(参考: <https://www.rccm.co.jp/icem/pukiwiki/index.php?%E7%94%BB%E9%9D%A2%E3%81%AE%E5%90%8C%E6%9C%9F>)

ParaView によるアニメーションの作り方

gif ファイルで保存したい場合（こっちの方が簡単）

- ① File メニュー→Save Animation を選ぶ。
- ② Save Animation ダイアログが現れる。Files of type は PNG images (*.png)にする。ここでは、例として File name に aaa を指定したとする。
- ③ Save Animation Options ダイアログが現れる。②で指定したファイル名の後ろに、Sif fix format で指定したフォーマットで連番がつく。デフォルトの「.%04d」の場合、②で指定したファイル名の後ろにピリオドがつき、その後ろに 0 埋めで 4 桁の整数がつく。例の場合、aaa.0000.png, aaa.0001.png, ... というファイル名で保存される。
- ④ 端末  を開き、③で保存した png ファイルがある場所へ移動する。
- ⑤ 以下のコマンドを実行する。

```
convert -delay <1/100 秒単位のフレーム間隔> <②で指定したファイル名>.[0-9]*.png <保存したい gif ファイル名>.gif
```

「.[0-9]*.png」はピリオドの後ろに 0 から 9 の数字のどれかがつき、その後ろに任意の文字 0 文字以上ついて、.png で終わる文字列を表す。UNIX の正規表現を使っている。例の場合で、フレーム間隔を 50/100 秒、aaa.gif という gif ファイル名にしたい場合は、以下のようになる。




```
convert -delay 50 aaa.[0-9]*.png aaa.gif
```


- ⑥ 生成された gif ファイルは、PowerPoint に埋め込むことができる。
- ⑦ ③で保存した png ファイルは捨てても良い。

m4v ファイルで保存したい場合（gif ファイルが PowerPoint で埋め込めない場合）

- ① File メニュー→Save Animation を選ぶ。
- ② Save Animation ダイアログが現れる。Files of type は Ogg-Theora video files (*.ogv)にする。
- ③ Save Animation Options ダイアログが現れる。Frame Rate は 1 秒間あたりのフレーム数。多いほど早く動く。

※ ogv ファイルを再生してみた時に画像が歪んだり、砂嵐のようになったりする時は画面サイズが大きすぎる。Tools メニュー→Lock View Size Custom...で画面の大きさを、例えば 640×480 pixel² にすれば良い。

- ④ 画面左下の  →HandBrake  をクリックする。
- ⑤  Open Source ボタンを押して、③で保存した ogv ファイルを開く。
- ⑥ Summary タブにして、
 - Preset が Official > General > Fast1080p30 になっていること

- Align A/V Start だけにチェックが入っていること
 - Format が MPEG-4 (avformat) になっていること
- を確認する.
- ⑦ Save As: にファイル名 (拡張子は.m4v のまま), To: に保存先のフォルダを指定する.
 - ⑧  Start Encoding ボタンを押して, 終了するまで待つ.
 - ⑨ 生成された m4v ファイルは, PowerPoint に埋め込むことができる.

simpleFoam の計算手順

※ residualControl, momentumPredictor, nNonOrthogonalCorrectors は system/fvSolution の SIMPLE ブロック内で設定される.

残差が residualControl で規定した値より小さくなるまで以下を繰り返す.

{

momentumPredictor が yes の時に以下を行う.

{

運動方程式

$$U(\nabla \cdot U) = -\nabla p + \nu_{eff}(\nabla U + \nabla U^T) \quad (1)$$

を速度 U について解く. U 以外の項は固定している.

// Solving for Ux, Initial residual ...

}

// 運動方程式を離散化して, 非対角項を右辺に H としてまとめる.

$$// \quad U = \frac{H}{A} - \frac{\nabla p}{A} \quad (A \text{ は対角項, } H \text{ はそれ以外の速度を含むベクトル}) \quad (2)$$

以下を nNonOrthogonalCorrectors + 1 回繰り返す.

{

式(2)を連続の式

$$\nabla \cdot U = 0 \quad (3)$$

に代入した式 (圧力方程式)

$$\nabla \cdot \left(\frac{\nabla p}{A} \right) = \nabla \cdot \left(\frac{H}{A} \right) \quad (4)$$

を p について解く. ただし, p の非直交補正に関する項は最新の値で固定している.

// Solving for p, Initial residual ...

}

体積流量 ϕ を更新して, [連続の式の誤差を評価する](#).

```
// time step continuity errors : sum local ...
```

式(2)で速度 U を更新する.

乱流に関する式を解く.

```
}
```

pisoFoam の計算手順

※ momentumPredictor, nCorrectors, nNonOrthogonalCorrectors は system/fvSolution の PISO ブロック内で設定される.

規定した終了条件になるまで以下を繰り返す.

```
{
```

momentumPredictor が yes の時に以下を行う.

```
{
```

運動方程式

$$\frac{\partial U}{\partial t} + U(\nabla \cdot U) = -\nabla p + \nu_{eff}(\nabla U + \nabla U^T) \quad (1)$$

を速度 U について解く. U 以外の項は固定している.

```
// Solving for Ux, Initial residual ...
```

```
}
```

```
// 運動方程式を離散化して, 非対角項を右辺に  $H$  としてまとめる.
```

$$// \quad U = \frac{H}{A} - \frac{\nabla p}{A} \quad (A \text{ は対角項, } H \text{ はそれ以外の速度を含むベクトル}) \quad (2)$$

以下を nCorrectors 回繰り返す.

```
{
```

以下を nNonOrthogonalCorrectors + 1 回繰り返す.

```
{
```

式(2)を連続の式

$$\nabla \cdot U = 0 \quad (3)$$

に代入した式 (圧力方程式)

$$\nabla \cdot \left(\frac{\nabla p}{A} \right) = \nabla \cdot \left(\frac{H}{A} \right) \quad (4)$$

を p について解く。ただし、 p の非直交補正に関する項は最新の値で固定している。

※ `nNonOrthogonalCorrectors + 1` 回目に行う圧力方程式(4)の計算は、`system/fvSolution` の `pFinal` で設定した条件で行われる。

```
// Solving for p, Initial residual ...  
}
```

体積流量 ϕ を更新して、[連続の式の誤差を評価する](#)。

```
// time step continuity errors : sum local ...
```

式(2)で速度 U を更新する。

```
}
```

乱流に関する式を解く。

```
}
```

`pimpleFoam` の計算手順

※ `residualControl`, `nOuterCorrectors`, `momentumPredictor`, `nCorrectors`, `nNonOrthogonalCorrectors`, `turbOnFinalIterOnly` は `system/fvSolution` の `PIMPLE` ブロック内で設定される。

規定した終了条件になるまで以下を繰り返す。

```
{
```

残差が `residualControl` で規定した値より小さくなるまで、あるいは `nOuterCorrectors` 回以下を繰り返す。

```
{
```

`momentumPredictor` が `yes` の時に以下を行う。

```
{
```

運動方程式

$$\frac{\partial U}{\partial t} + U(\nabla \cdot U) = -\nabla p + \nu_{eff}(\nabla U + \nabla U^T) \quad (1)$$

を速度 U について解く。 U 以外の項は固定している。

※ `nCorrectors` 回目に行う運動方程式(1)の計算は、`system/fvSolution` の末尾に `Final` がつく部分で設定した条件が使われる。

```
// Solving for Ux, Initial residual ...
```

```
}
```

```
// 運動方程式を離散化して、非対角項を右辺に $H$ としてまとめる.
```

```
//  $U = \frac{H}{A} - \frac{\nabla p}{A}$  ( $A$ は対角項,  $H$ はそれ以外の速度を含むベクトル) (2)
```

```
以下を nCorrectors 回繰り返す.
```

```
{
```

```
    以下を nNonOrthogonalCorrectors + 1 回繰り返す.
```

```
    {
```

```
        式(2)を連続の式
```

```
         $\nabla \cdot U = 0$  (3)
```

```
        に代入した式 (圧力方程式)
```

```
         $\nabla \cdot \left( \frac{\nabla p}{A} \right) = \nabla \cdot \left( \frac{H}{A} \right)$  (4)
```

```
        を $p$ について解く. ただし,  $p$ の非直交補正に関する項は最新の値で固定している.
```

```
        ※ nCorrectors 回目かつ nNonOrthogonalCorrectors + 1 回目  
        に行う圧力方程式(4)の計算は, system/fvSolution の末尾に Final  
        がつく部分で設定した条件が使われる.
```

```
        // Solving for p, Initial residual ...
```

```
    }
```

```
    体積流量 phi を更新して, 連続の式の誤差を評価する.
```

```
    // time step continuity errors : sum local ...
```

```
    式(2)で速度 $U$ を更新する.
```

```
}
```

```
乱流に関する式を解く.
```

```
※ turbOnFinalIterOnly が yes の場合, nCorrectors 回目の時だけ解く.
```

```
※ nCorrectors 回目に行う乱流に関する式の計算は, system/fvSolution  
の末尾に Final がつく部分で設定した条件が使われる.
```

```
}
```

```
}
```

参考になるページ

- オープン CAE コンサルタント OCSE^2 (元デンソー 野村悦治さん) http://mogura7.zenno.info/~et/wordpress/ocse/?page_id=33
- PENGUINS (日鐵プラント設計 春日悠さん) <http://penguinitis.g1.xrea.com/study/OpenFOAM/index.html> または <http://penguinitis.g1.xrea.com>
- 株式会社ソフトフローの技術広場の OpenFOAM ユーティリティ https://www.softflow.jp/tech_category/openfoam-utilities/