#54 research / OpenFOAM /

OpenFOAM回転体関係例題への取り組み

#openfoam

Created by nakagawa_s 2019-07-16 19:01:15 +0900 Updated by nakagawa_s 2019-12-19 17:08:23 +0900

OpenFOAM-v1812の回転体関係チュートリアル

OpenFOAM-v1812の標準例題の中で，次のような回転体に関わるものがある。この文書では，これら例題の特徴，使われる機能などを確認する。

tutorial	solver	steady	絶対速度	相対速度	移動メッシュ	そのほか
mixerVessel2D	simpleFoam	steady	yes	partly	no	MRF
mixerVesselAMI2D	pimpleFoam laminar	unsteady	yes	no	yes	dynamicMeshDict cyclicAMI
mixer	SRFSimpleFoam	steady	not solved	yes	no	cyclic Urel SRFVelocity
rotor2D	SRFPimpleFoam	unsteady	not solved	yes	no	Urel SRFFreestreamVelocity

事前学習

回転体関係チュートリアルの実施前に，それらの例題で利用されているテクニックなどについて，整理する。

development for the incompressible Navier-Stokes formulation in the rotating frame

https://openfoamwiki.net/index.php/See_the_MRF_development

https://openfoamwiki.net/index.php/Sig_Turbomachinery_MRF_Library

https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Rotating_reference_frame.html

Algebraic tensor operations in OpenFOAM

Operation of vectors	**mathematical description**	**OpenFOAM**
Inner product	a・b	a & b
Outer product	a b	a * b
Cross product	a×b	a ^ b

回転系で働く慣性力（見かけの力）

遠心力　centrifugal force

回転の中心から外に向かって働く力。向心力と釣り合う。向心力は慣性系においても回転座標系においても作用するのに対し、遠心力は回転座標系においてのみ作用する。

回転中心からの回転座標系における位置を r とし、回転座標系の慣性系に対する角速度を ω とするとき、遠心力は Fcf は次式で求められる。

Fcf = m ω x (ω x r) = m ω ^2 r - m ω (ω・r)

mathematical description	**ω x (ω x r)**
Description in OpenFOAM	omega_ ^ (omega_ ^ (mesh_.C() - origin_))

コリオリカ Coriolis force

回転座標系上で移動した際に移動方向と垂直な方向働く力，転向力（てんこうりょく）とも呼ばれる。

力の大きさは，移動速度に比例する。回転座標系での速度 v' ，回転軸の方向の単位ベクトル n とすると，次式で求められる。
Fcr = -2 m ω n x v'

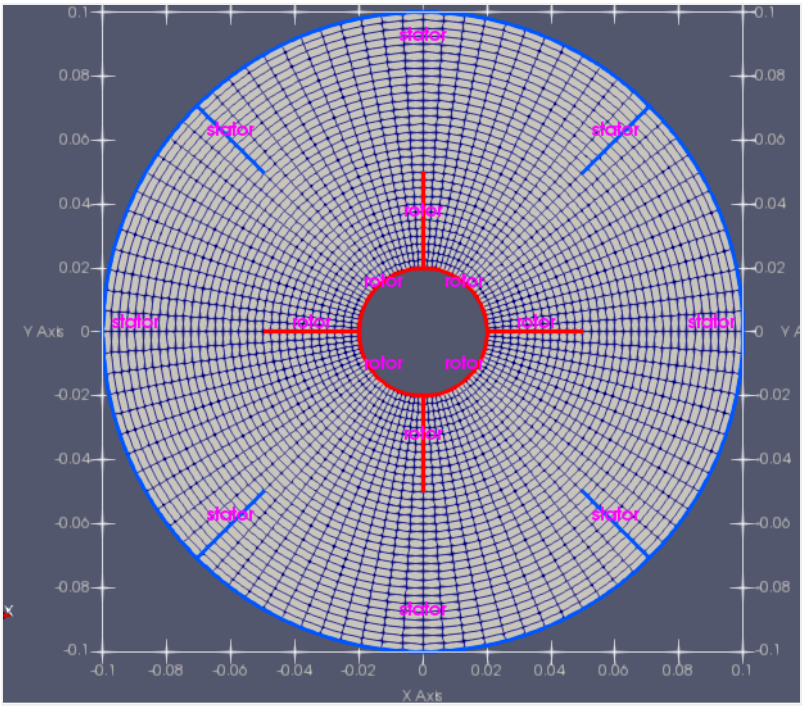
mathematical description	**-2 ω x v'**
Description in OpenFOAM	2.0 * omega_ ^ Urel

オイラーカ Euler force

回転の角加速度に伴って生じる慣性力である。回転速度が一定でない場合に考える。
Feuler = - m (d ω /dt) x r

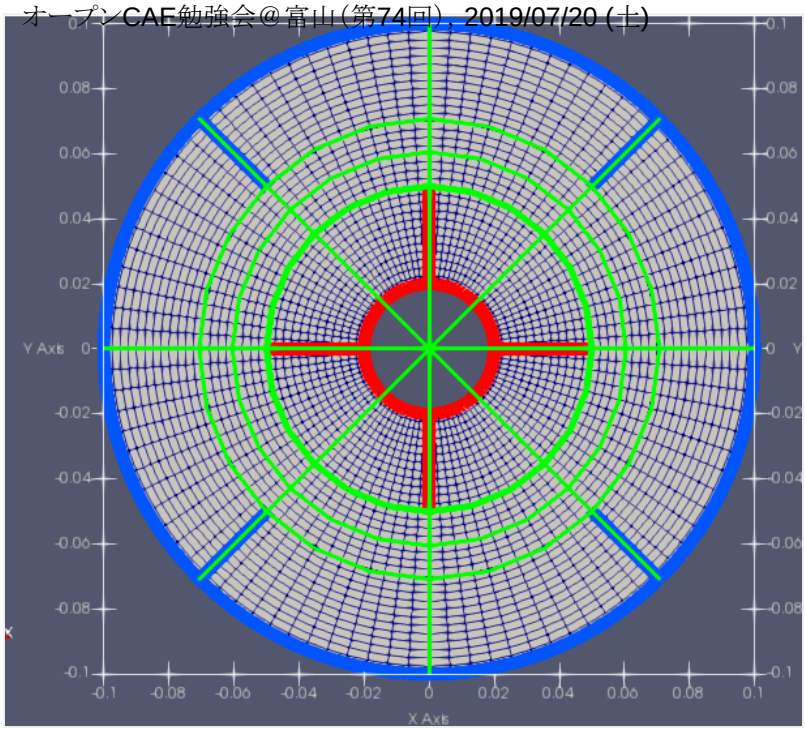
m4マクロ言語プロセッサによる複雑なblockMeshDictの生成

mixerVessel2D および mixerVesselAMI2D では，形状およびメッシュを利用する。下記のようなドーナッツ状の計算領域をblockMeshで作成する。きれいなメッシュを作成するため，かなり手の込んだblockMeshDictを用意する必要がある。この設定ファイルを効率的に作成・修正するため，これらの例題では**m4マクロ言語プロセッサ**を利用している。



m4 [1] とは，文字列の置き換えなどを実行するマクロ言語プロセッサである。system/blockMeshDict.m4ファイルに書かれた定義から，blockMeshDictを作成している。複雑な形状をblockMeshで作成するために，座標の計算等をプログラム的に実施している。

blockMeshDict.m4 では，ハブの直径，rotor翼先端の半径，stator翼先端の半径などを変数として設定している。これらの値を使って節点座標を計算し，下記のような32個のブロックを作成している。



なお，blockMeshDictが存在する状態で， `paraFoam -block` を実行すると，ブロック構成等が確認できる。

例題ごとの内容確認

incompressible/pimpleFoam/laminar/mixerVesselAMI2D/

使われる機能，特徴など

この例題では，rotor領域内のセルとrotorパッチを実際に回転させる。考え方としては，現実の現象をそのまま再現するものであり，シンプルである。しかし，静止部分と回転部分が共存するため，モデル作成には工夫が必要となる。

回転するセルと，静止したセルとの間は，cyclicAMIというタイプのpatchを作成する。AMIでは，同じ位置に2枚の重なったpatchが存在することになる。

- 境界条件：Cyclic Arbitrary Mesh Interface (AMI)
    - https://www.openfoam.com/documentation/guides/latest/doc/guide-bcs-coupled-cyclic-ami.html

Allrunによる実行内容

Allrunを解読すると，下記が順に実行されている。計算負荷がやや大きく，並列計算を実行している。並列数は，system/decomposeParに記述されている。

sh

```
m4 < system/blockMeshDict.m4 > system/blockMeshDict
blockMesh
topoSet
decomposePar
mpirun -np 4 pimpleFoam -parallel
reconstructPar
```

メッシュ生成

計算対象形状やメッシュは，先に示した通り，mixerVessel2D例題と同一である。

円の内側の16個のブロックには，rotorという名前がつけられている。この円形領域の外周には，AMI1という名前のpatchを作成する。

円の外側の16個のブロックには名前はない。この内周部には，AMI2という名前のpatchが作成される。

つまり，AMI1 および AMI2 という2つのpatchを同じ位置に作成している。

AMIの設定

topoSetでは，system/topoSetDictを読み込み，"AMI" と名付けた faceSet を作成する。blockMeshで作成した2つのpatches（AMI1，AMI2）の2つを元にしている。

topoSetDict

```
actions
(
    {
        name    AMI;
        type    faceSet;
        action  new;
        source  patchToFace;
        patch   "AMI.*";
    }
);
```

作成された set の情報は，constant/polyMesh/sets/に保存される。paraFoamコマンドでparaviewを起動し，"With Sets" にチェックを入れて有効にすることで，可視化して確認することができる。

cyclicAMI境界条件を使用するとき，boundaryファイルにおいて，ペアとなるパッチ面の指定や，そのパッチ面の位置関係などを指定する。この例題では，blockMeshでのpatch作成時に記載した情報が，boundaryファイルに書かれていることがわかる。メッシュ作成時より後のタイミングでこれらの情報を追加したい場合には，createPatchユーティリティを使用することができる。

各変数のファイルにおいて，cyclicAMIタイプを指定する。

constant/polyMesh/boundary

```
<patchName>
{
    type            cyclicAMI;
    neighbourPatch  <coupled patch name>;
    transform       <transform type>;
    ...
}
```

Uなど

```
<patchName>
{
    type            cyclicAMI;
}
```

回転の設定: dynamicMeshDict

回転は，constant/dynamicMeshDictで指定する。z軸周りの正方向に回転している。

constant/dynamicMeshDict

```
dynamicFvMesh   dynamicMotionSolverFvMesh;
motionSolver    solidBody;
cellZone        rotor;
solidBodyMotionFunction  rotatingMotion;
origin        (0 0 0);
axis          (0 0 1);
omega         6.2832; // rad/s
```

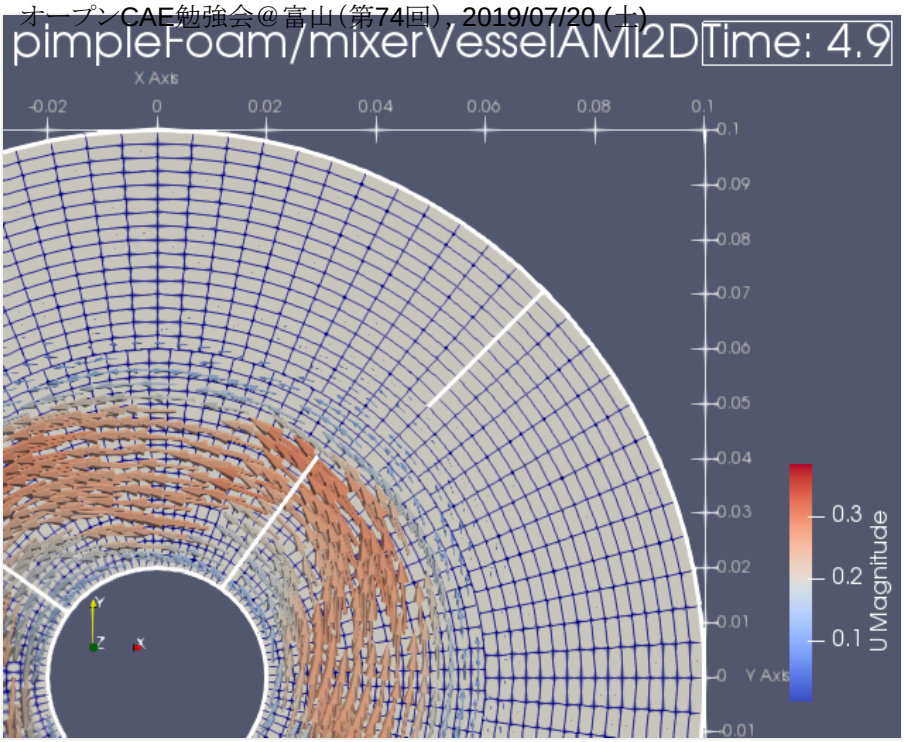
Uの境界条件で，movingWallVelocityが与えられていることを確認する。

0/U

```
    rotor
    {
        type            movingWallVelocity;
        value           uniform (0 0 0);
    }
```

計算実行

可視化



patch/rotor, patch/stator, patch/back だけを読み込んで，羽が回転する様子を確認する。

注意：paraviewでアニメーション再生しても羽の位置が動かない場合がある。このときは，VTK polyhedraオプションの横にある選択肢から，no caching オプションを選ぶことで，修正される。

incompressible/simpleFoam/mixerVessel2D

使われる機能，特徴など

- m4マクロ言語プロセッサによる複雑なblockMeshDictの生成，ならびに，blockMeshによるドーナッツ状メッシュの生成
- simpleFoam
- MRF

Allrunによる実行内容

Allrunを見る。解読すると，下記を順に実行することになる。

sh

```
m4 < system/blockMeshDict.m4 > system/blockMeshDict
blockMesh
simpleFoam
```

メッシュ生成

m4[1]マクロ言語プロセッサにより，system/blockMeshDict.m4ファイルに書かれた定義から，blockMeshDictを作成している。

blockMeshDictのblock部分を見ると，rotorという名前を指定したブロックが複数作成されていることを確認できる。これらのblockに該当する領域は，rotorという名前のついたcellZoneとなる。constant/polyMesh/cellZoneファイルに記録される。

c++

```
blocks
(
    // block0
    hex (0 2 13 12 48 50 61 60)
    rotor
    (12 12 1)
    simpleGrading (1 1 1)
```

次の図が計算領域，壁面境界（rotorとstator），中心軸周りに設定されたrotor領域（セルゾーン；水色部分）を示す。

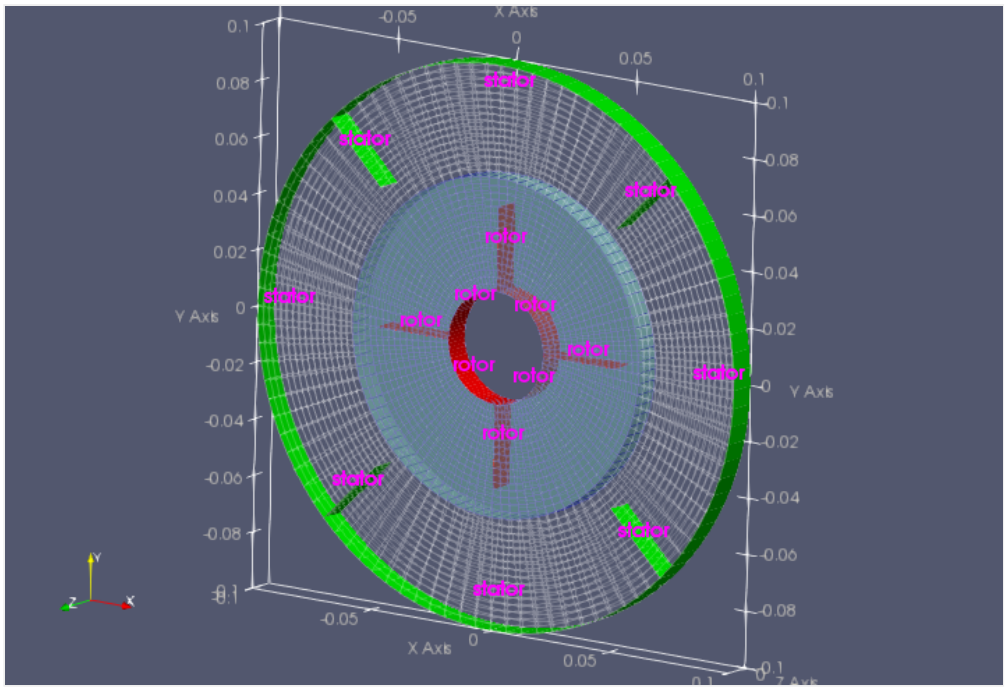


Fig. Simulation model

MRFの設定

constant/MRFPropertiesファイルの内容を確認する。セルの一部（cellZone）に rotor という名前がつけられており，その部分に回転を与えることが指示されている。この場合，グローバル座標の原点を通るz軸の周りに，104.72 rad/s = 16.7回転/s = 1000回転/分 の回転を設定している。

MRFProperties

```
MRF1
{
    cellZone    rotor;
    active      yes;
    // Fixed patches (by default they 'move' with the MRF zone)
    nonRotatingPatches ();
    origin    (0 0 0);  //座標の原点
    axis      (0 0 1);  //軸ベクトル
    omega     104.72;   //角速度 rad/s；1回転 = 360 deg = 2PI rad = 6.28 rad
}
```

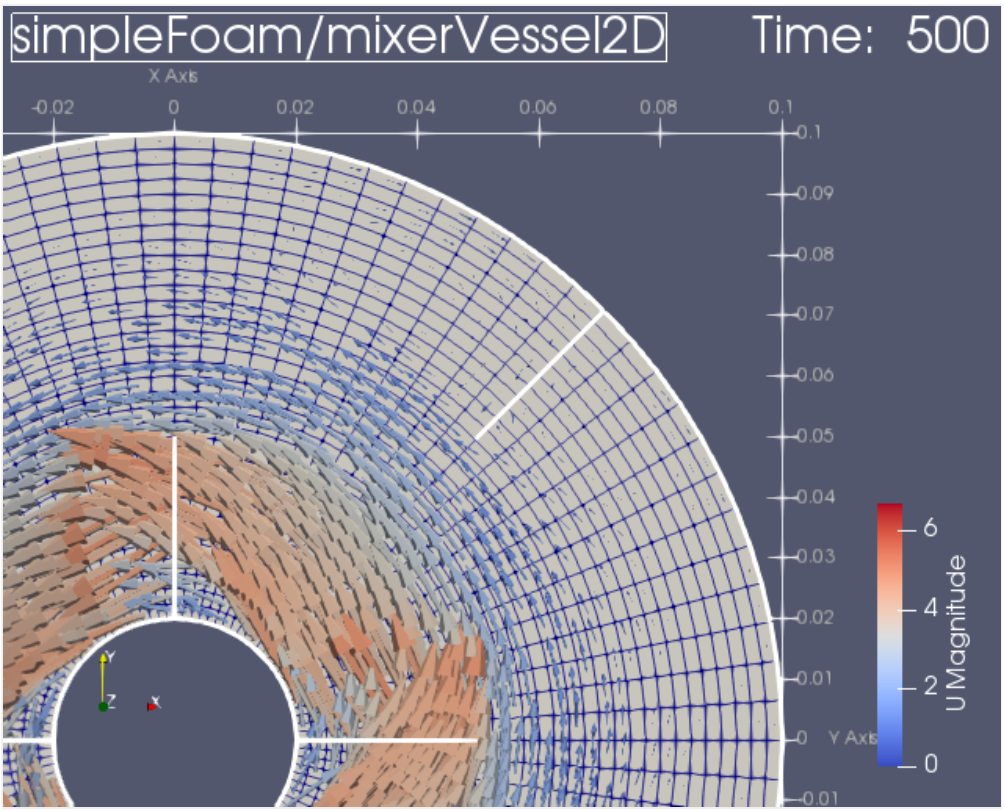
可視化

paraFoam コマンドを実行し，計算領域やメッシュを確認する。patchごとに読み込み，その位置を確認する。

Propertiesタブ上部において，with Zone にチェックを入れて有効にすると，Mesh Parts のなかに cellZone/rotor が表示される。この部分だけを読み込むと，回転領域が確認できる。

計算実行

この例題は，simpleFoamによる定常計算を実行している。



中心のrotor面およびrotor領域が静止している（ように考えている）が，実際には，上図の反時計回り(z軸の正の回転方向)に回転している。外側のstator面は静止している。

回転部で定常な流れが生じており，そのなかで，羽が上手の位置関係になったときの流れを表していると考えるとよいのか？本来，静止翼と回転翼の位置関係は時々刻々と変化するはずであり，この形状・モデルでの適用がふさわしくないのかもしれない。

incompressible/SRFSimpleFoam/mixer/

使われる機能，特徴など

計算領域全体が回転している状態を考えている。計算は回転系で実施し，遠心力やコリオリカを付与する。定常計算。

- SRF, SRFSimpleFoam

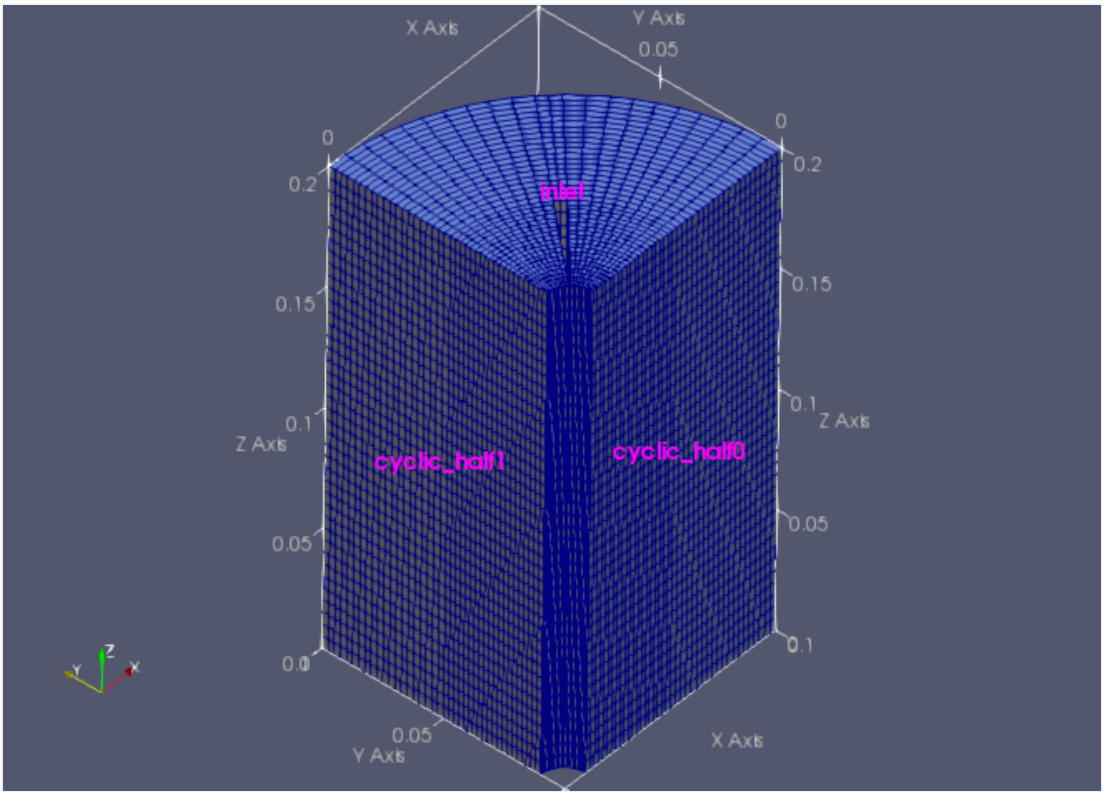
静止した円管の中心部にかくはん翼が存在し，翼および中心軸が回転している。計算領域は，周期境界条件（cyclic）を利用して，円管の4分の1の部分だけとしている。円管の入り口から，軸方向に10m/sの一様流が流入する。

実行内容

sh

```
blockMesh
SRFSimpleFoam
```

メッシュ生成



回転の設定: constant/SRFProperties

回転に関する設定は，constant/SRFPropertiesファイルに記述する。a (global) axis and revolutions-per-minute [rpm] を指定する。

constant/SRFProperties

```
SRFModel        rpm;
origin          (0 0 0);
axis            (0 0 1);
rpmCoeffs
{
    rpm         1000;
}
```

境界条件でSRFVelocityを指定すると，relativeスイッチを使用して，その面の回転(yes)・非回転(no)を変更できる。noSlipなどとすると，静止壁（回転系に対しての静止した壁→SRF条件で回転）となる。

Urel boundaryField 抜粋

```
boundaryField
{
    inlet
    {
        type            SRFVelocity;
        inletValue      uniform (0 0 -10);
        relative        no;
        value           uniform (0 0 0);
    }
    outlet
    {
        type            pressureInletOutletVelocity;
        value           $internalField;
    }
    innerWall
    {
        type            noSlip;
    }
    outerWall
    {
        type            SRFVelocity;
        inletValue      uniform (0 0 0);
        relative        no;
        value           uniform (0 0 0);
    }
    cyclic_half0
    {
        type            cyclic;
    }
    cyclic_half1
    {
        type            cyclic;
    }
}
```

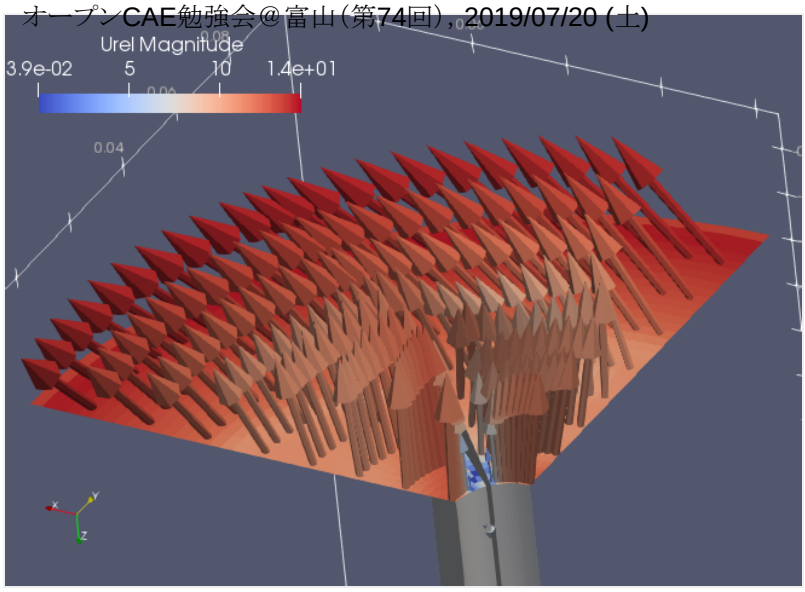
計算実行

可視化

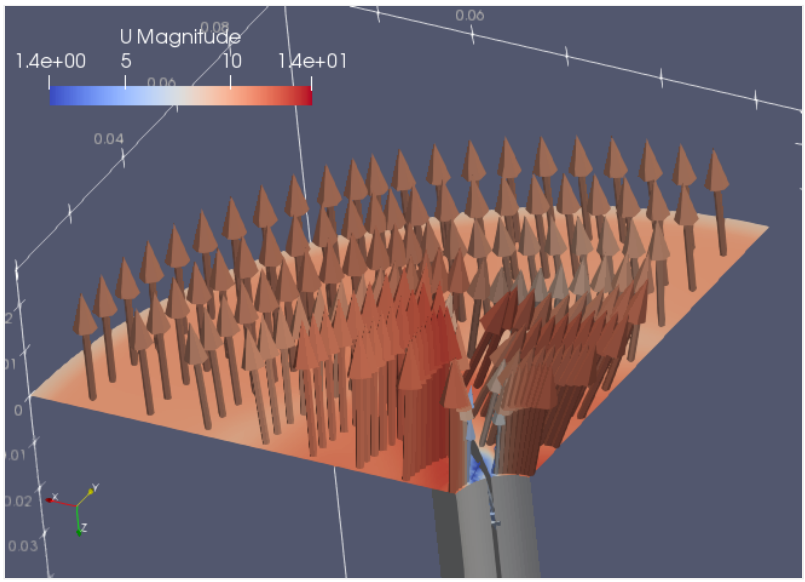
出口（outlet）での速度ベクトルを次に示す。下記の図において，z軸は下向きが正である。回転軸を (0 0 1) と与えているため，図では時計回りに中心部のかくはん翼が回転している。

相対速度 Urel では，翼が時計回りに回転するとき，静止した円筒壁面付近の流体は，回転から取り残される（回転とは逆方向に動く）ように見える。そのため，出口でのベクトルが反時計回り方向に傾いている。

一方，絶対速度 U では，出口での速度ベクトルの多くは，主流（z軸マイナス方向）を向く。翼の周りの流体は，翼の影響を強く受けて時計回り方向へ傾いている。



↑Fig. 回転系で観測した速度（相対速度）Urel



↑Fig. 静止系で観測した速度（絶対速度）U

計算負荷削減のために、対称性を活かした1/4モデルを使っている。結果を表示するときは，全体を使いたい場合がある。そのためには，paraview の Transform フィルタを使用する。今回のような軸対称の場合は Rotation 欄に必要な角度を入力する。

incompressible/SRFPimpleFoam/rotor2D

使われる機能，特徴など

形状は，mixerVessel2Dなどに近い。ただし，外周部の円筒面は壁ではない。十分広い流体領域の一部である円筒領域と取り出したものと考えられる。

周囲の速度に SRFFreestreamVelocity 境界条件を使用する。これによって，UrelにはSRF領域の回転に応じて流向が変化する流れが与えられるが，Uとしては1方向の定常流れが付与されることになる。

周囲流の流入角度が変化することになるため，非定常計算を実施する。

- SRF
- SRFPimpleFoam
- SRFFreestreamVelocity 境界条件

Allrunによる実行内容

sh

```
m4 < system/blockMeshDict.m4 > system/blockMeshDict
blockMesh
SRFPimpleFoam
```

メッシュ生成

メッシュ生成後のboundaryファイルを確認する。freestream境界はpatchタイプである。

boundaryファイル抜粋

```
    freestream
    {
        type            patch;
        nFaces          96;
        startFace       5784;
    }
```

0/Urelファイルで freestream の境界条件は，SRFFreestreamVelocity と指定されている。

0/Urel

```
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);
boundaryField
{
    rotor
    {
        type            noSlip;
    }
    freestream
    {
        type            SRFFreestreamVelocity;
        UInf            (1 0 0);
        value           uniform (0 0 0);
    }
    "(front|back)"
    {
        type            empty;
    }
}
```

パッチ面での速度U_p は，swept angle theta [rad]から計算される。　参考：SRFFreestreamVelocityFvPatchVectorField Class Reference

```
U_p =  cos(theta)*UInf_ + sin(theta)*(srf.axis() ^ UInf_)  - srf.velocity(patch().Cf())
```

SRFFreestreamVelocityFvPatchVectorField.H

```
    \f[
        U_p = cos(\theta) U_{Inf} + sin(\theta) (n \times U_{Inf}) - U_{p,srf}
    \f]

    where
    \vartable
        U_p     | patch velocity [m/s]
        U_{Inf} | free stream velocity in the absolute frame [m/s]
        \theta  | swept angle [rad]
        n       | axis direction of the SRF
        U_{p,srf} | SRF velocity of the patch [m/s]
    \endvartable
```

回転の設定: constant/SRFProperties

回転に関する設定は，constant/SRFPropertiesファイルに記述する。a (global) axis and revolutions-per-minute [rpm] を指定する。

constant/SRFProperties

```
SRFModel        rpm;
origin          (0 0 0);
axis            (0 0 1);
rpmCoeffs
{
    rpm         60;
}
```

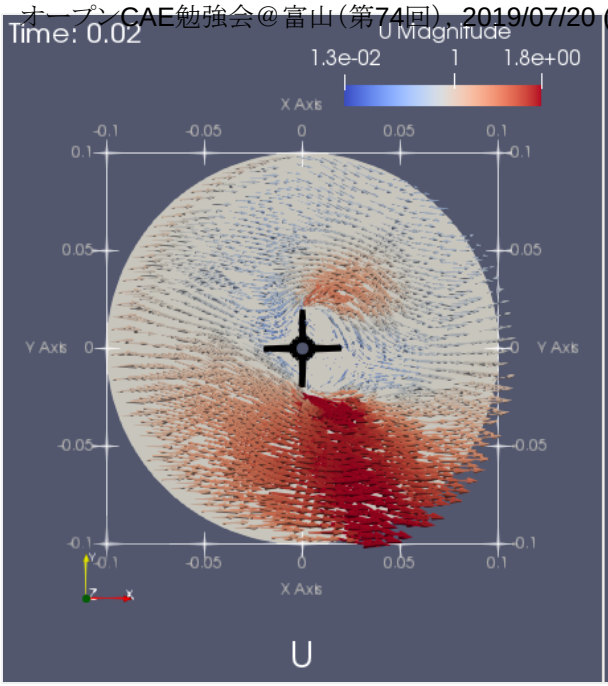
計算実行

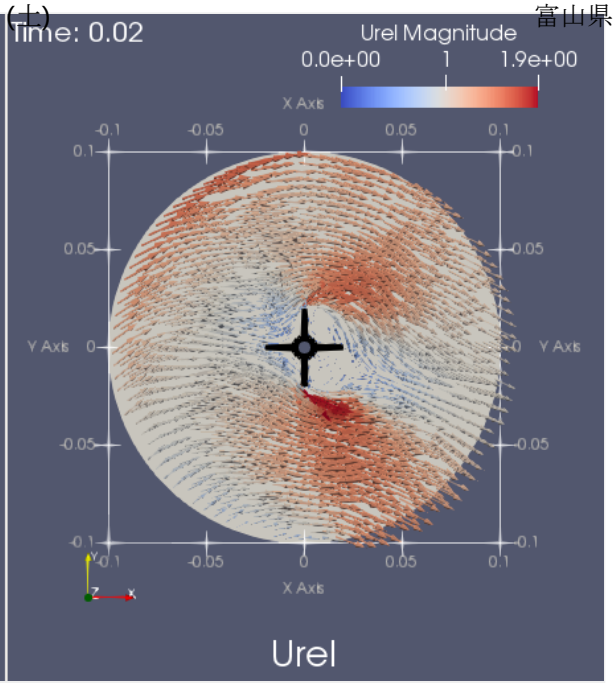
可視化

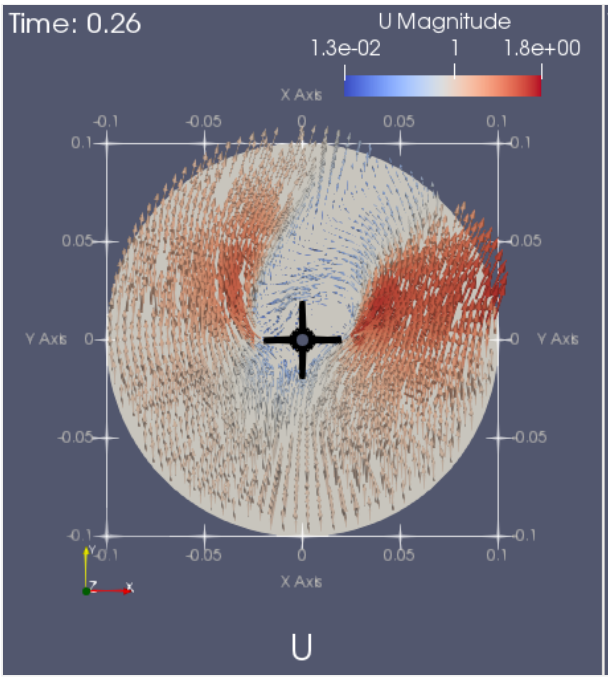
絶対速度Uと相対速度Urelとを比較する。計算領域全体がSRF領域であるため，中心のかくはん翼は静止して見えるが，実際には回転している。逆に，静止座標系では1方向に流れる流れが，SRFFreestreamVelocityの効果で，方向を変えながら流れるように見える。

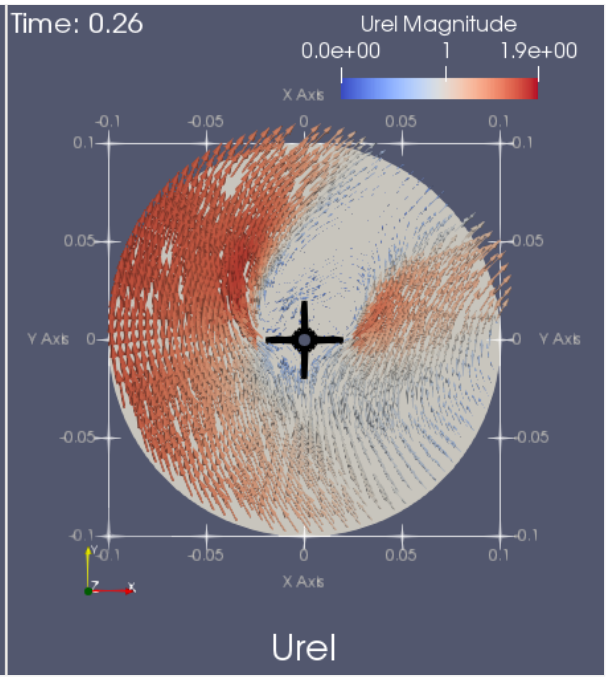
時刻0では，UInfを(1 0 0)としているため，周囲の流れはx軸正方向に1m/sである。

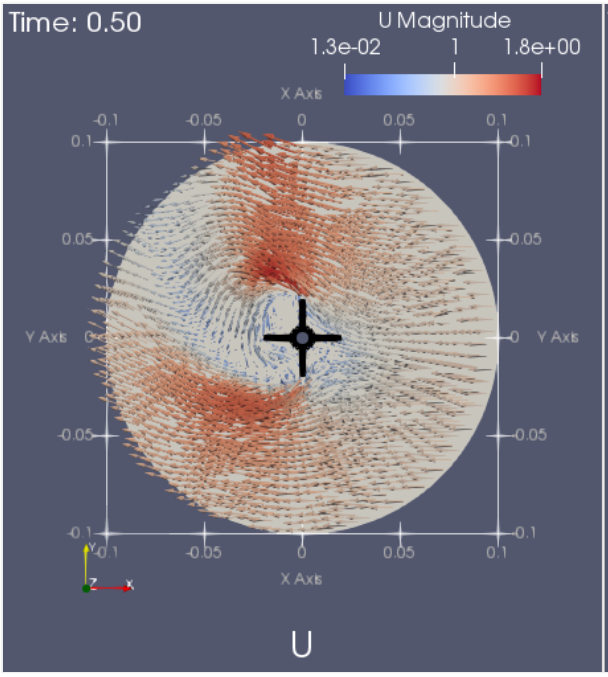
回転数を60rpmとしていることから，1秒で1回転する。約0.25秒では，回転軸が1/4回転し，主流の向きが1/4回転分変化している。

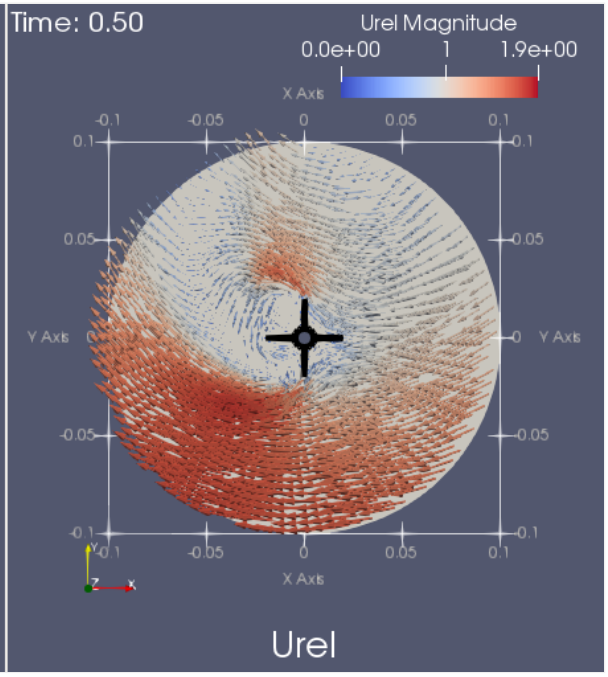

Time: 0.02
U Magnitude
1.3e-02 1 1.8e+00
X Axis
Y Axis
U

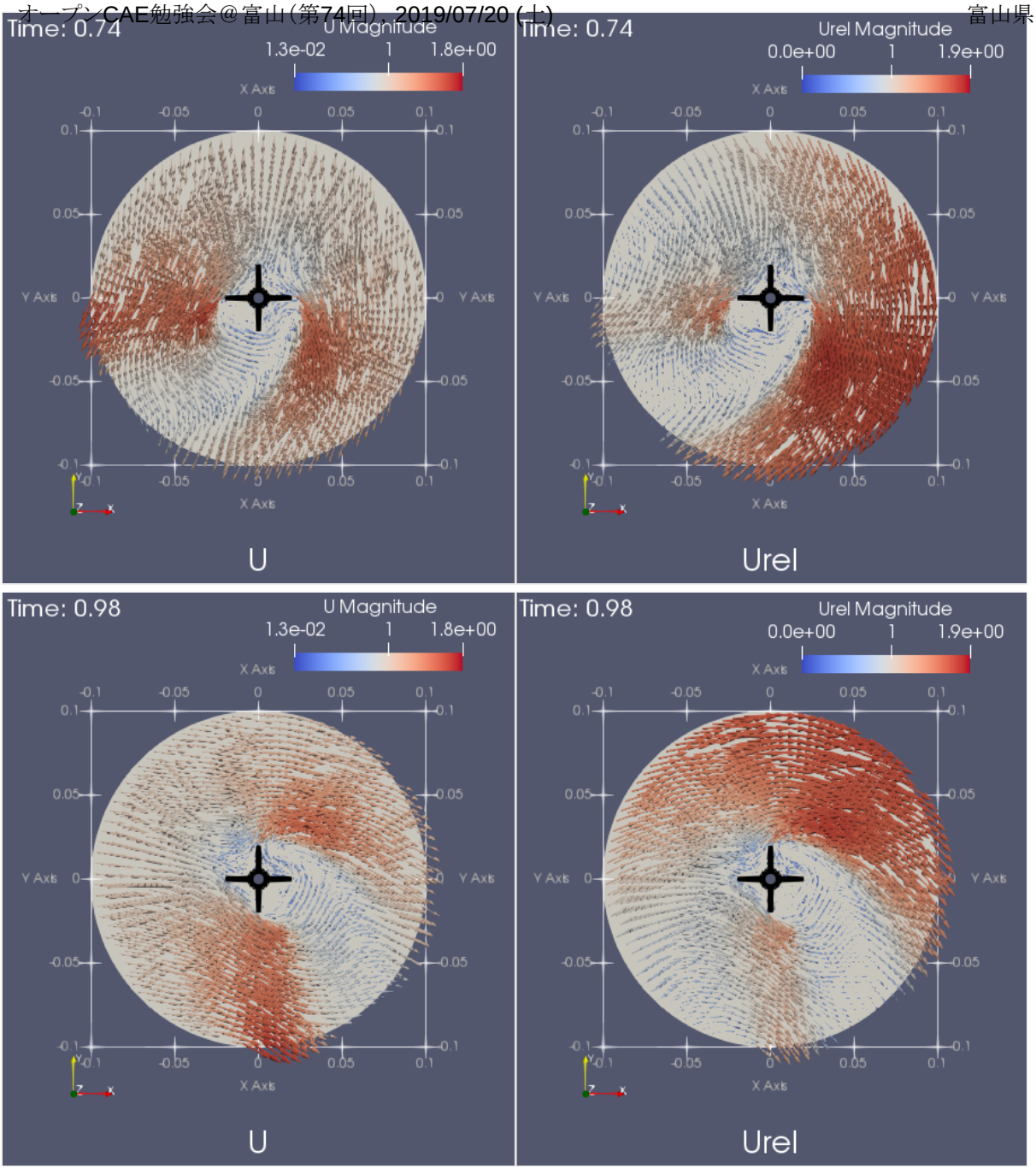

Time: 0.02
Urel Magnitude
0.0e+00 1 1.9e+00
X Axis
Y Axis
Urel


Time: 0.26
U Magnitude
1.3e-02 1 1.8e+00
X Axis
Y Axis
U


Time: 0.26
Urel Magnitude
0.0e+00 1 1.9e+00
X Axis
Y Axis
Urel

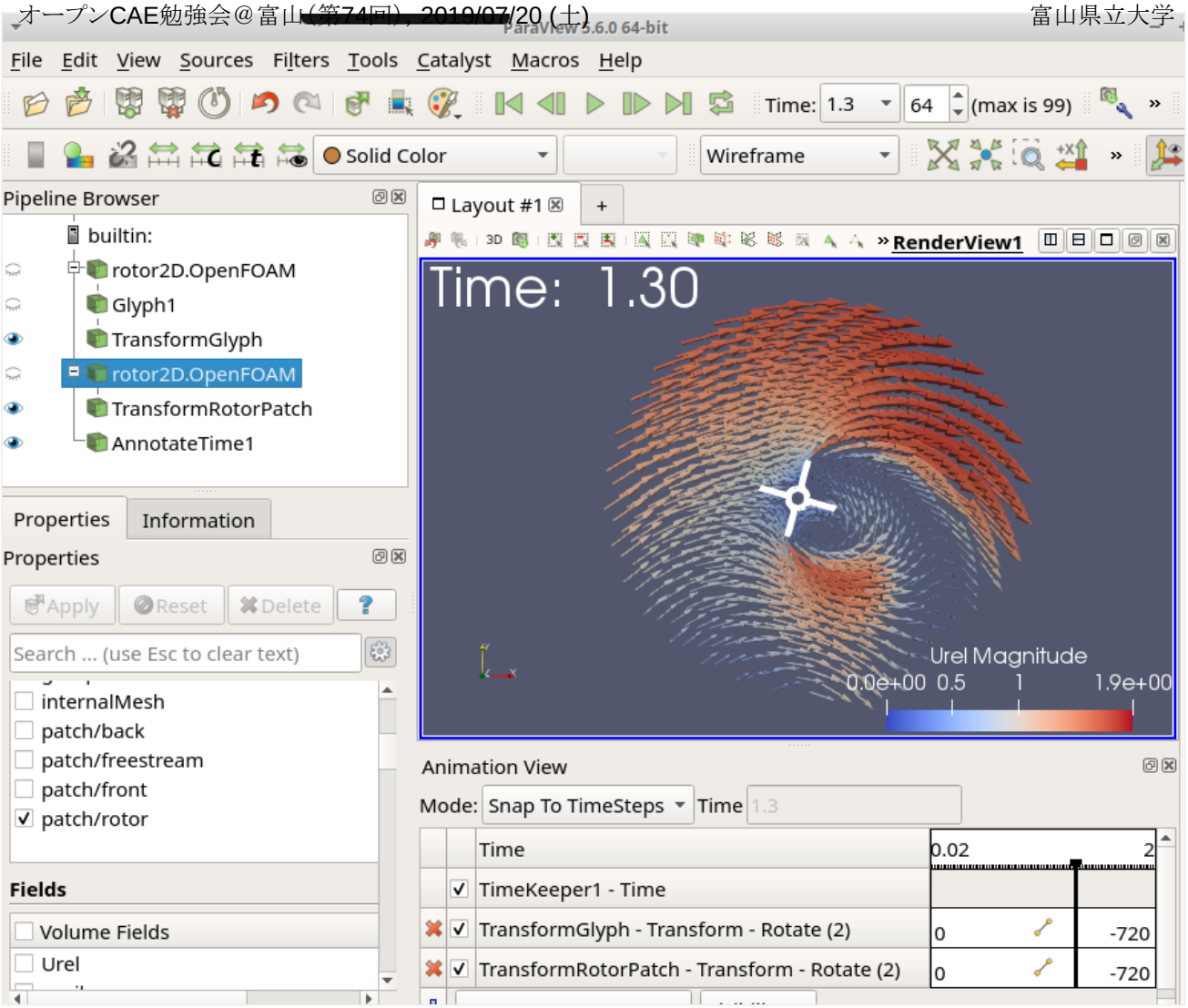

Time: 0.50
U Magnitude
1.3e-02 1 1.8e+00
X Axis
Y Axis
U


Time: 0.50
Urel Magnitude
0.0e+00 1 1.9e+00
X Axis
Y Axis
Urel



実際の流れを想像するためには，paraview の Transform 機能と，animation 機能を利用するとよい。

Animation View を開く。[Transform1] [Transform - Rotate (2)]をプラスする。これで，z軸周りの回転を含めた動画が作成できる。ただし，回転系で計算を実施しているため，この動画は正しくないことになる。



コードでみる違い

主として，次の場所に存在するソースコードを確認する。（ `$WM_PROJECT_DIR` の中にある。）

applications/solvers/incompressible/simpleFoam/

applications/solvers/incompressible/simpleFoam/SRFSimpleFoam/

src/finiteVolume/cfdTools/general/MRF/

src/finiteVolume/cfdTools/general/SRF/

simpleFoam と SRFSimpleFoam の基礎式

simpleFoam の Ueqn.H 抜粋

applications/solvers/incompressible/simpleFoam/UEqn.H

Ueqn.H

```
tmp<fvVectorMatrix> tUEqn
(
    fvm::div(phi, U)
  + MRF.DDt(U)
  + turbulence->divDevReff(U)
 ==
    fvOptions(U)
);
```

SRFSimpleFoam の UrelEqn.H 抜粋

applications/solvers/incompressible/simpleFoam/SRFSimpleFoam/UrelEqn.H

Ueqn.H

```
// Relative momentum predictor
tmp<fvVectorMatrix> tUrelEqn
(
    fvm::div(phi, Urel)
  + turbulence->divDevReff(Urel)
  + SRF->Su()
 ==
    fvOptions(Urel)
);
```

関連コード

MRF

src/finiteVolume/cfdTools/general/MRF/MRFZoneList.H

```
        //- Return the frame acceleration
        tmp<volVectorField> DDt
        (
            const volVectorField& U
        ) const;
```

src/finiteVolume/cfdTools/general/MRF/MRFZoneList.C

```
Foam::tmp<Foam::volVectorField> Foam::MRFZoneList::DDt
(
    const volVectorField& U
) const
{
    tmp<volVectorField> tacceleration
    (
        volVectorField::New
        (
            "MRFZoneList:acceleration",
            U.mesh(),
            dimensionedVector(U.dimensions()/dimTime, Zero)
        )
    );
    volVectorField& acceleration = tacceleration.ref();

    forAll(*this, i)
    {
        operator[](i).addCoriolis(U, acceleration);
    }
    return tacceleration;
}
```

src/finiteVolume/cfdTools/general/MRF/MRFZone.C

```
void Foam::MRFZone::addCoriolis
(
    const volVectorField& U,
    volVectorField& ddtU
) const
{
    if (cellZoneID_ == -1)
    {
        return;
    }

    const labelList& cells = mesh_.cellZones()[cellZoneID_];
    vectorField& ddtUc = ddtU.primitiveFieldRef();
    const vectorField& Uc = U;

    const vector Omega = this->Omega();

    forAll(cells, i)
    {
        label celli = cells[i];
        ddtUc[celli] += (Omega ^ Uc[celli]);
    }
}
```

SRF

solvers/incompressible/simpleFoam/SRFSimpleFoam/SRFSimpleFoam.C

```
U = Urel + SRF->U();
```

applications/solvers/incompressible/simpleFoam/SRFSimpleFoam/UrelEqn.H

```
    tmp<fvVectorMatrix> tUrelEqn
    (
        fvm::div(phi, Urel)
      + turbulence->divDevReff(Urel)
      + SRF->Su()
     ==
        fvOptions(Urel)
    );
```

src/finiteVolume/cfdTools/general/SRF/SRFModel/SRFModel/SRFModel.H

```
// Reference to the relative velocity field
     Urel_;
//- Return velocity of SRF for complete mesh
    tmp<volVectorField> U() const;
```

src/finiteVolume/cfdTools/general/SRF/SRFModel/SRFModel/SRFModel.C

```
Foam::tmp<Foam::DimensionedField<Foam::vector, Foam::volMesh>>
Foam::SRF::SRFModel::Fcoriolis() const
{
    return volVectorField::Internal::New
    (
        "Fcoriolis",
        2.0*omega_ ^ Urel_
    );
}


Foam::tmp<Foam::DimensionedField<Foam::vector, Foam::volMesh>>
Foam::SRF::SRFModel::Fcentrifugal() const
{
    return volVectorField::Internal::New
    (
        "Fcentrifugal",
        omega_ ^ (omega_ ^ (mesh_.C() - origin_))
    );
}
```

```
Foam::tmp<Foam::DimensionedField<Foam::vector, Foam::volMesh>>
Foam::SRF::SRFModel::Su() const
{
    return Fcoriolis() + Fcentrifugal();
}


Foam::vectorField Foam::SRF::SRFModel::velocity
(
    const vectorField& positions
) const
{
    tmp<vectorField> tfld =
        omega_.value()
      ^ (
            (positions - origin_.value())
          - axis_*(axis_ & (positions - origin_.value()))
        );
    return tfld();
}


Foam::tmp<Foam::volVectorField> Foam::SRF::SRFModel::U() const
{
    return volVectorField::New
    (
        "Usrf",
        omega_ ^ ((mesh_.C() - origin_) - axis_*(axis_ & (mesh_.C() - origin_)))
    );
}
```

src/finiteVolume/cfdTools/general/SRF/derivedFvPatchFields

```
Class
    Foam::SRFVelocityFvPatchVectorField
Group
    grpInletBoundaryConditions grpWallBoundaryConditions
Description
    Velocity condition to be used in conjunction with the single
    rotating frame (SRF) model (see: SRFModel class)

    Given the free stream velocity in the absolute frame, the condition
    applies the appropriate rotation transformation in time and space to
    determine the local velocity.

    The optional \c relative flag switches the behaviour of the patch
    such that:

        - relative = yes: inlet velocity applied 'as is':

        \f[
            U_p = U_{in}
        \f]

        - relative = no : SRF velocity is subtracted from the inlet velocity:

        \f[
            U_p = U_{in} - U_{p,srf}
        \f]

    where
    \vartable
        U_p     | patch velocity [m/s]
        U_{in}  | user-specified inlet velocity
        U_{p,srf} | SRF velocity
    \endvartable

Usage
    \table
        Property     | Description             | Required    | Default value
        inletValue   | inlet velocity          | yes         |
        relative     | inletValue relative motion to the SRF? | yes     |
    \endtable

    Example of the boundary condition specification:
    \verbatim
    <patchName>
    {
        type            SRFVelocity;
        inletValue      uniform (0 0 0);
        relative        yes;
        value           uniform (0 0 0);    // initial value
    }
    \endverbatim
```

```
See also
    Foam::fixedValueFvPatchField
SourceFiles
    SRFVelocityFvPatchVectorField.C


Class
    Foam::SRFWallVelocityFvPatchVectorField
Group
    grpWallBoundaryConditions
Description
    Wall-velocity condition to be used in conjunction with the single rotating
    frame (SRF) model (see: FOAM::SRFModel)

    The condition applies the appropriate rotation transformation in time and
    space to determine the local SRF velocity of the wall.

        \f[
            U_p = - U_{p,srf}
        \f]

    where
    \vartable
        U_p     = patch velocity [m/s]
        U_{p,srf} = SRF velocity
    \endvartable

    The normal component of \f$ U_p \f$ is removed to ensure 0 wall-flux even
    if the wall patch faces are irregular.

Usage
    Example of the boundary condition specification:
    \verbatim
    <patchName>
    {
        type            SRFWallVelocity;
        value           uniform (0 0 0);    // Initial value
    }
    \endverbatim

See also
    Foam::SRFModel
    Foam::SRFVelocityFvPatchVectorField
    Foam::fixedValueFvPatchField
SourceFiles
    SRFWallVelocityFvPatchVectorField.C
```

```
Class
    Foam::SRFFreestreamVelocityFvPatchVectorField
Description
    Freestream velocity condition to be used in conjunction with the single
    rotating frame (SRF) model (see: SRFModel class)

    Given the free stream velocity in the absolute frame, the condition
    applies the appropriate rotation transformation in time and space to
    determine the local velocity using:

    \f[
        U_p = cos(\theta) U_{Inf} + sin(\theta) (n \times U_{Inf}) - U_{p,srf}
    \f]

    where
    \vartable
        U_p     | patch velocity [m/s]
        U_{Inf} | free stream velocity in the absolute frame [m/s]
        \theta  | swept angle [rad]
        n       | axis direction of the SRF
        U_{p,srf} | SRF velocity of the patch [m/s]
    \endvartable

Usage
    \table
        Property        | Description               | Required | Default value
        UInf            | freestream velocity       | yes      |
        relative        | UInf relative to the SRF? | no       |
    \endtable

    Example of the boundary condition specification:
    \verbatim
    <patchName>
    {
        type            SRFFreestreamVelocity;
        UInf            uniform (0 0 0);
        relative        no;
        value           uniform (0 0 0);    // initial value
    }
    \endverbatim

See also
    Foam::freestreamFvPatchField
    Foam::SRFVelocityFvPatchVectorField
SourceFiles
    SRFFreestreamVelocityFvPatchVectorField.C
```

1. http://www.gnu.org/software/m4/m4.html ⮌

Comments 0