OpenFOAM Project: WhipLashMotion Library

Junfeng Yang

2009

Introduction

Whiplash-like motion causes lot of human neck injury during the vehicle crash accident. To investigate the transient blood flow inside the human vertebra during the whiplash-like motion. The OpenFOAM is a powerful CFD simulation tool to do such research. In this article, all the CFD simulation are implemented in OpenFOAM-1.5dev version.

To mimic the whiplash motion, the icoDyMFoam application was adapted for calculating dynamic mesh moments. With different needs for the users, several different ways of ,moving mesh have been developed. In this work, a new tool has been implemented, with the purpose of whiplash-like motion. The mesh movement is done with topology change and mesh deformation. This means that column cells extend in vertical direction depending on the movement. Currently, only the mesh motion has been implemented. The material of the moving object was regarded as solid wall. The focus is thus solely the movement and topology changes of the mesh. Further on, the soft or flexible wall (we can use compressible fluid) will be implemented. the interaction between wall and liquid domain will be investigated.

Background

As starting point the movingConeTopo tutorial located in the icoDymFoam folder of OpenFOAM-1.5-dev was used. This is a working tutorial that can be run by simply copying the movingConetopo folder to your run directory , and run it according to :

1: cp –r $FOAM_TUTORIAL/icoDyMFoam/movingConeTopo/ $FOAM_RUN

2: cd $FOAM_RUN/movingConeTopo

3: blockMesh . .

4: icoDyMFoam . .

The case will run. The paraFoam could check the mesh motion.

For the aim of the present work, a new case setup was named WhiplashMotion. The mesh will move according the experimental motion data . The new mesh can be seen in Figure 1.

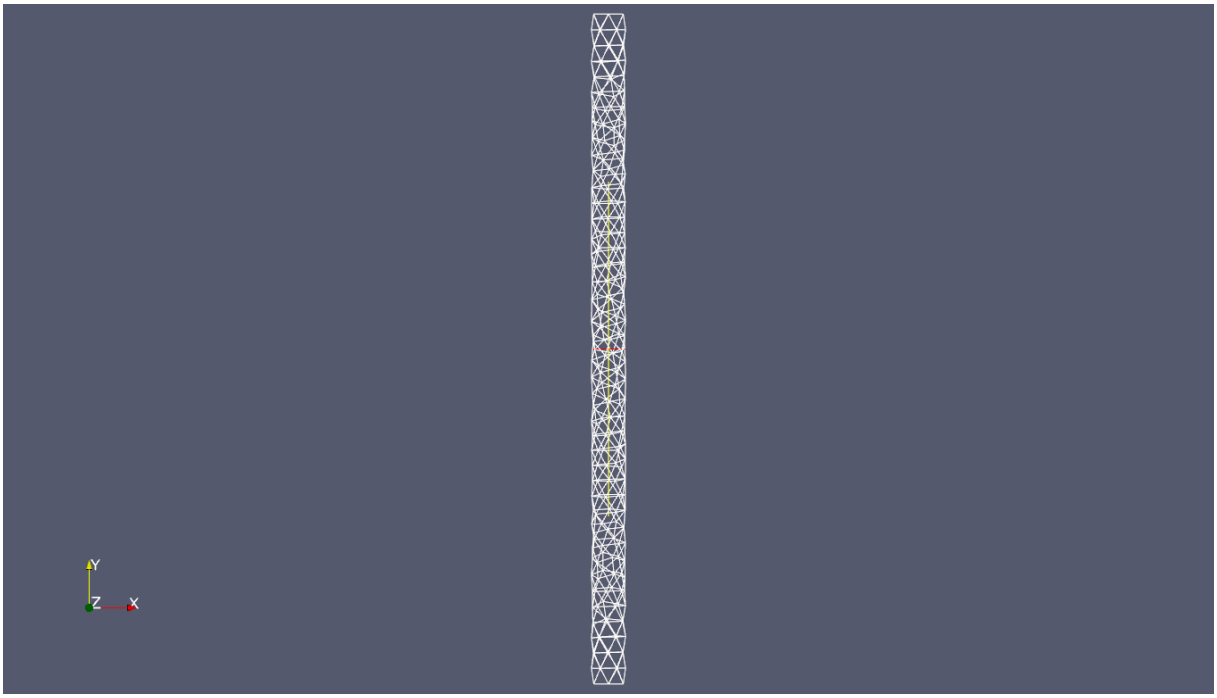


Figure 1 Simple pipe mesh to represent blood vessels inside the vertebral

The new mesh motion implementation was generated by MatLab code for the new case, so a dynamicWhiplashfvMesh library was created based on the movingconeTopoFvMesh library:

```
1: cp –r $FOAM_SRC/dynamicFvMesh/dynamicFvMesh \
    $WM_PROJECT_USER_DIR/dynamicWhiplashFvMesh
```

The library was copied and the names were changed to the dynamicWhiplashFvMesh. Copied were also the Make folder containing the files : files and options

```
1: cp –r $FOAM_SRC/topoChangerFvMesh/Make/ \
   $WM_PROJECT_USER_DIR/dynamicWhiplashFvMesh
```

The files were rewritten into the following line order to only compile the dynamicWhiplashFvMeshlibrary.

```
1: dynamicWhiplashFvMesh.C
2: LIB = $(FOAM_USER_LIBBIN)/ libmyDynamicFvMesh
```

In the options file, it should contain following line:

```
1: -I$(LIB_SRC)/dynamicFvMesh/lnInclude \
```

If it does not, please add the above line into the options file like :

```
1: EXE_INC = \
2:    -I$(LIB_SRC)/dynamicFvMesh/lnInclude
```

The copied dynamicFvmesh can now be adopted for the new functionality. To compile the changes made in the library, the command:

```
1: wmake libso
```

Was used. The complete dynamicWhiplashFvMesh.C can be found in the appendix. The library were called libmyDynamicFvMesh. In order to used the new library for a case, the constant/dynamicFvMeshDict dictionaryshould sprcify the name of the new library. This case is based on the movingConeTopo tutorial. The dynamicFvMeshDict is the dictionary controlling the movement of the mesh. In this one can alter the velocity, amplitude and period of the object. In this one also have to describe at which position the moving cells are located. The dynamicWhiplashFvMesh file can be seen in the appendix.

Methodology

In order to make this work the dynamicWhiplashFvMesh.C and dynamicWhiplashFvMesh.H had to be changed in accordance with the changes of the mesh. The most interesting part however is the changes made for the movement. Initially the different regions where marked up as can be seen in figure 2.

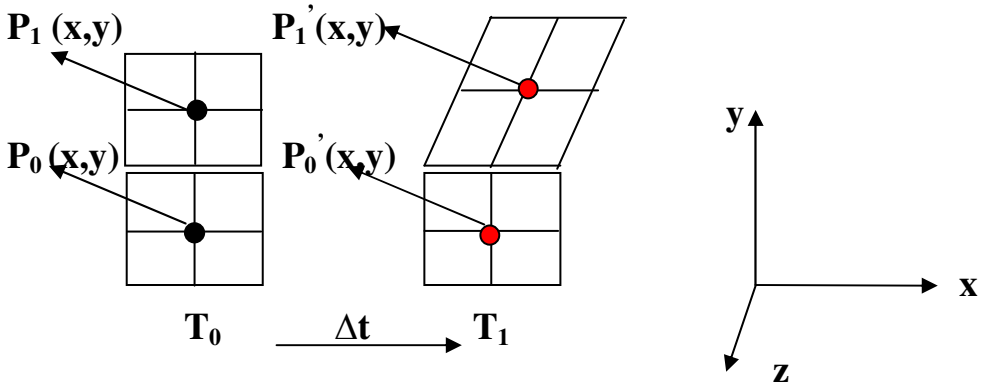


Figure 2 Cells motion explanation

In the Figure 2, the block 0 is fixed, the block 1 can move and change the shape. P1 and P0 are the central point of the block 1 and 0. Assuming after Δt time step (from T0 to T1), a vertical pipe moves as shown in figure. The central point of P0 and P1 become to P0' and P1'. The position of P0, P1 varied at both x-axis and y-axis direction (assume there is no movement in z-axis). The movement in x-axis can be described by a 5 degree polynomial equations: $x(i)=a*y(i)^5+b*y(i)^4+c*y(i)^3+d*y(i)^2+e*y(i)+f$

Where a,b,c,d,e,f are the know coefficients. x(i), y(i) are the position of central point of block in the Cartesian coordinates. Such coeeficients and y-axis position have been recalculated using Matlab code.

Take block 1 as example, in Δt time step, the total movement in x-axis of central point is equal to Δx=x-x(i), x(T1) is the current x position, x(T0) is the x position in last time step. For each block, all the cells are regarded to move the same distance in x-direction. By this way, all the cell move to a new position that was specified based on the known data at each time step. Finally, such movement will consist in a whiplash motion.

Mesh Generation and B.C

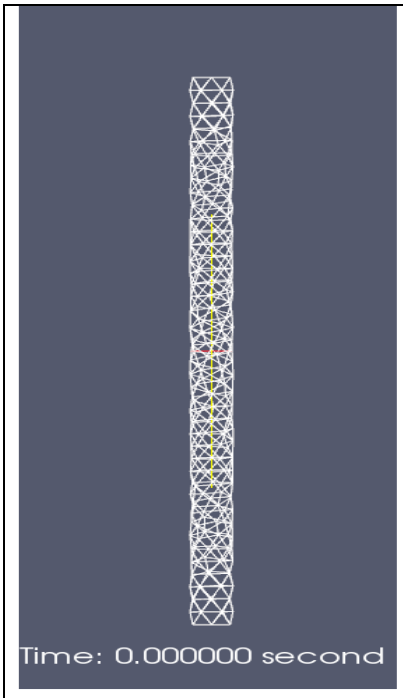
The pipe mesh was generated by the GAMBIT module. Then fluent3DMeshToFoam will convert it to FOAM mesh. checkMesh will show the mesh information.

Initial values and Boundary Conditions

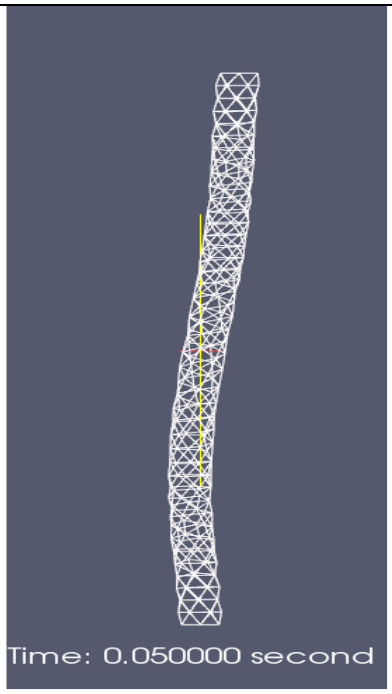
In the '0' folder, p and U files, all the initial value for pressure and velocity are defined, see appendix.

In the constant/polymesh/boundary file, the B.C is defined, see appendix.

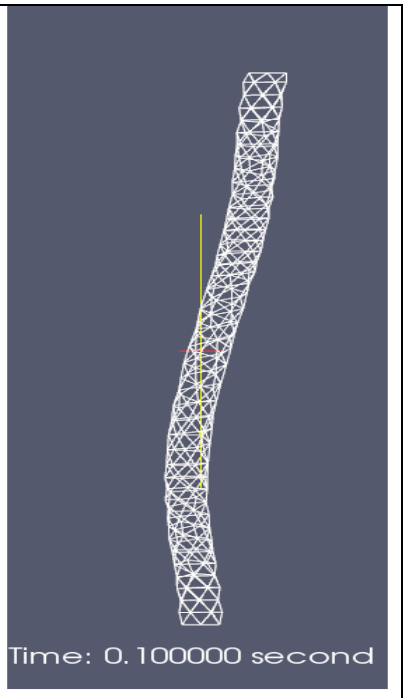
Results and Future Work



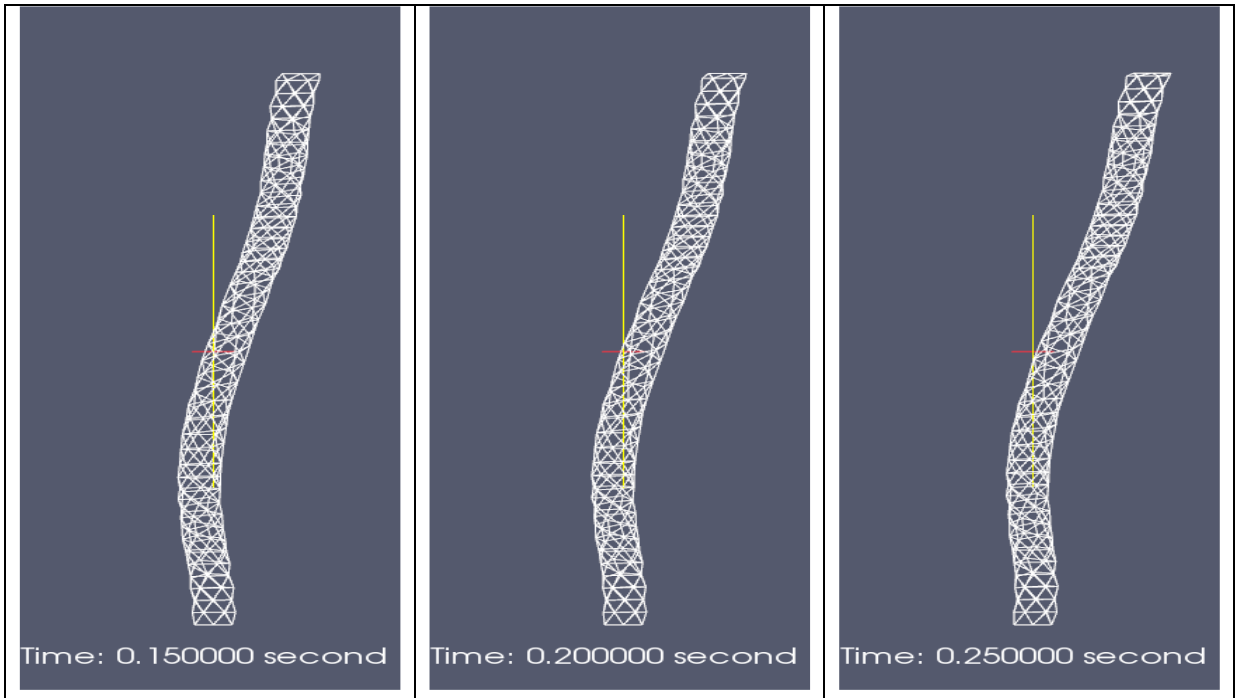
a)



b)



c)



d) **e)** **f)**

The results can be seen in Figure 3. The mesh moving according to the experimental motion data to perform the whiplash-like motion

It show the dynamicWhiplashfvMesh application can do the same whiplash-like motion simulation as commercial code: Fluent.

But, there are still a lot of work need to be done:

1. **To specify the proper B.C (like liquid property and wall property) for the whiplash motion and perform the simulation to calculate the pressure and velocity transient inside fluid domain.**
2. **To modeling the compressible flow or flexible wall (weak-contrain pipe)**
3. **Adaptive grid technology could be interesting aspect to develop to avoid the zero or negative volume cell that normally occur during the compress force applied on the cells**

Appendix A

The dynamicWhiplashFvMesh.C

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) Original Authors
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software; you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by the
    Free Software Foundation; either version 2 of the License, or (at your
    option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but
WITHOUT
    ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM; if not, write to the Free Software Foundation,
    Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

\*---------------------------------------------------------------------------*/

#include "dynamicWhiplashFvMesh.H"
#include "addToRunTimeSelectionTable.H"
#include "volFields.H"
#include "mathematicalConstants.H"

// * * * * * * * * * * * * * * * Static Data Members * * * * * * * * * * * * * * * //

namespace Foam
```

```
{
    defineTypeNameAndDebug(dynamicWhiplashFvMesh, 0);
    addToRunTimeSelectionTable(dynamicFvMesh,
dynamicWhiplashFvMesh, IOobject);
}

// * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * //

Foam::dynamicWhiplashFvMesh::dynamicWhiplashFvMesh(const
IOobject& io)
:
    dynamicFvMesh(io),
    dynamicMeshCoeffs_
    (
        IOdictionary
        (
            IOobject
            (
                "dynamicMeshDict",
                io.time().constant(),
                *this,
                IOobject::MUST_READ,
                IOobject::NO_WRITE
            )
        ).subDict(typeName + "Coeffs")
    ),
    newPoints_
    (
        IOobject
        (
            "points",
            io.time().constant(),
            meshSubDir,
            *this,
            IOobject::MUST_READ,
            IOobject::NO_WRITE
        )
    )
{
    Info<< "Performing a whiplash dynamic mesh calculation" << endl;

    FILE *file_1;
    FILE *file_2;
```

```
    //file_1=fopen("constant/vessel_motion_renew.mat", "r");
    //file_2=fopen("constant/vessel_node_y_renew.mat", "r");
    file_1=fopen("constant/vessel_motion.mat", "r");
    file_2=fopen("constant/vessel_node_y.mat", "r");

    Info<< "\nReading constant/vessel_motion_renew.mat:" << endl;
    for(int m=0;m<3000;++m){
      for(int v=0;v<6;++v){
        fscanf(file_1, "%f",&P_[m][v]);
        Info<< P_[m][v] << " ";
      }
      Info<<endl;
    }

    fclose(file_1);

    Info<< "\nReading constant/vessel_node_y_renew.mat:" << endl;
    for(int m=0;m<3000;++m){
      for(int v=0;v<7;++v){
        fscanf(file_2, "%f",&Y_[m][v]);
        Info<< Y_[m][v] << " ";
      }
      Info<<endl;
    }
    Info<<endl;

    fclose(file_2);

}

// * * * * * * * * * * * * * * * * Destructor  * * * * * * * * * * * * * * * //
Foam::dynamicWhiplashFvMesh::~dynamicWhiplashFvMesh()
{}

// * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * //

bool Foam::dynamicWhiplashFvMesh::update()
{

    float x, y, z;
    float delta_1=0.0, delta_2=0.0, delta_x=0.0;
    float J=0.0, K=0.0;
```

```
   float up_nod0,dw_nod0,up_nod1,dw_nod1; //0=last timestep; 1=current
timestep
   int update_y,section_id;
   //int N=time().value()/time().deltaT().value();
   int N=1+time().value()/0.001;

   Info<< "Integer time:" << N << endl;

   forAll(newPoints_, nIter)
   {
      update_y = 0;
      section_id = -1;
      z = newPoints_[nIter].component(vector::Z);
      y = newPoints_[nIter].component(vector::Y);
      x = newPoints_[nIter].component(vector::X);

      if (y<0 && update_y==0) //deal with y<0 section
      {
         //Info<< "Section_id = 0" <<endl;
         update_y = 1;
         section_id = 0;
         J = 0;
         delta_x = x - (P_[N-1][0] * pow(y,5)+
                   P_[N-1][1] * pow(y,4)+
                   P_[N-1][2] * pow(y,3)+
                   P_[N-1][3] * pow(y,2)+
                   P_[N-1][4] * pow(y,1)+
                   P_[N-1][5]);
      }
      for(int i=1;i<=6;++i)
         //N is time step, i is node index and section index
      {
         up_nod0 = Y_[N-1][i];   //last time step
         dw_nod0 = Y_[N-1][i-1];
         up_nod1 = Y_[N][i];     //current time step
         dw_nod1 = Y_[N][i-1];

         if(y>=dw_nod0 && y<=up_nod0 && update_y==0) //IN section 1~6
         {
            //Info<< "Section_id = " << i <<endl;
            update_y = 1;
            section_id = i;
            delta_1 = up_nod0-dw_nod0;
```

```
        delta_2 = up_nod1-dw_nod1;
        delta_x = x - (P_[N-1][0] * pow(y,5)+
                    P_[N-1][1] * pow(y,4)+
                    P_[N-1][2] * pow(y,3)+
                    P_[N-1][3] * pow(y,2)+
                    P_[N-1][4] * pow(y,1)+
                    P_[N-1][5]);
        J = (y-dw_nod0)*delta_2/delta_1+dw_nod1;
        break;
    }
}
if(update_y==0) //IN higher then section 6 part
{
    //Info<< "Section_id = 7" <<endl;
    update_y = 1;
    section_id = 7;
    J = Y_[N][6];
    delta_x = x - (P_[N-1][0] * pow(y,5)+
                P_[N-1][1] * pow(y,4)+
                P_[N-1][2] * pow(y,3)+
                P_[N-1][3] * pow(y,2)+
                P_[N-1][4] * pow(y,1)+
                P_[N-1][5]);
}
if(update_y)
{
    //Calculate X  value from motion equation:
    K = P_[N][0] * pow(J,5)+
        P_[N][1] * pow(J,4)+
        P_[N][2] * pow(J,3)+
        P_[N][3] * pow(J,2)+
        P_[N][4] * pow(J,1)+
        P_[N][5];
    newPoints_[nIter].component(vector::Z) = z;
    newPoints_[nIter].component(vector::Y) = J;
    newPoints_[nIter].component(vector::X) = delta_x+K;
}
else{
  Info<< "Found an un-updated point: " << x << y <<endl;
  //Add fatalError break!
  //Message("find a un-updated point: (%g,%g)",x,y);
  //Error("un-updated point error!");
}
```

```
    }

    fvMesh::movePoints(newPoints_);

    volVectorField& U =
        const_cast<volVectorField&>(lookupObject<volVectorField>("U"));
    U.correctBoundaryConditions();

    return true;
}
```

Appendix B

The dynamicWhiplashFvMesh.H

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) Original Authors
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software; you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by the
    Free Software Foundation; either version 2 of the License, or (at your
    option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but
WITHOUT
    ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM; if not, write to the Free Software Foundation,
    Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Class
    Foam::dynamicWhiplashFvMesh

Description
    Mesh motion specifically for the "pumping" system of an ink-jet
    injector.

    The set of points in the "pumping" region are compressed and expanded
    sinusoidally to impose a sinusoidal variation of the flow at the
    nozzle exit.
```

```
SourceFiles
   dynamicWhiplashFvMesh.C

Authors
   Håkan Nilsson
   OpenCFD

\*---------------------------------------------------------------------------*/

#ifndef dynamicWhiplashFvMesh_H
#define dynamicWhiplashFvMesh_H

#include "dynamicFvMesh.H"
#include "dictionary.H"
#include "pointIOField.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{

/*---------------------------------------------------------------------------*\
                    Class dynamicWhiplashFvMesh Declaration
\*---------------------------------------------------------------------------*/

class dynamicWhiplashFvMesh
:
   public dynamicFvMesh
{
   // Private data

      dictionary dynamicMeshCoeffs_;

      float P_[3000][6], Y_[3000][7];          /*P: coefficient Y: coordinate*/

      pointIOField newPoints_;

   // Private Member Functions

      //- Disallow default bitwise copy construct
      dynamicWhiplashFvMesh(const dynamicWhiplashFvMesh&);
```

```
        //- Disallow default bitwise assignment
        void operator=(const dynamicWhiplashFvMesh&);


public:

    //- Runtime type information
    TypeName("dynamicWhiplashFvMesh");


    // Constructors

        //- Construct from IOobject
        dynamicWhiplashFvMesh(const IOobject& io);


    // Destructor

        ~dynamicWhiplashFvMesh();


    // Member Functions

        //- Update the mesh for both mesh motion and topology change
        virtual bool update();
};


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

} // End namespace Foam

#endif
```

Appendix C

0 folder p file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.5                                   |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    wall
    {
        type            zeroGradient;
    }


    bottom
    {
        type            totalPressure;
        p0              uniform 0;
        U               U;
        phi             phi;
        rho             none;
        psi             none;
        gamma           1;
        value           uniform 0;
```

```
    }

    top
    {
        type            totalPressure;
        p0              uniform 0;
        U               U;
        phi             phi;
        rho             none;
        psi             none;
        gamma           1;
        value           uniform 0;
    }

}

// ************************************************************************* //
```

Appendix D

0 folder U file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.5                                   |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    wall
    {
        //type          fixedValue;
        type          movingWallVelocity;
        value         uniform (0 0 0);
    }

    bottom
    {
        type          pressureInletOutletVelocity;
        value         uniform (0 0 0);
    }

    top
    {
        type          pressureInletOutletVelocity;
```

```
        value           uniform (0 0 0);
    }

}

//
********************************************************************************
********* //
```

Appendix E

Polymesh /boundary

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       polyBoundaryMesh;
    location    "constant/polyMesh";
    object      boundary;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

3
(
    wall
    {
        type            wall;
        nFaces          10056;
        startFace       113038;
    }
    bottom-vent
    {
        type            patch;
        nFaces          48;
        startFace       127923;
    }
    top-vent
    {
        type            patch;
        nFaces          48;
        startFace       127971;
    }
)
```