OpenFOAM project: A modification of the movingConeTopoFvMesh library

Erik Bjerklund

2009

Introduction

Moving meshes are becoming more and more popular in CFD. The icoDyMFoam application in OpenFOAM can use different tools for calculating dynamic mesh movements. With different needs for the users, several different ways of moving meshes have been developed. In this work, a new tool has been implemented, with the purpose of having two objects meeting each other. The mesh movement is done with topology change and mesh deformation. This means that rows of cells are added and removed depending on the movement. The cas has been set up for the development of the new functionality consists of two domains with a sliding interface between the domains. In each domain, an object is moving parallell to the sliding interface. Currently, only the mesh motion has been implemented, and there is no coupling between the two domains across the sliding interface. The focus is thus solely on the movement and topology changes of the mesh.

Background

As starting point the $movingConeTopo$ tutorial located in the $icoDyMFoam$ folder of OpenFOAM-1.4.1-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

```
cp -r $FOAM_TUTUORIALS/icoDyMFoam/movingConeTopo/ $FOAM_RUN
cd $FOAM_RUN/movingConeTopo
blockMesh . .
icoDymFoam . .
```

Postprocessing in paraFoam will show a wedge mesh with a cone moving back and forth with adding and removing cells behind and in front of the moving object as can be seen in Figure 1.

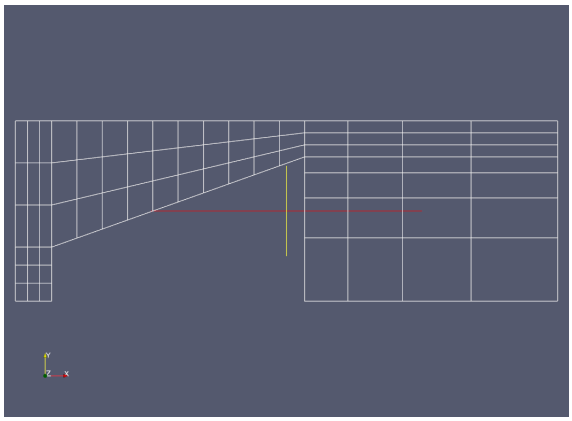

Figure 1: Initial mesh for the movingConeTopo tutorial

For the aim of the present work, a new case setup was named slidingConesTopo. The mesh was then altered to consist of two objects and made larger to allow more movement. The new mesh can be seen in figure 2

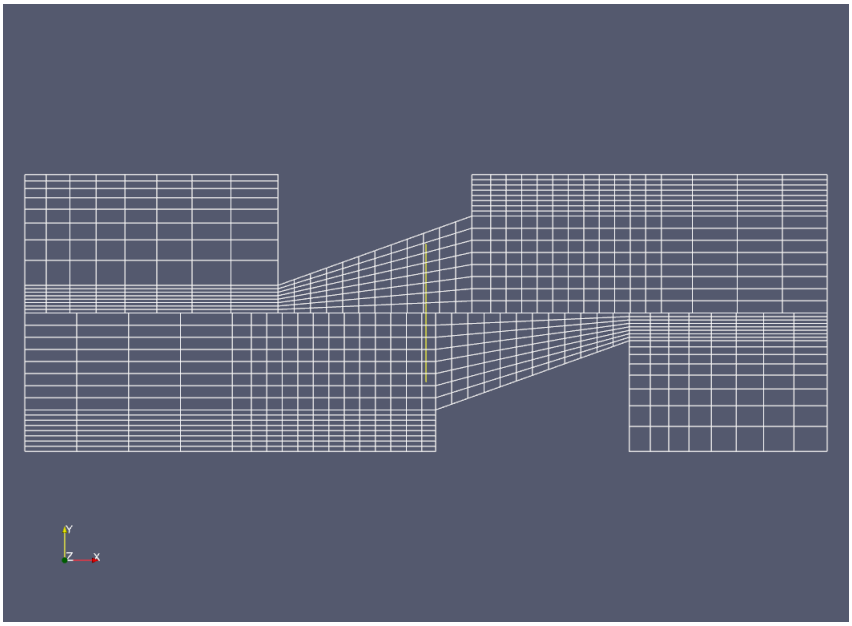

Figure 2: Initial mesh for the slidingConesTopo

The original mesh motion implementaion was insufficient for the new case, so a slidingConesfvMesh library was based on the movingConeTopoFvMesh library:

```
cp -r $FOAM_SRC/topoChangerFvMesh/movingConeTopoFvMesh \
$WM_PROJECT_USER_DIR/slidingConesFvMesh
cd $WM_PROJECT_USER_DIR/slidingConesFvMesh
sed s/movingConeTopoFvMesh/slidingConesTopo/ \
<movingConeTopoFvMesh.C >slidingConesFvMesh.C
sed s/movingConeTopoFvMesh/slidingConesTopo/ \
<movingConeTopoFvMesh.H >slidingConesFvMesh.H
```

The library was copied and the names were changed to $slidingConesFvMesh.*$. Copied were also the $Make$ folder containg the files $files$ and $options$.

```
cp -r $FOAM_SRC/topoChangerFvMesh/Make/ \
$WM_PROJECT_USER_DIR/slidingConesFvMesh/
```

The $files$ were rewritten into the following lines in order to only compile the slidingConesFvMesh library.

```
slidingConesFvMesh.C

LIB = $(FOAM_USER_LIBBIN)/libslidingConesFvMesh
```

In the $options$ file a line for including the files from the original library was added.

```
1     -I$(LIB_SRC)/topoChangerFvMesh/lnInclude \
```

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

```
1 cd $WM_PROJECT_USER_DIR/slidingConesFvMesh
2 wmake libso
```

was used. The complete $slidingConesFvMesh.C$ can be found in the appendix. The library were called libslidingConesFvMesh. In order to use the new library for a case,
the $constant/dynamicFvMeshDict$ dictionary should specify the name of the new library. This case is based on the movingConeTopo tutotial. 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 positions the edges of the obstacles that is going to be moved are located. Added to this file are the leftUpperObstacleEdge and rightUpperObstacleEdge as well as how thin or thick the added cells are allowed to be for the upper part of the mesh. The $dynamicFvMesh$ file can be seen in the appendix.

Method

In order to make this work the $slidingConesFvMesh.C$ and $slidingConesFvMesh.H$ had to be changed in accordance with the changes of the mesh. To start with the orignal implementation only knew of the two faces on the lower part of the mesh. Everything the software did was thus only applied to that part. So wherever there was $left$ and $right$ in the library, $leftUpper$ and $rightUpper$ was added, making the code almost twice as long. So all zone adding and mesh modifying is now independent of region. I can be altered so that the minimum and maximum thickness of the cells added and removed differs depending on where in the mesh it is located. These changes is easily located in the $slidingConesFvMesh.C$ code in the appendix. 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 3.

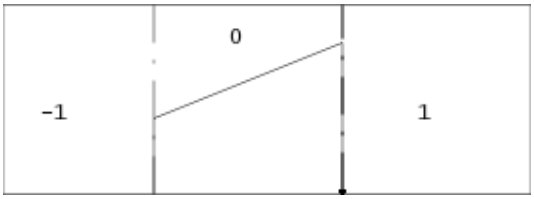

Figure 3: The mesh marked before the changes for the masking

This was then changed in the $vertexMarkup$ member function. The different regions with 0 , -1 and 1 in the first picture is not taken advantage of in the library. This is explained later with the code. The difference with left and right does not matter instead what was used was the difference of the upper and the lower areas. The sign of the region was however used in the new setup. can be seen in figure 4

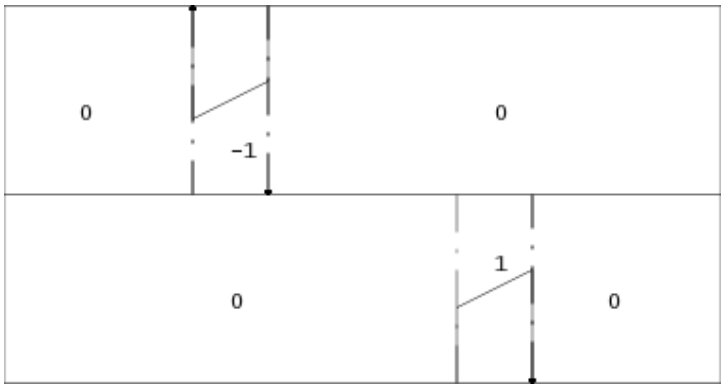

Figure 4: The mesh marked after the markup for the masking

The use of this will be explaind later. The reason for this markup is to mask certain regions for

the motion. So not all but only the regions that is going to be moved will move. The motions velocity of the objects are described with a sinusoidal function prescribed with amplitude and period defined in the $dynamicMeshDict$ as

```
1     vector curMotionVel_ =
2         motionVelAmplitude_*
3         Foam::sin(time().value()*M_PI/motionVelPeriod_);
```

This is then used to describe how much the points of the mesh is going to move as can be seen below in a cut out of the code. The $motionMask\_$ contains the information of the masking shown earlier with 0 , -1 and 1. With this information the points are updated and makes the mesh move to either the left or the right depending on the value of $motionMask\_$. The commented line is what the code used to look like before the changes and because of the $mag(motionMask\_)$ it did not matter if the area where marked with -1 or 1. That, however, is now the whole point of the mark up and the motion mask. The function $pos()$ is a boolean function that returns a 1 if the value inside is positive and a 0 if it is negative.

```
1   newPoints =
2             points() + (
3                 //pos(-0.5+mag(motionMask_)) // cells above the body
4                 -pos(-0.5-motionMask_) // Upper
5                 +pos(-0.5+motionMask_) // Below
6             )*curMotionVel_*time().deltaT().value();
7
8     curLeft_ += curMotionVel_.x()*time().deltaT().value();
9     curRight_ += curMotionVel_.x()*time().deltaT().value();
10    curLeftUpper_ -= curMotionVel_.x()*time().deltaT().value();
11    curRightUpper_ -= curMotionVel_.x()*time().deltaT().value();
```

Note also the change of the last two lines where there is a $-=$ instead of $+=$

Result and Discussion

The result can be seen as two snapshots figure 5.

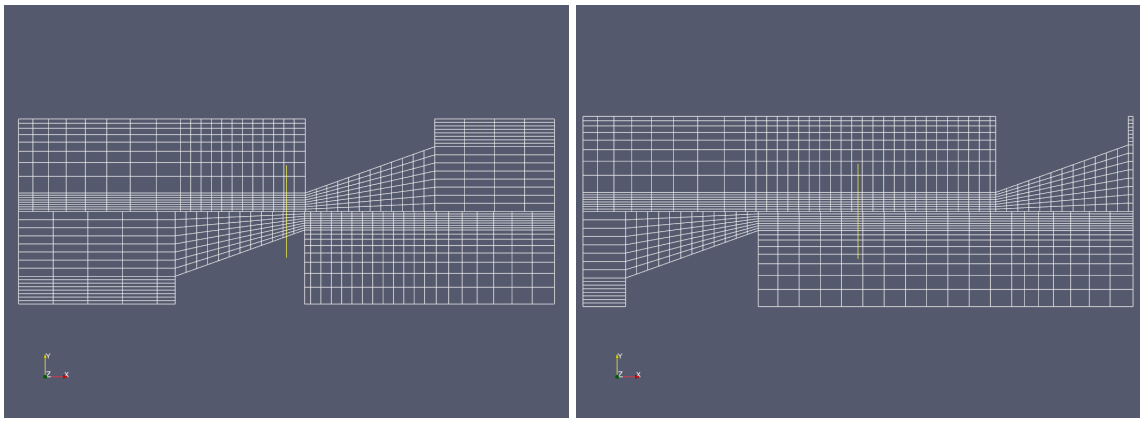

Figure 5: The movement of the mesh at different timesteps

The settings for these movements can be seen in the $dynamicMeshDict$ where it can be noted that there is a difference of the thickness of the cells added and removed. Because it is based on a sinusoidal function the objects will also back up and move towards each other again. If one only want one turn the remedy is to put the $motionVelPeriod$ in the $dynamicMeshDict$ to the same as the $endTime$ in the $controlDict$. One big flaw of the library at this point is that the position of the faces are hardcoded. If one wants to do changes in the mesh the library has to be changed and altered as well. This was the case already in the original implementation in OpenFOAM-1.4.1-dev. Still it should probably be changed in order for the library to be useful. The next step to do for this case is to add the possibility of flow over the sliding interface. At this point arbitrary boundary conditions are used on all the faces and no consideration is taken to the calculations which was also mentioned in the beginning of the report.

Appendix

In this section some of the most important files codes are shown

The dynamicFvMeshDict

The complete dynamicFvMeshDict.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.4                                   |
|   \\  /    A nd           | Web:      http://www.openfoam.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/

FoamFile
{
    version         2.0;
    format          ascii;

    root            "";
    case            "";
    instance        "";
    local           "";

    class           dictionary;
    object          dynamicMeshDict;
}

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

dynamicFvMeshLibs   1("libslidingConesFvMesh.so");

dynamicFvMesh       slidingConesFvMesh;

slidingConesFvMeshCoeffs
{
    motionVelAmplitude      (-0.001 0 0);
    motionVelPeriod         10;

    leftEdge                -0.00145;
    leftObstacleEdge        -0.007;
    rightObstacleEdge       -0.0035;
    leftUpperObstacleEdge   -0.010;
    rightUpperObstacleEdge  -0.0065;


```

```
41
42      left
43      {
44          minThickness        6e-4;
45          maxThickness        8e-4;
46      }
47
48      right
49      {
50          minThickness        6e-4;
51          maxThickness        8e-4;
52      }
53
54      leftUpper
55      {
56          minThickness        3e-4;
57          maxThickness        4e-4;
58      }
59
60      rightUpper
61      {
62          minThickness        3e-4;
63          maxThickness        4e-4;
64      }
65  }
66
67
68  // ************************************************************************* //
```

The slidingConesFvMesh.C

The complete slidingConesFvMesh.C file.

```
1  /*---------------------------------------------------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  1.4                                   |
5  |   \\  /    A nd           | Web:      http://www.openfoam.org               |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  License
9      This file is part of OpenFOAM.
10     OpenFOAM is free software; you can redistribute it and/or modify it
11     under the terms of the GNU General Public License as published by the
12     Free Software Foundation; either version 2 of the License, or (at your
13     option) any later version.
14     OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
15     ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
```

```
16      FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
17      for more details.
18      You should have received a copy of the GNU General Public License
19      along with OpenFOAM; if not, write to the Free Software Foundation,
20      Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
21  \*---------------------------------------------------------------------------*/
22  #include "slidingConesFvMesh.H"
23  #include "Time.H"
24  #include "mapPolyMesh.H"
25  #include "layerAdditionRemoval.H"
26  #include "addToRunTimeSelectionTable.H"
27  #include "volMesh.H"
28  // * * * * * * * * * * * * * * Static Data Members * * * * * * * * * * * * * //
29
30  namespace Foam
31  {
32      defineTypeNameAndDebug(slidingConesFvMesh, 0);
33      addToRunTimeSelectionTable
34      (
35          topoChangerFvMesh,
36          slidingConesFvMesh,
37          IOobject
38      );
39  }
40
41
42  // * * * * * * * * * * * * * * Private Member Functions  * * * * * * * * * * * * * //
43
44  Foam::tmp<Foam::scalarField> Foam::slidingConesFvMesh::vertexMarkup
45  (
46      const pointField& p,
47      const scalar& curLeft,
48      const scalar& curRight,
49      const scalar& curLeftUpper,
50      const scalar& curRightUpper
51  ) const
52  {
53      Info<< "Updating vertex markup.  curLeft: "
54          << curLeft << " curRight: " << curRight
55      << " curLeftUpper: " << curLeftUpper
56      << " curRightUpper: " << curRightUpper << endl;
57
58      tmp<scalarField> tvertexMarkup(new scalarField(p.size()));
59      scalarField& vertexMarkup = tvertexMarkup();
60
61      forAll (p, pI)
62      {
63          if      (p[pI].x() > curLeft - 1e-10 &&
64                   p[pI].x() < curRight + 1e-10 &&
65                   p[pI].y() < 0.0025)
66              vertexMarkup[pI] = 1;
```

```
67          else if (p[pI].x() > curLeftUpper - 1e-10 &&
68                   p[pI].x() < curRightUpper + 1e-10 &&
69                   p[pI].y() > 0.0025)
70              vertexMarkup[pI] = -1;
71          else
72              vertexMarkup[pI] = 0;
73      }
74      return tvertexMarkup;
75  }
76
77
78  void Foam::slidingConesFvMesh::addZonesAndModifiers()
79  {
80      // Add zones and modifiers for motion action
81
82      if
83      (
84          pointZones().size() > 0
85       || faceZones().size() > 0
86       || cellZones().size() > 0
87      )
88      {
89          Info<< "void slidingConesFvMesh::addZonesAndModifiers() : "
90              << "Zones and modifiers already present.  Skipping."
91              << endl;
92
93          if (topoChanger_.size() == 0)
94          {
95              FatalErrorIn
96              (
97                  "void slidingConesFvMesh::addZonesAndModifiers()"
98              )   << "Mesh modifiers not read properly"
99                  << abort(FatalError);
100         }
101
102         return;
103     }
104
105     Info<< "Time = " << time().timeName() << endl
106         << "Adding zones and modifiers to the mesh" << endl;
107
108     const vectorField& fc = faceCentres();
109     const vectorField& fa = faceAreas();
110
111     labelList zone1(fc.size());
112     boolList flipZone1(fc.size(), false);
113     label nZoneFaces1 = 0;
114     labelList zone2(fc.size());
115     boolList flipZone2(fc.size(), false);
116     label nZoneFaces2 = 0;
117     labelList zone3(fc.size());
```

```
118     boolList flipZone3(fc.size(), false);
119     label nZoneFaces3 = 0;
120     labelList zone4(fc.size());
121     boolList flipZone4(fc.size(), false);
122     label nZoneFaces4 = 0;
123
124     forAll (fc, faceI)
125     {
126         if
127         (
128             fc[faceI].x() > -0.003501
129          && fc[faceI].x() < -0.003499
130          && fc[faceI].y() < 0.0025
131         )
132         {
133             if ((fa[faceI] & vector(1, 0, 0)) < 0)
134                 flipZone1[nZoneFaces1] = true;
135
136             zone1[nZoneFaces1] = faceI;
137             Info<< "face " << faceI << " for zone 1.  Flip: "
138                 << flipZone1[nZoneFaces1] << endl;
139             nZoneFaces1++;
140         }
141         else if
142         (
143             fc[faceI].x() > -0.00701
144          && fc[faceI].x() < -0.00699
145          && fc[faceI].y() < 0.0025
146         )
147         {
148             zone2[nZoneFaces2] = faceI;
149
150             if ((fa[faceI] & vector(1, 0, 0)) > 0)
151                 flipZone2[nZoneFaces2] = true;
152
153             Info << "face " << faceI << " for zone 2.  Flip: "
154                 << flipZone2[nZoneFaces2] << endl;
155             nZoneFaces2++;
156         }
157         else if
158         (
159             fc[faceI].x() > -0.006501
160          && fc[faceI].x() < -0.006499
161          && fc[faceI].y() > 0.0025
162         )
163         {
164             zone3[nZoneFaces3] = faceI;
165
166             if ((fa[faceI] & vector(1, 0, 0)) < 0)
167                 flipZone3[nZoneFaces3] = true;
168
```

```
169             Info << "face " << faceI << " for zone 3.  Flip: "
170                 << flipZone3[nZoneFaces3] << endl;
171             nZoneFaces3++;
172         }
173         else if
174         (
175             fc[faceI].x() > -0.0101
176          && fc[faceI].x() < -0.00999
177          && fc[faceI].y() > 0.0025
178         )
179         {
180             zone4[nZoneFaces4] = faceI;
181
182             if ((fa[faceI] & vector(1, 0, 0)) > 0)
183                 flipZone4[nZoneFaces4] = true;
184
185             Info << "face " << faceI << " for zone 4.  Flip: "
186                 << flipZone4[nZoneFaces4] << endl;
187             nZoneFaces4++;
188         }
189     }
190
191     zone1.setSize(nZoneFaces1);
192     flipZone1.setSize(nZoneFaces1);
193     zone2.setSize(nZoneFaces2);
194     flipZone2.setSize(nZoneFaces2);
195     zone3.setSize(nZoneFaces3);
196     flipZone3.setSize(nZoneFaces3);
197     zone4.setSize(nZoneFaces4);
198     flipZone4.setSize(nZoneFaces4);
199     Info << "zone: " << zone1 << endl;
200     Info << "zone: " << zone2 << endl;
201     Info << "zone: " << zone3 << endl;
202     Info << "zone: " << zone4 << endl;
203
204     List<pointZone*> pz(0);
205     List<faceZone*> fz(4);
206     List<cellZone*> cz(0);
207
208     label nFz = 0;
209
210     fz[nFz] =
211         new faceZone
212         (
213             "rightExtrusionFaces",
214             zone1,
215             flipZone1,
216             nFz,
217             faceZones()
218         );
219     nFz++;
```

```
220     fz[nFz] =
221         new faceZone
222         (
223             "leftExtrusionFaces",
224             zone2,
225             flipZone2,
226             nFz,
227             faceZones()
228         );
229     nFz++;
230     fz[nFz] =
231         new faceZone
232         (
233             "rightUpperExtrusionFaces",
234             zone3,
235             flipZone3,
236             nFz,
237             faceZones()
238         );
239     nFz++;
240     fz[nFz] =
241         new faceZone
242         (
243             "leftUpperExtrusionFaces",
244             zone4,
245             flipZone4,
246             nFz,
247             faceZones()
248         );
249     nFz++;
250     fz.setSize(nFz);
251     Info << "Adding mesh zones." << endl;
252     addZones(pz, fz, cz);
253     // Add layer addition/removal interfaces
254     topoChanger_.setSize(4);
255     label nMods = 0;
256     topoChanger_.set
257     (
258         0,
259         new layerAdditionRemoval
260         (
261             "right",
262             nMods,
263             topoChanger_,
264             "rightExtrusionFaces",
265             readScalar(motionDict_.subDict("right").lookup("minThickness")),
266             readScalar(motionDict_.subDict("right").lookup("maxThickness"))
267         )
268     );
269     nMods++;
270     topoChanger_.set
```

```
271     (
272         1,
273         new layerAdditionRemoval
274         (
275             "left",
276             nMods,
277             topoChanger_,
278             "leftExtrusionFaces",
279             readScalar(motionDict_.subDict("left").lookup("minThickness")),
280             readScalar(motionDict_.subDict("left").lookup("maxThickness"))
281         )
282     );
283     nMods++;
284     topoChanger_.set
285     (
286         2,
287         new layerAdditionRemoval
288         (
289             "rightUpper",
290             nMods,
291             topoChanger_,
292             "rightUpperExtrusionFaces",
293             readScalar(
294                 motionDict_.subDict("rightUpper").lookup("minThickness")
295                ),
296             readScalar(
297                 motionDict_.subDict("rightUpper").lookup("maxThickness")
298            )
299         )
300     );
301     nMods++;
302     topoChanger_.set
303     (
304         3,
305         new layerAdditionRemoval
306         (
307             "leftUpper",
308             nMods,
309             topoChanger_,
310             "leftUpperExtrusionFaces",
311             readScalar(
312                 motionDict_.subDict("leftUpper").lookup("minThickness")
313                 ),
314             readScalar(
315                 motionDict_.subDict("leftUpper").lookup("maxThickness")
316            )
317         )
318     );
319     nMods++;
320     Info << "Adding " << nMods << " mesh modifiers" << endl;
321     // Write mesh modifiers
```

```
322     topoChanger_.write();
323 }
324
325
326 // * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * * //
327
328 // Construct from components
329 Foam::slidingConesFvMesh::slidingConesFvMesh(const IOobject& io)
330 :
331     topoChangerFvMesh(io),
332     motionDict_
333     (
334         IOdictionary
335         (
336             IOobject
337             (
338                 "dynamicMeshDict",
339                 time().constant(),
340                 *this,
341                 IOobject::MUST_READ,
342                 IOobject::NO_WRITE
343             )
344         ).subDict(typeName + "Coeffs")
345     ),
346     motionVelAmplitude_(motionDict_.lookup("motionVelAmplitude")),
347     motionVelPeriod_(readScalar(motionDict_.lookup("motionVelPeriod"))),
348     /*curMotionVel_
349     (
350         motionVelAmplitude_*
351         Foam::sin(time().value()*M_PI/motionVelPeriod_)
352     ),*/
353     leftEdge_(readScalar(motionDict_.lookup("leftEdge"))),
354     curLeft_(readScalar(motionDict_.lookup("leftObstacleEdge"))),
355     curRight_(readScalar(motionDict_.lookup("rightObstacleEdge"))),
356     curLeftUpper_(
357                     readScalar(motionDict_.lookup("leftUpperObstacleEdge"))
358       ),
359     curRightUpper_(
360                      readScalar(motionDict_.lookup("rightUpperObstacleEdge"))
361        ),
362     motionMask_
363     (
364         vertexMarkup
365         (
366             points(),
367             curLeft_,
368             curRight_,
369             curLeftUpper_,
370             curRightUpper_
371         )
372     )
```

```
373 {
374     addZonesAndModifiers();
375     curLeft_ = average(
376         faceZones()[
377             faceZones().findZoneID("leftExtrusionFaces")
378         ]().localPoints()).x();
379     curRight_ = average(
380         faceZones()[
381             faceZones().findZoneID("rightExtrusionFaces")
382         ]().localPoints()).x();
383     curLeftUpper_ = average(
384         faceZones()[
385             faceZones().findZoneID("leftUpperExtrusionFaces")
386         ]().localPoints()).x();
387     curRightUpper_ = average(
388         faceZones()[
389             faceZones().findZoneID("rightUpperExtrusionFaces")
390         ]().localPoints()).x();
391 }
392
393 // * * * * * * * * * * * * * * * * Destructor  * * * * * * * * * * * * * * * * //
394 Foam::slidingConesFvMesh::~slidingConesFvMesh(){}
395
396 // * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * * //
397
398 bool Foam::slidingConesFvMesh::update()
399 {
400     autoPtr<mapPolyMesh> topoChangeMap = topoChanger_.changeMesh();
401
402     // Calculate the new point positions depending on whether the
403     // topological change has happened or not
404     pointField newPoints;
405
406     vector curMotionVel_ =
407         motionVelAmplitude_*
408         Foam::sin(time().value()*M_PI/motionVelPeriod_);
409
410     bool meshChanged = topoChangeMap.valid();
411
412     if (meshChanged)
413     {
414         Info << "Topology change. Calculating motion points" << endl;
415
416         if (topoChangeMap().hasMotionPoints())
417         {
418             motionMask_ =
419                 vertexMarkup
420                 (
421                     topoChangeMap().preMotionPoints(),
422                     curLeft_,
423                     curRight_,
```

```
                    curLeftUpper_,
                    curRightUpper_
                );
        }
        else
        {
            motionMask_ =
                vertexMarkup
                (
                    points(),
                    curLeft_,
                    curRight_,
                    curLeftUpper_,
                    curRightUpper_
                );
        }
        // Correct mesh motion for correct volume continuity
        movePoints(topoChangeMap().preMotionPoints());
        resetMotion();
        setV0();
    }
    else
    {
        Info << "No topology change" << endl;
        // Set the mesh motion
    }
  newPoints =
            points() + (
                //pos(-0.5+mag(motionMask_)) // cells above the body
                -pos(-0.5-motionMask_) // Upper
                +pos(-0.5+motionMask_) // Below
//                + pos(motionMask_ - 0.5)*  // cells in front of the body
//                  (points().component(vector::X)/curRight)
//                + pos(-motionMask_ - 0.5)*  // cells behind the body
//                  ((points().component(vector::X)-leftEdge)
//                   /(curLeft_ - leftEdge_))
            )*curMotionVel_*time().deltaT().value();

    curLeft_ += curMotionVel_.x()*time().deltaT().value();
    curRight_ += curMotionVel_.x()*time().deltaT().value();
    curLeftUpper_ -= curMotionVel_.x()*time().deltaT().value();
    curRightUpper_ -= curMotionVel_.x()*time().deltaT().value();
    // The mesh now contains the cells with zero volume
    Info << "Executing mesh motion" << endl;
    movePoints(newPoints);
    //  The mesh now has got non-zero volume cells
    return meshChanged;
}
// ************************************************************************* //
```

The slidingConesFvMesh.H

The complete slidingConesFvMesh.H file.

```
#ifndef slidingConesFvMesh_H
#define slidingConesFvMesh_H
#include "topoChangerFvMesh.H"
#include "motionSolver.H"
namespace Foam
{
class slidingConesFvMesh:public topoChangerFvMesh
{
    // Private data
        //- Motion dictionary
        dictionary motionDict_;
        //- Motion velocity amplitude
        vector motionVelAmplitude_;
        //- Motion velocity period
        scalar motionVelPeriod_;
        //- Motion velocity period //unnecessary, overwritten in the .C/R33K
        //vector curMotionVel_;
        //- Left edge
        scalar leftEdge_;
        //- Current left obstacle position
        scalar curLeft_;
        //- Current right obstacle position
        scalar curRight_;
        //- Current left Upper obstacle position
        scalar curLeftUpper_;
        //- Current right Upper obstacle position
        scalar curRightUpper_;
        //- Vertex motion mask
        scalarField motionMask_;
    // Private Member Functions
        //- Disallow default bitwise copy construct
        slidingConesFvMesh(const slidingConesFvMesh&);
        //- Disallow default bitwise assignment
        void operator=(const slidingConesFvMesh&);
        //- Add mixer zones and modifiers
        void addZonesAndModifiers();
        //- Markup motion vertices
        tmp<scalarField> vertexMarkup
        (
            const pointField& p,
            const scalar& curLeft,
            const scalar& curRight,
            const scalar& curLeftUpper,
            const scalar& curRightUpper
        ) const;
public:
```

```
47      //- Runtime type information
48      TypeName("slidingConesFvMesh");
49      // Constructors
50          //- Construct from database
51          explicit slidingConesFvMesh(const IOobject& io);
52      // Destructor
53          virtual ~slidingConesFvMesh();
54      // Member Functions
55          //- Update the mesh for both mesh motion and topology change
56          virtual bool update();
57  };
58  } // End namespace Foam
59  #endif
```