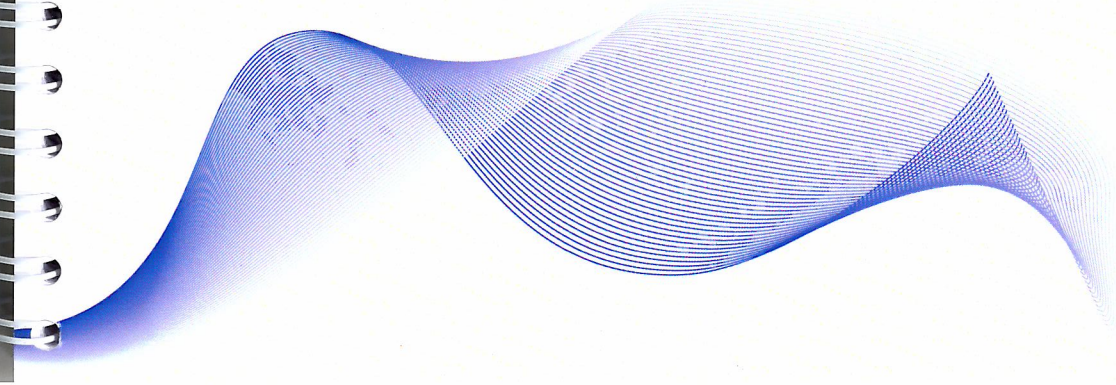
OpenFOAM®

OpenFOAM course

Overset Mesh

Virtual

esi



OpenFOAM

The open source CFD toolbox

Dynamic mesh and Overset Course

OpenCFD Ltd

Notes v1712 rev 2.0. 4/2/2018

Copyright and Disclaimer

Contents

1 Overview

1.1 Dynamic Mesh Treatment

Motivation

- Interaction of the flowing media with the objects obstructing the flow
- Simulation of a flow induced by moving boundaries (turbine, compressor)
- Simulation of body movement induced by the flow (ship hull floating on a water surface)
- Fluid structure interaction – including deformation of the solid

Type of the dynamic mesh treatment

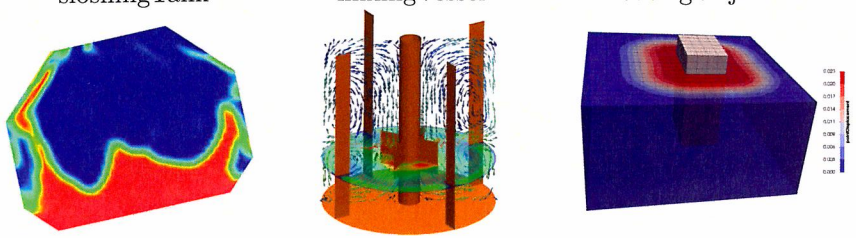
- Before studying the overset mesh implementation in OpenFOAM, we will review the dynamic mesh functionality in OpenFOAM
- Dynamic mesh functionality is driven by settings in `dynamicMeshDict`
- Here we select the type of dynamic mesh treatment (refinement, solid body movement, overset ...) by setting the `dynamicFvMesh` to one of the possible classes
- Then we select the motion solver and the motion function

Types of dynamic mesh treatment in OpenFOAM— `dynamicMotionSolverFvMesh`

- Mesh can move as a solid object (sloshing tank)
  - tutorial: "sloshingTank.*" under the `interDyMFoam` solver
- Domain may consist of moving and static meshes. The interface between these sections needs to be created with Arbitrary Mesh Interface AMI
  - tutorial: mixerVesselAMI2D under the `pimpleDyMFoam` solver
- Section of the mesh can deform or morph while other section is static.

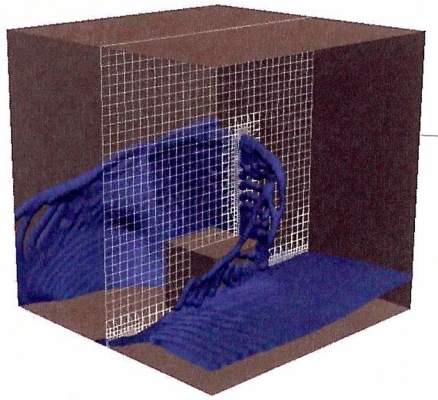
  - tutorial: floatingObject under the `interDyMFoam` solver

sloshingTank mixingVessel floatingObject



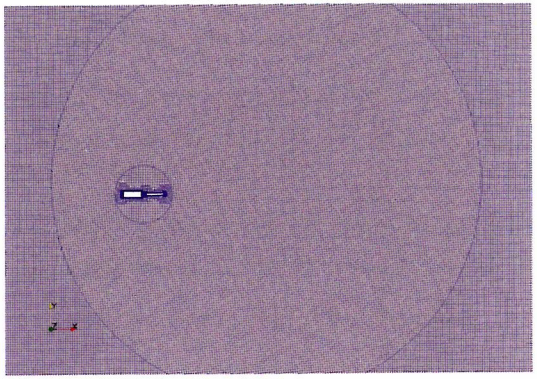
Types of dynamic mesh treatment in OpenFOAM— `dynamicRefineFvMesh`

- Number of mesh points is changing ([un]refinement) — mesh density follows scalar value
  - `dynamicFvMeshClass: dynamicRefineFvMesh`
  - tutorial: damBreakWithObstacle under the `interDyMFoam` solver



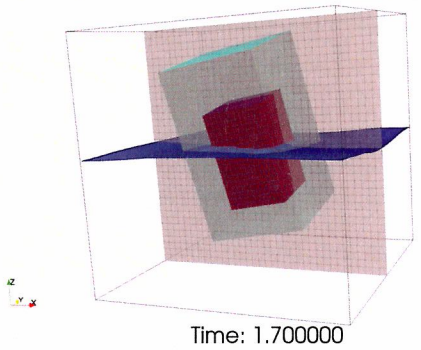
Types of dynamic mesh treatment in OpenFOAM— `dynamicMultiMotionSolverFvMesh`

- Individual cellZones undergoes specific mesh movements
  - `dynamicFvMeshClass: dynamicMultiMotionSolverFvMesh`
  - tutorial: box2D_moveDynamicMesh under the `moveDynamicMesh` solver ...
  - ... where multiple rotation cell zones are divided by `AMI` interfaces



Types of dynamic mesh treatment in OpenFOAM— `dynamicOversetFvMesh`

- Overset mesh moves with the solid body on top of the background mesh (sometimes called chimera grid)
  - `dynamicFvMeshClass: dynamicOversetFvMesh`
  - In the 1706 version can be used only with solvers containing keyword "overset"
  - Any type of mesh motion can be used with overset mesh
  - Overset mesh can be used with steady state solvers for parametric studies as well for mesh motion transient studies
  - Overset mesh will be explained in a detail later

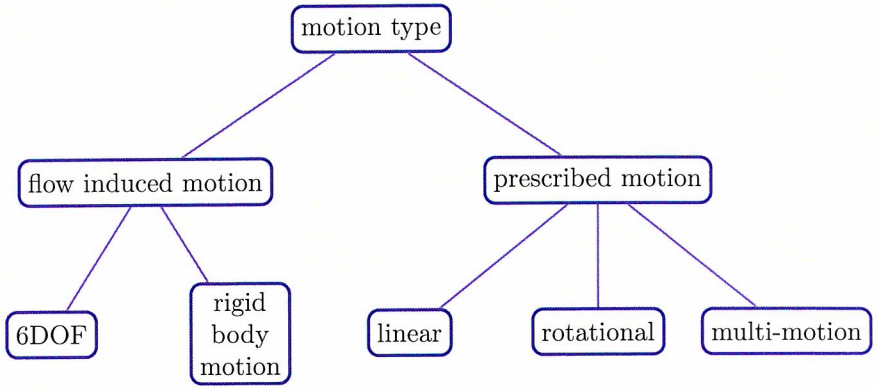


There is also one more library for topological changes in the mesh, but it is now obsolete, superseded by combination of solid body movement with AMI or ACMI.

1.2 Motion types

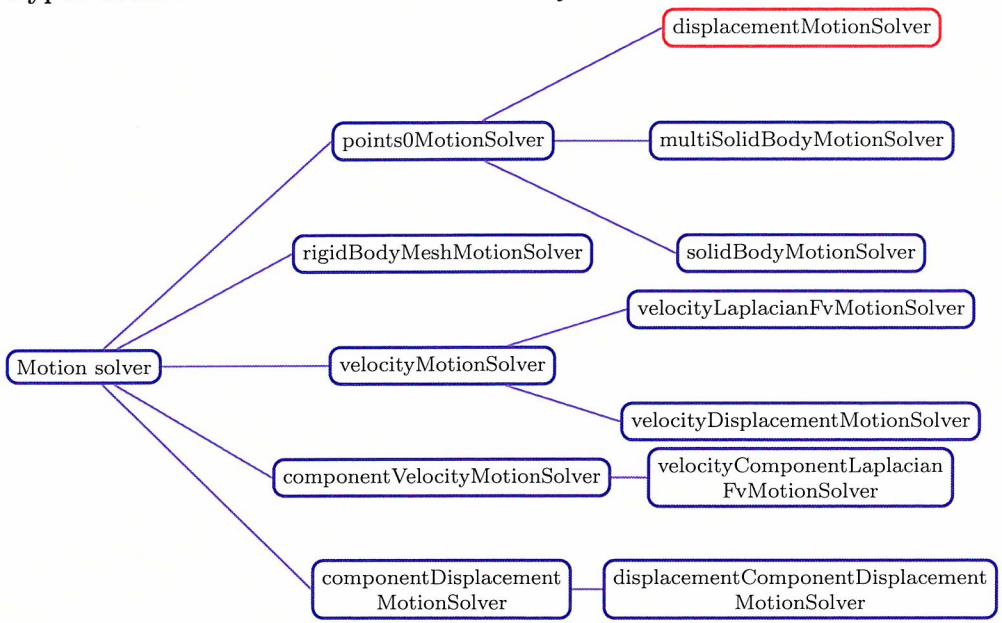
Class hierarchy

- Once the mesh treatment is chosen we must describe the mesh motion
- From the functional point of view:

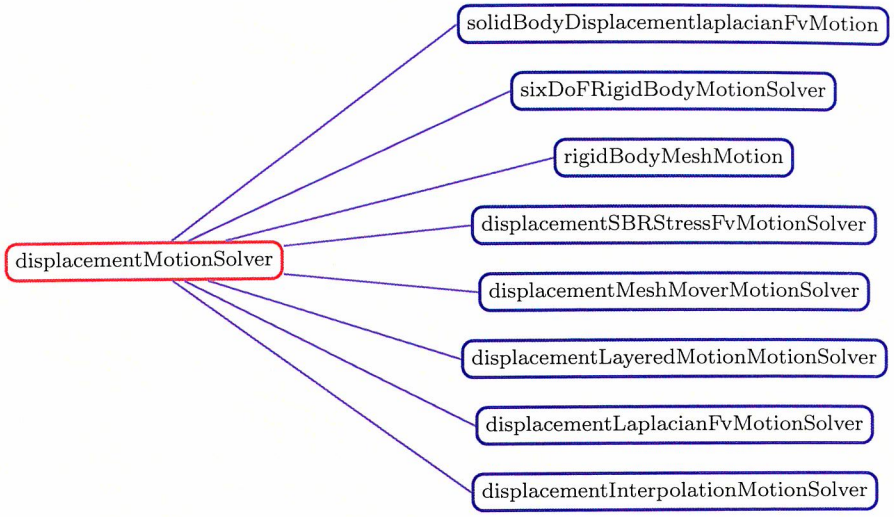


- Motion is described by various classes allowing diffusive morphing of the mesh points, topological changes or moving the mesh as a solid block
- The modelling approach and motion solver needs to be specified in the `constant/dynamicMeshDict` dictionary

Types of motion solvers — class hierarchy



Types of motion solvers — class hierarchy — `displacementMotionSolver`



- Details could be found in
  - Tutorials
  - Constructors of the appropriate class source code

To list the tutorials with various solvers use *e.g.*

```
>> find $FOAM_TUTORIALS -name dynamicMeshDict |xargs grep solver
```

- Solvers manipulating with the mesh points position are used typically with 6DOF or 3DOF approaches; in addition, tutorial `snakeRiverCanyon` shows how the mesh could be mapped onto the surface; diffusive (laplacian) solver (displacementMotionSolver) is also used in snappyHexMesh to displace the snapped mesh to create room for prismatic layers. (see snappyHexMesh section of Extended source guide documentation for details)

How to set and use dynamic mesh

- Dynamic mesh can be used in conjunction with any solver with DyM in the name of the solver

```
>> find $FOAM_SOLVERS -name '*DyM*'
```

- The type of motion and solvers are runtime selectable except the overset
- The main driving file is `dynamicMeshDict`
- If relative motion of points takes place (*e.g.* 6DOF, only part of the domain is moving, overset) the `pointDisplacement` file is needed as well
- Overset dynamic mesh can be used currently only with dedicated solvers:

  overPotentianFoam — potential flow with stationary mesh
  overLaplacianDyMFoam — laplacianFoam (moving mesh)
  overSimpleFoam — stationary mesh
  overPimpleDyMFoam — incompressible flow, moving mesh
  overRhoSimpleDyMFoam — compressible flow, stationary mesh
  overRhoPimpleDyMFoam — compressible flow, moving mesh
  overInterDyMFoam — incompressible VOF, moving mesh

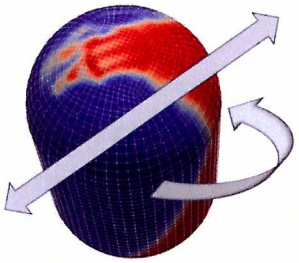
- Let us see some tutorials in detail, first solid body movement and later overset

2 Dynamic Meshes in OpenFOAM

2.1 Prescribed motion tutorial

Overview

- Tutorial `sloshingCylinder` shows simple setup of two combined motion on the whole mesh
- Tutorial for `interDyMFoam` solvers flow inside moving capsule is located in `$FOAM_TUTORIALS/multiphase/interDyMFoam/laminar`
- Movement is a combination of oscillating linear motion and rotation



- Dynamic motion is prescribed only in `dynamicMeshDict`
- Whole mesh is moved as a solid body, therefore only the `polyMesh/points` is stored in each time directory
- Water is sloshing inside the capsule as an effect of the movement

`dynamicMeshDict`

- File `dynamicMeshDict` is located in `constant` directory
- has two main sections: selection of the `dynamicFvMesh` library and the solver selection

```
17 dynamicFvMesh   dynamicMotionSolverFvMesh;
18
19 motionSolver    solidBody; // prescribed mesh motion library
20
21 solidBodyMotionFunction multiMotion;
22
23 oscillation
24 {
25     solidBodyMotionFunction oscillatingLinearMotion;
26     oscillatingLinearMotionCoeffs
27     {
28         amplitude     (0.1 0 0);
29         omega         18.8945578;
30     }
31 }
32
33 rotation
34 {
35     solidBodyMotionFunction  rotatingMotion;
36     rotatingMotionCoeffs
37     {
38         origin        (0 0.02 0);
39         axis          (0 0 1);
40         omega         18.8945578;
41     }
42 }
```

Runing the case

- Copy the case from tutorial:

```
>> run
>> cp -r $FOAM_TUTORIALS/multiphase/interFoam/laminar/sloshingCylinder .
>> cd sloshingCylinder
>> cp $FOAM_TUTORIALS/multiphase/interFoam/laminar/damBreak/\
 damBreak/system/decomposeParDict .
```

- Change two last lines of `Allrun` script to:

```
9  runApplication decomposePar
10 runParallel $(getApplication)
```

- ...to run the solver in parallel

- Before running the tutorial, let us modify the rotation of the capsule to a `Function1` entry as:

```
omega           table
(
    (0      0)
    (0.1    0.1)
    (1      40)
);
```

- ... which will start with rotation more smoothly

Running the case

- Inspect the log file for details of the used setup

```
Selecting dynamicFvMesh dynamicMotionSolverFvMesh
Selecting motion solver: solidBody
Selecting solid-body motion function multiMotion
Selecting solid-body motion function oscillatingLinearMotion
Constructed SBMF 0 : oscillation of type oscillatingLinearMotion
Selecting solid-body motion function rotatingMotion
Constructed SBMF 1 : rotation of type rotatingMotion
Applying solid body motion to entire mesh
```

- Before running the solver, we can test the mesh motion setup by running:
  - `moveMesh` which will execute only the mesh motion and save the mesh motion into the time directories according to the settings in `controlDict`
  - `moveDynamicMesh` solver, who will return also the mesh quality parameters every iteration

Other `solidBodyMotionFunctions`

- Prescribed motion specified in `dynamicMeshDict`
- `solidBodyMotionFunction` specifies choice of motion function, from

Motion	Description
`linearMotion`	Constant velocity translation
`rotatingMotion`	Constant velocity rotation about C of G
`oscillatingLinearMotion`	Oscillating displacement
`oscillatingRotatingMotion`	Oscillating rotation
`tabulated6DoFMotion`	Tabulated 6DoF motion function
`SDA`	Ship design analysis 3DoF motion
`multiMotion`	Combination of the above

`Function1` entries in setup (1)

- The entries for `omega` in `rotatingMotion` could could be `Function1` –time dependent entries
- However only few functions could be used (those with implemented integration function):
  - inline table

```
omega      table
(
    (  0    0.0)
    (100   10.0)
);
```

  - OpenFOAM table file

```
omega      tableFile;
file       "$FOAM_CASE/myDataFile"
outOfBounds  clamp;
```

`Function1` entries in setup (2)

- CSV file

```
omega             csvFile;
file              "$FOAM_CASE/myDataFile"
outOfBounds       clamp;
nHeaderLine       1;
refColumn         0;
mergeSeparators   false;
componentColumns  (1);
```

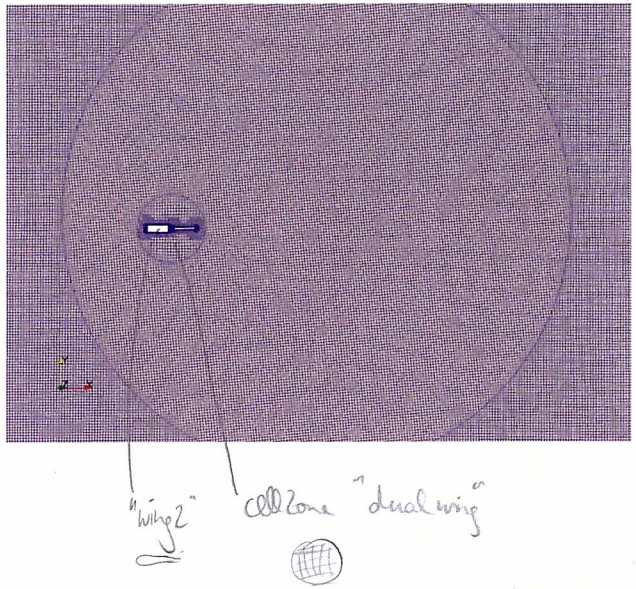
- Polynomial

```
omega      polynomial // y = 0.1 + 1.3x^2 + 2.7x^3
(
    (0.1     0)
    (1.3   2.0)
    (2.7   3.0)
);
```

- Zero and One

```
omega             zero; // or: omega         one;
```

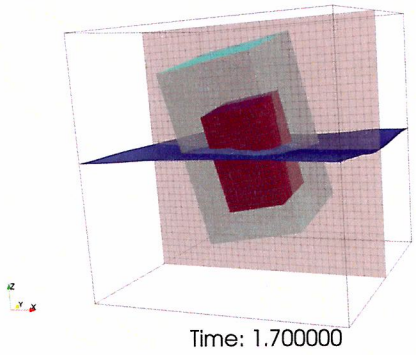
- Some of the types of mesh motions can be set via `pointDisplacement` file
- Mainly the 6DOF or prescribed motion function
- Tutorial `relativeMotion` for `moveDynamicMesh` solver shows combination of prescribed motions nested in a combination of motions



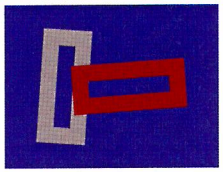
3 Overset mesh

Overset mesh – Overview

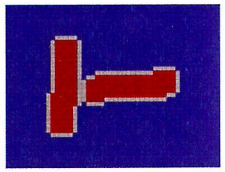
- Introduces cell-to-cell mapping between overlapping cells
- Requires two sets of mesh
  - Background static mesh holding typically the external BCs
  - Overset mesh with the patch representing the moving body
- New boundary condition `overset` and patch type `overset` are surrounding the overset part of the mesh
- Overset mesh is moving together with the object in question using any mesh motion solver
  - 6DOF
  - rigid body
  - solid body motion
  - laplacian point displacement
- Mapping is used to determine which part of the mesh is calculated, interpolated or holes (not used part)
- Overset mesh can be used for static mesh computations to allow easy parametric studies
  - One part with different position (*e.g.* angle of attack)
  - Different shape of the object in the similar volume and common interface to the rest of the geometry
- Dynamic mesh computations without deformation of the mesh (thus allowing much larger freedom in movement)



Overset mesh – Implementaion



problems when there are more than 2 overset grids are overlapping



- Picture on right – Background mesh depicted in blue; grey cells with the flux interpolation; red cells with flux blockage (representing the hole)
- Mesh zones are checked for overlapping and geometrical constrains (*e.g.* walls empty ...)
- Boundary overlapping cells and cells not reachable from non-constraint patches are marked as holes (inactive cells with no flow across)
- Cell based stencil is added to any matrix providing fully implicit coupling or can be used explicitly in the top-level solvers to interpolate any additional user supplied fields
- Performance and robustness keep increasing with the continuous development

explicit or implicit coupling
default

Stencils

- Stencils are used in two ways:
  - added to the matrix providing fully implicit coupling
  - on a solver level provides explicit pressure–velocity coupling and user supplied fields via `oversetInterpolationRequired`
- There are several stencils implemented: (for interpolation between meshes → donor/acceptor cells)

cellVolumeWeight — 1st order, hole detection problems

more robust at the moment

inverseDistance — 2nd order, bounded, generally recommended

leastSquares — 2nd order, potentially unbounded

↳ method for taking from the donor cells

helper mesh is used: a "voxel mesh"

Overset solvers

- To address various use of overset mesh, it is implemented with several solvers
- In the latest release the overset mesh movement is bound only to specific solvers:

overPotentialFoam — potential flow, stationary mesh
overLaplacianDyMFoam — laplacianFoam (moving mesh)
overSimpleFoam — steady state mesh
overPimpleDyMFoam — incompressible flow, moving mesh
overRhoSimpleFoam — compressible flow, stationary mesh
overRhoPimpleDyMFoam — compressible flow, moving mesh
overInterDyMFoam — incompressible VOF, moving mesh

new method in v1712: tracking Inverse Distance

Setup

- Overset case setup is similar to any other dynamic mesh setup
- Main differences are:
  - Field file `0/zoneID` indicating a mesh zone in which all cells are located
  - Mesh zone specifies also ordering of interpolation in locations of multiple overlapping
  - Typically the coarsest level should be in zone 0
  - Patches next to the interpolated cells should be set to type `overset`
  - Overset stencil method together with the variables to apply the stencil to should be specified in `system/fvSchemes` file
  - In `dynamicMeshDict` the specification of the `dynamicFvMesh` type
  - Typically we use a laplacian displacement with diffusion set to 1
  - Specification of the mesh motion (either in the `dynamicMeshDict` or in the `pointDisplacement` file)

3.1 Simple two boxes tutorial

Overview

- Let us inspect very simple tutorial solving for heat transfer using `overLaplacianDyMFoam`
- Geometry is represented by a background mesh created using `blockMesh` and the hollow square as the overset part
- In `blockMeshDict` these two entities are created as separate blocks

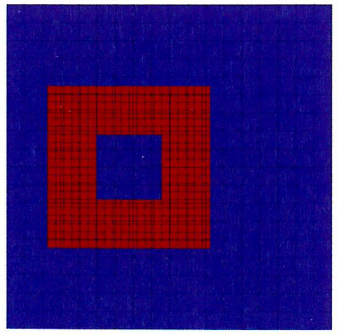
```
51 blocks
52 (
53     hex (0 1 2 3 4 5 6 7) (18 18 1) simpleGrading (1 1 1)
54
55     hex (8 9 10 11 12 13 14 15) movingZone (20 20 1) simpleGrading (1
         1 1)
56 );
```

Mesh

- After building a volumetric mesh, `topoSet` is run to create two cellSets — one for each block
- Utility `subsetMesh` is run to delete a hole and create patch of the name

```
>> subsetMesh box -patch hole -overwrite
```

- Utility `topoSet` is run again to create the cell sets again
- `setFields` then creates the `zoneID` field file with blocks belonging to appropriate groups



creating zoneID

```
defaultFieldValues
(
    volScalarFieldValue zoneID 123
);

regions
(
    // Set cell values
    // (does zerogradient on boundaries)
    cellToCell
    {
        set c0;

        fieldValues
        (
            volScalarFieldValue zoneID 0
        );
    }

    cellToCell
    {
        set c1;

        fieldValues
        (
            volScalarFieldValue zoneID 1
        );
    }

);
```


zoneID file

- Patches description could be seen in `constant/polyMesh/boundary` file
- In `0/zoneID` file the BCs are `zeroGradient` except the patch free which is of type overset
- In field file for temperature, the overset patch is set automatically together with empty BC by including `caseDicts/setConstraintTypes`

```
boundaryField
{
    free
    {
        type            overset;
        value           uniform 1;
    }
    walls
    {
        type            zeroGradient;
    }
    hole
    {
        type            zeroGradient;
    }
    frontAndBack
    {
        type            empty;
    }
    left1
    {
        type            zeroGradient;
    }
    right1
    {
        type            zeroGradient;
    }
}
```

Mesh motion setup

- Since the `dynamicMeshDict` does not specify a motion itself, but the overset library only

```
motionSolverLibs ( "libfvMotionSolvers.so" );

solver          displacementLaplacian;

displacementLaplacianCoeffs
{
    diffusivity     uniform 1;
}

dynamicFvMesh       dynamicOversetFvMesh;
```

- ...motion is specified in the `0/pointDisplacement` file

pointDisplacement

- Patches belonging to the overset part of the mesh have the `patchType` is set to `overset`
- Mesh movement is set via boundary condition using a table method of `Function1` for time changing BCs

```
17 dimensions      [0 1 0 0 0 0 0];
18
19 internalField   uniform (0 0 0);
20
21 boundaryField
22 {
23     ".*"
24     {
25         type            uniformFixedValue;
26         uniformValue    (0 0 0);
27     }
28
29     "(free|hole)"
30     {
31         patchType       overset;
32         type            uniformFixedValue;
33         uniformValue    table
34         (                 x y z
35             (0.0          (0 0 0))
36             (1.0          (0.31 0 0))
37             (2.0          (0 0 0))
38         );
39     }
40 }
```

Additional settings

- Specifying library `liboverset.so` in `controlDict` to list `libs` helps all pre-processing utilities to recognise `overset` boundary condition
- In `fvSchemes` file the stencil type (interpolation method) must be chosen

```
17 ...
18 oversetInterpolation
19 {
20     // Interpolation scheme to use for overset calculation
21     method      inverseDistance;
22 }
```

searchBox (xmin ymin zmin)(xmax ymax zmax);
searchBoxDivisions (100 100 20);
number of divisions in each direction
→ for hdpv/voxel mesh

3.2 Cylinder tutorials

3.2.1 Base case – no moving mesh

Overview

- 2D case with a static cylinder
- We will inspect the tutorial and make some changes to it
- First we will copy the tutorial

```
>> run
>> cp -r $FOAM_TUTORIAL/incompressible/overPimpleDyMFoam/cylinder .
```

- Cylinder geometry provided as VTK geometry file

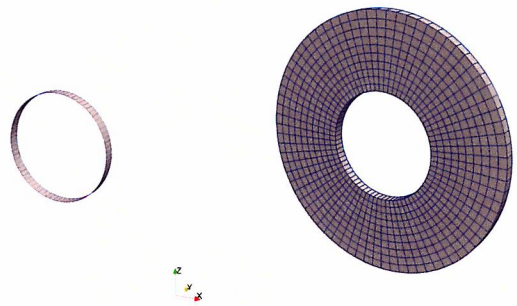
```
/
├── Allclean
├── Allrun
├── Allrun.pre
├── cylinderAndBackground.....case with merged meshes where the flow is solved for
│   ├── 0.orig
│   ├── Allclean
│   ├── Allrun
│   ├── Allrun.pre
│   ├── constant
│   └── system
└── cylinderMesh.....case with the overset geometry and mesh
    ├── Allrun.pre
    ├── constant
    └── system
```

Create Overset Mesh

- Overset mesh must be created in the first step
- In this tutorial we have a file `cylinder.vtk`
- File `Allrun.pre` describes the workflow, if we were to repeat by hand:

```
>> extrudeMesh
>> createPatch -overwrite
```

- Utility `extrudeMesh` reads `system/extrudeMeshDict` file
- ...where extrusion from mesh is chosen with `extrudeModel` is set to `linearNormal`



- Extruding the mesh will create patches on the mesh which needs to be renamed
- Renaming could be done manually or by employing utility `createPatch` reading `createPatchDict`

```
patches
(
    {
        name oversetPatch;

        patchInfo
        {
            type overset;
        }

        constructFrom patches;

        patches (otherSide);
    }

    {
        name walls;

        patchInfo
        {
            type wall;
        }

        constructFrom patches;

        patches (originalPatch);
    }
...
```

- Overset mesh case directory need to contain also dummy standard files as `controlDict`,`fvSolution`, `fvSchemes`

- Main case is typically called `background` or, as it is here, `cylinderAndBackground`
- Before merging the case into the background case, it is necessary to check mesh quality and boundary types and names

Background mesh case

- The background mesh case needs to contain a background mesh (structured or unstructured)
- After the mesh is generated we have to merge the overset mesh using a `mergeMesh` utility

```
>> mergeMeshes . ../cylinderMesh -overwrite
```

- Utility `topoSet` is used here to create distinct `cellSets` which will be turned into `zoneID`
- `topoSet` here first creates a cellSet from a region specified by a point in a background mesh

```
17 actions
18 (
19     {
20         name    c0;
21         type    cellSet;
22         action  new;
23         source  regionToCell;
24         sourceInfo
25         {
26             insidePoints ((-4.999 0 -3.999));
27         }
28     }
```

- Then new `cellSet` `c1` is created as a copy

```
31 {
32     name    c1;
33     type    cellSet;
34     action  new;
35     source  cellToCell;
36     sourceInfo
37     {
38         set c0;
39     }
40 }
```

- and finaly the copy `c1` is inverted to contain the overset mesh

```
{
    name    c1;
    type    cellSet;
    action  invert;
}
);
```

More information on the cell set specification can be found in the Appendix.

Setting zoneID

- Sets created in the previous step are now used to set a `zoneID` volumetric field in the domain
- Field is specified from the `cellSets`

```
cellToCell
{
    set c0;

    fieldValues
    (
        volScalarFieldValue zoneID 0
    );
}
```

Running the case

- Once all fields are set properly we can run the solver (`overPimpleDyMFoam`)
- Warning message at the top of the log file is reporting the overset is not set as a first boundary patch

```
inverseDistance : detected 2 mesh regions
    zone:0 nCells:7200  voxels:(89 1 89) bb:(-5.00002 -0.0500161 -4.00002) (9.00002 0.0500161 4.00002)
    zone:1 nCells:750  voxels:(89 1 89) bb:(-1.19895 -0.0500034 -1.19974) (1.2 0.0500034 1.19974)
--> FOAM Warning :
    From function bool Foam::oversetPolyPatch::master() const
    in file oversetPolyPatch/oversetPolyPatch.C at line 149
    The master overset patch is not the first patch. Generally the first patch should be an overset
        patch to guarantee consistent operation.
Overset analysis : nCells : 7950
    calculated   : 7773
    interpolated : 105 (interpolated from local:105  mixed local/remote:0  remote:0)
    hole         : 72
```

- The warning refers to the recommendation the overset patch should be the first for correct stencil formulation
- Remedy is to add boundary patch entry into the `blockMeshDict`


```
oversetPatch
{
    type overset;
    faces ();
}
```

DebugSwitch for controlDict: (prints more info than just normal log info)

```
DebugSwitches
{
    overset 1;
}
```

3.2.2 Rotating ellipsoid

Overview of the changes

- One of the advantages of overset mesh is quick parametric study
- Here we can easily change the bluff-body to a different shape
- We will deform the mesh in `cylinderMesh` case into an ellipsoid

1. Create a copy of the cylinder case

```
>> run
>> cp -r cylinder cylinderEllipsoid
>> cd cylinderEllipsoid
>> ./Allclean
```

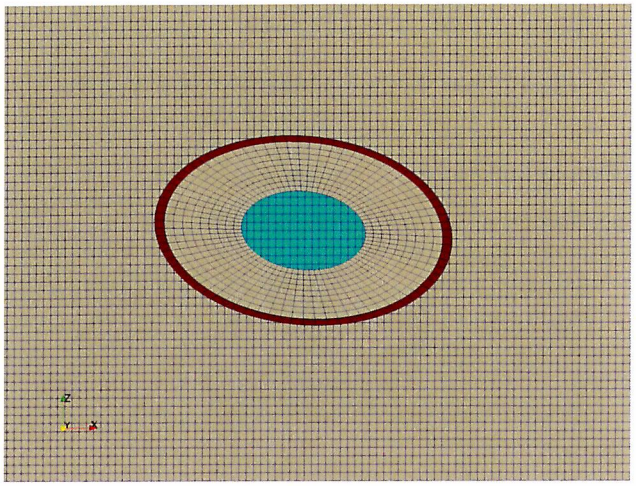
2. In `cylinderMesh` directory deform a mesh

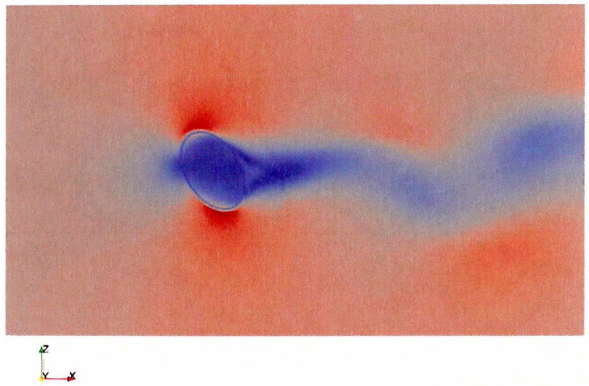
```
>> extrudeMesh
>> transformPoints -scale '( 2.0 1.25 1.25 )'
>> createPatch -overwrite
```

- We must also scale up during the transformation to have enough cells masked
- Once the mesh is deformed we can merge the mesh into the background case

Prescribe a motion

- Motion to the ellipsoid can be prescribed with a `solidBody` Motion in `dynamicMeshDict`





SolidBodyMotion

- We will first setup the movement based on the `solidBodyMotion` function
- `dynamicMeshDict` has the following content

```
dynamicFvMesh       dynamicOversetFvMesh;

dynamicOversetFvMeshCoeffs
{
//    layerRelax 0.3;
}

solver          solidBody;

cellZone                cylinderZone;
solidBodyMotionFunction rotatingMotion;

rotatingMotionCoeffs
    {
        origin      (0.0 0.0 0.0);
        axis        (0 1 0);
        omega       2.0;
    }
```

- For the `solidBody` a `cellZone` must be created which contains all the cells in overset region
- `cellZone` can be created by a `topoSet` where we only need to set extra entry in topoSetDict

```
{
    name    cylinderZone;
    type    cellZoneSet;
    action  new;
    source  setToCellZone;
    sourceInfo
    {
        set c1;
    }

}
```

> `cellZone` can be created by CLI (command line environment) `setSet` (see appendix). By adding the entry to the `topoSetDict` the `cellZone` creation is automated as running `topoSet` is part of the workflow.

- Before running the solver, setup could be tested by executing `moveMesh` or `moveDynamicMesh`, which will perform `checkMesh` every time–step

the time steps in the control dict are small → PaWan made them bigger for our run

- Various motions can be prescribed to the body:
  - (oscillating) linear motion
  - (oscillating) rotating motion
  - axis rotation motion
  - SDA
  - tabulated 6DoF
- ...here we leave the testing on the user

GENERAL INFO ABOUT OVERSET MESHING.

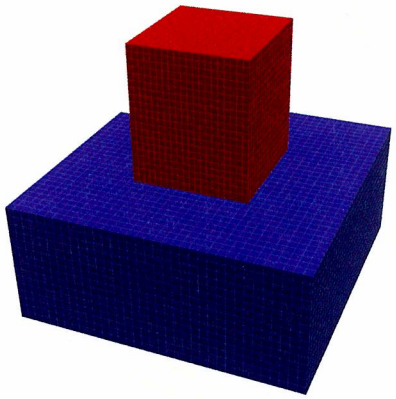
- cellToCellStencil → 4 methods: inverseDistance, leastSquares, ...
- a single matrix is formed : explicit or implicit
- helper mesh (voxels):
  - used between each pair of 2 meshes (in case there are more objects and meshes)
- DebugSwitches ← controlDict
  (
  oversetMesh 1;
)
- dimension of boundBox (ll) for voxel mesh is global → where we expect mesh overlapping (boundary interaction)
  ↳ adapt size of cells according to size of movement

3.3 Floating Object tutorial

Floating body tutorial – overset version

- There are several `floatingObject` tutorials in OpenFOAM
- Description of tutorial using 6DOF with mesh deformation is in the appendix A1; appendix A2 contains the same case but using nDOF case using Rigid Body Motion (RBM) for reference; To avoid any mesh deformation problems user may employ the overset method as shown here
- We will study the `floatingObject` tutorial with overset
- Tutorial is located in

  `$FOAM_TUTORIALS/multiphase/overInterDyMFoam/floatingBody`
- Directory `background` holds the whole case with the background mesh
- Directory `floatingObject` holds only the mesh description for the floating object
- The setup itself is very similar to the standard dynamic mesh setup
- additional `zoneID` indicating mesh zone (not a cell zone) is needed; `zoneID` should be set to `0` for the coarsest mesh. `zoneID` field an be created with a `topoSet` and `setFields` after the mesh generation



- New patch type `overset` should be used for the patches of the overset mesh; Boundary condition overset should be then used accordingly in the `0` directory
- New dictionary entry setting for `oversetInterpolation` method in the `fvSchemes` should be set:

```
oversetInterpolation
{
    method          inverseDistance;
}

oversetInterpolationRequired
{
    alpha.water;
}
```

- Usually the `oversetInterpolationRequired` can stay empty – meaning full implicit treatment of the field
- Sometimes it makes sense to enable explicit interpolation for the field specified in the list.

dynamicMeshDict file (1)

- The `overset` of `dynamicFvMesh` class is selected

```
17 motionSolverLibs    ("libsixDoFRigidBodyMotion.so");
18
19 dynamicFvMesh       dynamicOversetFvMesh;
20
21 dynamicOversetFvMeshCoeffs
22 {
23 }
24
25 solver          sixDoFRigidBodyMotion;
26
27 // 6DOF solver, settings very similar to rigid Body movement
```

- Any motion solver could be chosen, here it is 6-DOF solver (rigid body motion solver is the other option here)
- Notice that the `innerDistance` is set to much further distance than the mesh spans to make sure the overset mesh is not morphed

dynamicMeshDict file (2)

- The

```
sixDoFRigidBodyMotionCoeffs
{
    patches         (floatingObject);
    innerDistance   100.0;
    outerDistance   101.0;
    centreOfMass    (0.5 0.5 0.3);
    // Cuboid dimensions
    Lx              0.24;
    Ly              0.24;
    Lz              0.4;
    rhoSolid        700;
    mass            #calc "$rhoSolid*$Lx*$Ly*$Lz";
    momentOfInertia #codeStream
    {
        codeInclude
        #{
            #include "diagTensor.H"
        #};

        code
        #{
            scalar sqrLx = sqr($Lx);
            scalar sqrLy = sqr($Ly);
            scalar sqrLz = sqr($Lz);
            os  <<
                $mass
               *diagTensor(sqrLy + sqrLz, sqrLx + sqrLz, sqrLx + sqrLy)/12.0;
        #};
        };
        report          on;
        accelerationRelaxation 0.6;
        solver
        {
            type Newmark;
        }
        constraints
        {
            fixedLine
            {
                sixDoFRigidBodyMotionConstraint line;
                //centreOfRotation (0.5 0.45 0.1);
                direction (0 0 1);
            }
        }
    }
```

- Notice that the `innerDistance` is set to much further distance than the mesh spans to make sure the overset mesh is not morphed

BC setting

- The `pointDisplacemet` needs to be set accordingly
- Notice the `overset` type instead of `calculated` used with non-overset types of mesh treatment

```
dimensions      [0 1 0 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    #includeEtc "caseDicts/setConstraintTypes"

    stationaryWalls
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    atmosphere
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    floatingObject
    {
        type            calculated;
        value           uniform (0 0 0);
    }
    oversetPatch
    {
        patchType       overset;
        type            zeroGradient;
    }
    sides
    {
        patchType       overset;
        type            zeroGradient;
    }
}
```

Settings of the numerical schemes

- The point displacement solver must be specified in the `fvSolution`

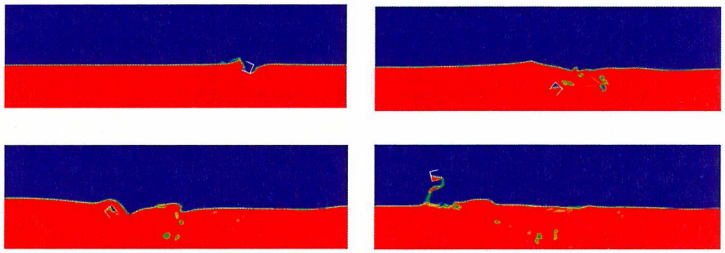
```
"cellDisplacement.*"
{
    solver          PCG; or GAMG, whatever we want to use
    preconditioner  DIC;

    tolerance       1e-06;
    relTol          0;
    maxIter         100;
}
```

- All continuous methods for wall distance calculation are supported (`Poisson`, `advectionDiffusion`); `meshWave` method is not supported

3.4 Multiphase tutorial

Overview

- The interaction with different phases in multiphase flow is always numerical challenge
- Following tutorial shows how to treat a case, where body diving into a water needs to be treated numerically
- The body has also prescribed type of motion with rotation along it's own axis while following a translation trajectory
- This demonstration case is not part of standard battery of tutorials distributed with OpenFOAM

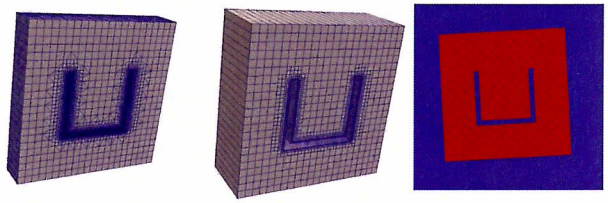


Mesh

- Mesh is created in a similar way as in the Cylinder tutorial
- First the `snappyHexMesh` is used to create the overset mesh around the cup in the directory `cup-snappyHexMesh`
- 3D mesh is in the next step extruded into a 2D mesh in the directory `cup-extruded`


```
22 constructFrom patch;
23 sourceCase "../cup-snappyHexMesh";
24 sourcePatches (xmin);
25
26 // If construct from patch: patch to use for back (can be same as
       sourcePatch)
27 exposedPatchName xmin;
28
29 // Flip surface normals before usage. Valid only for extrude from
       surface or
30 // patch.
31 flipNormals true;
32
33 //- Linear extrusion in point-normal direction
34 extrudeModel        linearNormal;
35 ...
```

- Further the overset mesh with the cup is merged into the case `backgroundMesh`



Meshing process

Setup

- Motion is set via `dynamicMeshDict` as a `multiMotion`
- Combination of two rotational motion functions is used

```
24 multiSolidBodyMotionSolverCoeffs
25 {
26     movingzone
27     {
28         solidBodyMotionFunction multiMotion;
29        //arc path
30         rotation1
31         {
32            solidBodyMotionFunction    rotatingMotion;
33            rotatingMotionCoeffs
34            {
35               origin       (0.00 0.00 4.375);
36               axis         (-1 0 0);
37               omega        0.4;//10
38            }
39         }
```

```
        //rotation around own axis
        rotation2
        {
            solidBodyMotionFunction    rotatingMotion;
            rotatingMotionCoeffs
            {
                origin      (0.00 3.42 1.85075);
                axis        (1 0 0);
                omega       1.5;//35
            }
        }
    }

}
```

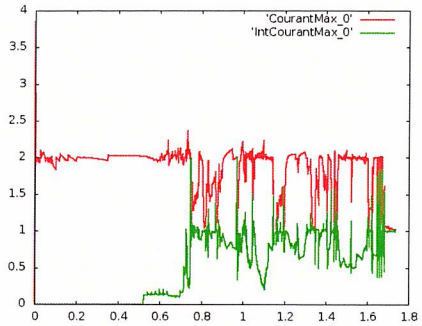
- Robust numerical setup is needed to avoid divergence in the moment of impact to the water surface
- Typically high velocity manifested by high maxCoAlpha and its consequences on presssure field is causing the convergence problems; therefore the maximum Courant numbers are kept strictly on a low values in `controlDict`:

```
maxCo           2.0;
maxAlphaCo      1.0;
```

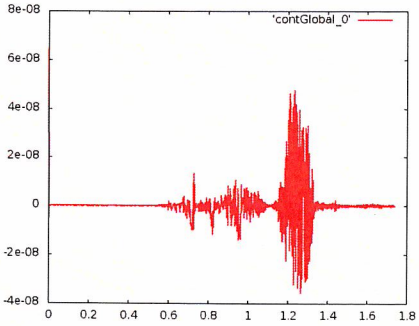
- Algebraic solvers are set with very tight tolerances on convergence `relTol` to 0.001
- MULES algorithm is setting 3 correctors
- PIMPLE setup reads 4 outer correctors and 2 PISO correctors together with 1 nonOrthogonal correctors the pressure is solved for 16 times each time steps
- Momentum equation is solved as well as `momentumPredictor` flag is set to `yes`
- Transport equations for turbulent scalars are solved every PIMPLE outer loop (`turbOnFinalIterOnly` is set to `no`)
- These settings are needed to keep the system stable with second order treatment in space (except for the turbulent scalars — see `fvSchemes` file)

Solver performance

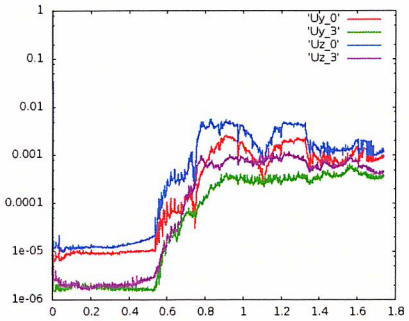
- Analysing log file from the solver can help us understand whether the above mentioned setup leads to successful solution
- First we try to identify the moment of impact with the water surface looking at maximum Courant numbers
- Clearly between 0.5 and 0.6 seconds the maximum Courant number is lowering as the parameter limiting the time step is suddently the interface Courant number

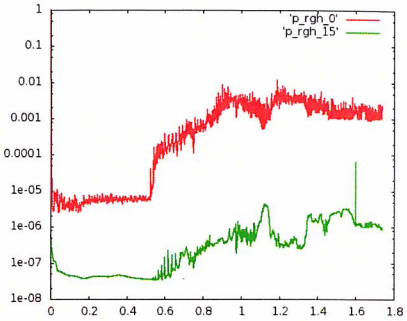


- Next what we should check are the conservation errors, here it will be the global error
- We can see the struggle of the solver when the body interacts with the water surface, but overall the continuity error is within reasonable bounds

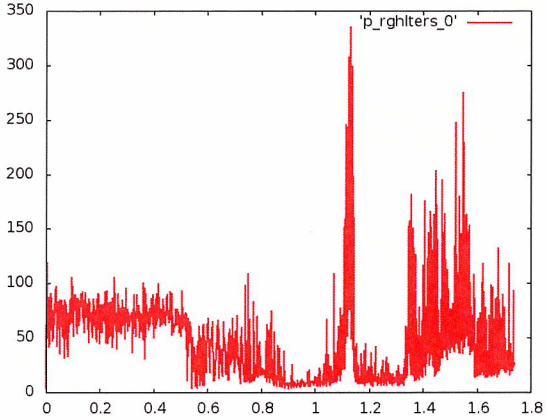


- Performance of the algebraic solvers and suitability of the number of correctors can be assessed looking at the residuals
- For each variable the initial residual at first pass and final pass are plotted





- From these graphs we can see the drop in residuals which advocates for the number of correctors
- Analogically we can see the number of iteration on pressure `p_rgh`



- Which shows rather high number of iterations
- Considering the speed of movement and interaction with the interface the solver is performing well and maintains stability, even at increased cost

3.5 Other examples

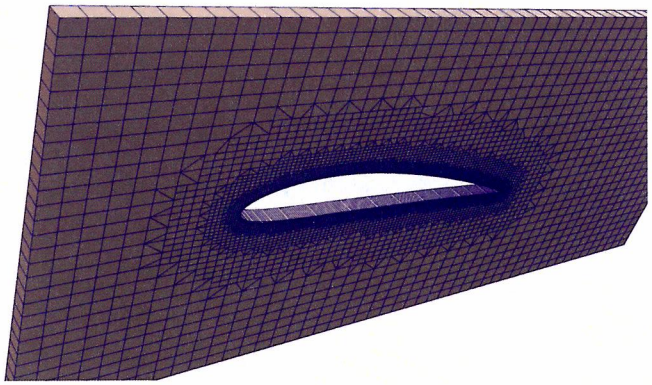
Various movements

- Tutorials for `overPimpleDyMFoam` `simpleRotor` and `twoSimpleRotors` are showing howto use overset together with solidBodyMovement allowing full rotation of the solid body without deforming the mesh

```
dynamicFvMesh       dynamicOversetFvMesh;

dynamicOversetFvMeshCoeffs
{
//    layerRelax 0.3;
}

solver          multiSolidBodyMotionSolver;

multiSolidBodyMotionSolverCoeffs
{
    movingZone1
    {
        solidBodyMotionFunction rotatingMotion;
        rotatingMotionCoeffs
        {
            origin      (0.005 0.005 0.005);
            axis        (0 0 1);
            omega       100.0;
        }
    }

    movingZone2
    {
        solidBodyMotionFunction rotatingMotion;
        rotatingMotionCoeffs
        {
            origin      (0.009 0.005 0.005);
            axis        (0 0 1);
            omega       -100.0;
        }
    }
}
```

- ...where `movingZone1` and `movingZone2` are `cellZones` defined in the `blockMeshDict`
- Tutorial `aeroFoil` for `overSimpleFoam` shows how to set the case for parametric study of various angles of attack
- The case has more complex structure, as the aerofoil is meshed using `snappyHexMesh` and then using utility `extrudeMesh` the mesh is flattened to a 2D mesh reading `extrudeMeshDict`:

```
constructFrom patch;
sourceCase "../aeroFoil_snappyHexMesh";
sourcePatches (symFront);

// If construct from patch: patch to use for back (can be same as sourcePatch)
exposedPatchName symBack;

// Flip surface normals before usage. Valid only for extrude from surface or
// patch.
flipNormals false;

//- Linear extrusion in point-normal direction
extrudeModel        linearNormal;

nLayers             1;

expansionRatio      1.0;

linearNormalCoeffs
{
    thickness       0.05;
}

// Do front and back need to be merged? Usually only makes sense for 360
// degree wedges.
mergeFaces false;   //true;

// Merge small edges. Fraction of bounding box.
mergeTol 0;
```



4 Closing remarks

Tips for running overset simulation

- The method is non-conservative and may lead to pressure fluctuations in closed domains. In incompressible flow user may employ flux adjustment in `fvSolution` file:

```
PIMPLE
    {
        ...
        oversetAdjustPhi    true;
        ...
    }
```

> This correction will work only when the overset boundary is closed

- When selecting the stencil, keep in mind the cellVolumeWeight method is first-order accurate, bounded and volume conservative. The inverseDistance method is up to second order accurate and bounded.
- For parallel runs, use `PCG` solver as GAMG is not yet supported (planned for next release)
- Also use `hierarchical` method of decomposition over the the scotch (faster and more robust solution compared to scotch) (for parallel runs)
- With steady state simulations with `overSimpleFoam` the relaxation factor for velocity might be reduced even for `consistent` formulation of `SIMPLE` formulation *e.g.*

```
...
equations
{
    U               0.5;
}
...
```

- The first patch (ID 0) should be overset patch (even with zero size); This is needed to trigger the overset interpolation (preferred, but not necessary)
- When running transient cases use `Euler` scheme with a small time step as `backwards` and `Crank-Nicolson` schemes may give non-physical hotspots with moving meshes; They perform well when the mesh is not moving

  → move mesh slowly (with Euler)

- Cell size close to the overset patch should be of the same size to minimise interpolation errors
- Time step must be small enough to accommodate for a sequential change of `cellType` from blocked to interpolated and then to calculated; This leads to a mesh motion Courant number which must be kept below one, for second order discretisation smaller then 0.5
- In the overset region there should be at least 4+ cells between the body patch and the overset patch; cells next to one of the patch are blocking the flow and cells next to overset patch are used to interpolate the fluxes.

Stay tuned

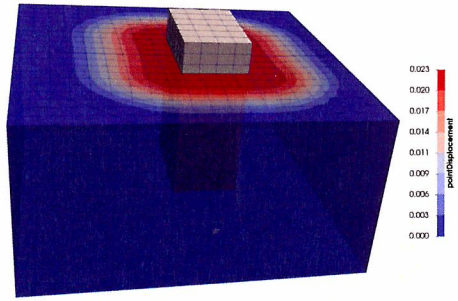
- Stay tuned for the latest development
- Check our twitter and release webinars for the latest news on the development
- Register at our GitLab web `develop.openfoam.com` for free; you can:
  - Get the latest bug–fixes
  - Report bugs you find yourself
  - See the fixes introduced, directly in the source code

A Appendix

A.1 Mesh motion imposed by flow tutorial – 6-DOF

6-DOF - Overview

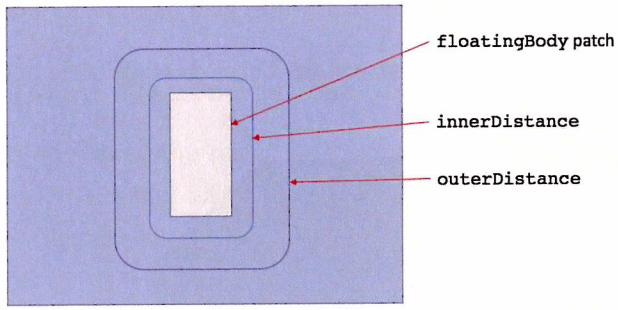
- 6-ODF allows mesh solid motion prescribed by mass (and its centre), moment of inertia and various constraints and restraints
- In version 2.3 of OpenFOAM 6-DOF was introduced as a solver `sixDoFRigidBodyMotion` to a `dynamicMotionSolverFvMesh` mesh treatment class
- 6-DOF method applies explicit correction of the motion to the solid body represented by a wall patch
- 6-DOF on various bodies can be applied via BC specification in `pointDisplacement`
- Both approaches will be shown here on a floating body example



6DOF tutorial - floating object

- `floatingObject` tutorial is located in `$FOAM_TUTORIALS/multiphase/interDyMFoam/RAS/`
- Class `dynamicFvMesh` is set to `dynamicMotionSolverFvMesh`
- We have to register the precompiled dynamic library `libsixDoFRigidBodyMotion.so` to be able to use `motionSolver sixDoFRigidBodyMotion` as shown

```
17 dynamicFvMesh       dynamicMotionSolverFvMesh;
18
19 motionSolverLibs    ("libsixDoFRigidBodyMotion.so");
20
21 motionSolver        sixDoFRigidBodyMotion;
```



- Parameters to the solver `sixDoFRigidBodyMotionCoeffs` may contain:

```
23 patches          (floatingObject);
24 innerDistance    0.05;
25 outerDistance    0.35;
26
27 centreOfMass     (0.5 0.45 0.35);
28
29 // Cuboid dimensions
30 Lx               0.3;
31 Ly               0.2;
32 Lz               0.5;
33
34 // Density of the solid
35 rhoSolid         500;
36
37 // Cuboid mass calculated from expression
38 mass             #calc "$rhoSolid*$Lx*$Ly*$Lz";
39
40 // Cuboid moment of inertia about the centre of mass
41 momentOfInertia #codeStream
42 {
43     codeInclude
44     #{
45         #include "diagTensor.H"
46     #};
47
48     code
49     #{
50         scalar sqrLx = sqr($Lx);
51         scalar sqrLy = sqr($Ly);
52         scalar sqrLz = sqr($Lz);
53         os  <<
54             $mass
55            *diagTensor(sqrLy + sqrLz, sqrLx + sqrLz, sqrLx + sqrLy)/12.0;
56     #};
57 };
58 ...
```

- Most important settings is:

patches the list of patches to apply fluid forces to

mass of the object in question

centreOfMass initial centre of mass in GCS

momentOfIntertia is a tensor determining the torque needed for a desired angular acceleration (also known as rotational inertia)

- For cylinder rotating around the central axis, the moment of Inertia can be expressed as:

$$I = \frac{1}{2} M R^2 \quad (1)$$

- For various other objects you may find expressions in the literature
- By definition it is:

$$I = \int_Q r^2 dm \quad (2)$$

Constraints to the movement

- Constrains are defined to avoid movement in particular direction or anchor the object

axis – allows only rotation around a fixed axis

line – allows translation along a line [reads: `centreOfRotation,direction`]

orientation – object is fixed in global CS

plane – allows movement on a plane [reads: `centreOfRotation, normal`]

point – fixes a point (centre of rotation) in global CS

Restraints to the movement (1)

- Restraints are defined to damp the acceleration of the body

linearAxialAngularSpring – reads following parameters:

refQ – Reference orientation where there is no moment
axis – Global axis around which the motion is sprung
stiffness – Spring stiffness coefficient (Nm/rad)
damping – Damping coefficient (Nms/rad)

linearDamper – reads Damping coefficient (Ns/m)

linearSpring – reads following parameters:

anchor – Anchor point, where the spring is attached to an immovable object
refAttachmentPt – Reference point of attachment to the solid body
stiffness – Spring stiffness coefficient (N/m)
damping – Damping coefficient (Ns/m)
restLength – Length of spring when no forces are applied to it

Restraints to the movement (2)

- Restraints are defined to damp the acceleration of the body

sphericalAngularDumper – reads Damping coefficient (Nms/rad)

sphericalAngularSpring – reads following parameters:

refQ – Reference orientation where there is no moment
stiffness – Spring stiffness coefficient (Nm/rad)
damping – Damping coefficient

tabulatedAxialAngularSpring – Axial angular spring with moment values drawn from an interpolation table. Linear damping.

refQ – Reference orientation where there is no moment
axis – Global axis around which the motion is sprung
moment – table of scalar values
convertToDegrees – boolean
damping – Damping coefficient (Nms/rad)

Sudden changes to the geometry could bring large deformations or sources to the system resulting in difficulties in finding solution. User should watch mesh quality and convergence of the system throughout the simulations (*e.g.* by plotting residuals & mass conservation errors)

Point Displacement

- File `pointDisplacement` contains settings for all patches for mesh points displacement
- When the 6-DOF setup is done in `dynamicMeshDict`, the file contains only BC setup:

```
dimensions      [0 1 0 0 0 0 0]; // [m]

internalField   uniform (0 0 0);

boundaryField
{
    stationaryWalls     // external patches
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    atmosphere          // fixed opening
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    floatingObject      // moving object
    {
        type            calculated;
        value           uniform (0 0 0); //initialisation only
    }
}
```

Output and practical tips

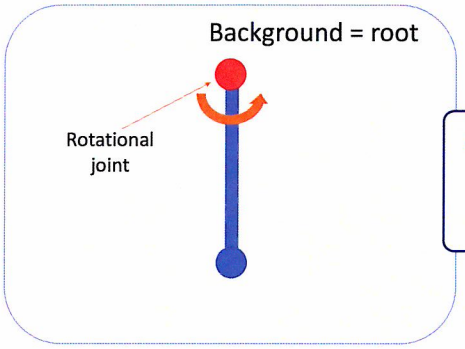
- Should the constraints and restraints defined not suffice, user can use `rigidBodyMotion` (shown later)
- Output from the solver contains CFL numbers (in multiphase also for the interface)
- Output contains also information about the centreOfRotation position and angular and linear momentums if switch `report` in `dynamicMeshDict` is on
- For a very sudden mesh movements smaller time step must be chosen and the `accelerationRelaxation` should be set higher
- In severe cases `accelerationDamping` should be employed
- The simulation will typically fail with too severe mesh deformation due to the body movement

- In this case the solution would be to employ `overset mesh treatment` where relative movement of the body to the background static mesh has no relevance

A.2 Mesh motion imposed by flow tutorial – RBM

Rigid body motion solver – overview

- Instead of six degrees of freedom we may use more general framework of rigid body motion
- Main principle of this approach is to define one or number of bodies and kinematic joints in between them
- Simple scheme shows pendulum, where rotational joint is defined with respect to the background "body" and fixed joint between the `rod` and `bob` bodies



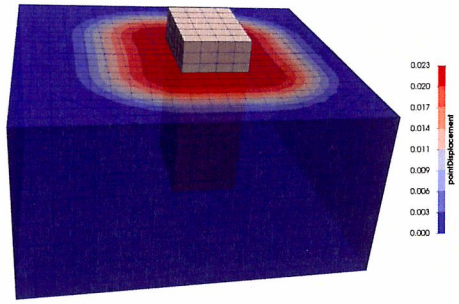
Pendulum example is used in a test provided to test the kinematics of the bodies and their joints. For details see: `$FOAM_APP/test/rigidBodyDynamics/`

Rigid body was implemented following Featherstone, R. (2008). Rigid body dynamics algorithms. Springer

Rigid body motion example

- Let's inspect the settings on the `floatingObject` tutorial:

```
>> cd $FOAM_TUTORIALS/multiphase/interDyMFoam/RAS/floatingObject
```

- The dynamic motion is set in `constant/dynamicMeshDict`
- and in the file `0/pointDisplacement`

Rigid body motion example — dynamicMeshDict (1)

- Solver for `dynamicMotionSolverFvMesh` is `rigidBodyMotion`

```
17 dynamicFvMesh        dynamicMotionSolverFvMesh;
18
19 motionSolverLibs    ("librigidBodyMeshMotion.so");
20
21 solver              rigidBodyMotion;
```

Rigid body motion example — dynamicMeshDict (2)

- Body itself is set in the same way as in 6-DOF
- `parent` body to the `floatingObject` is `root` (the static background)

```
32 bodies
33 {
34     floatingObject
35     {
36         type            cuboid; // sphere, masslessBody, compositeBody
37         parent          root;   // name of the body connected to
38
39         // Cuboid dimensions
40         Lx              0.3;
41         Ly              0.2;
42         Lz              0.5;
43         rho             500;
44         mass            #calc "$rho*$Lx*$Ly*$Lz";
45         L               ($Lx $Ly $Lz);
46         centreOfMass    (0 0 0.25);
47         transform       (1 0 0 0 1 0 0 0 1) (0.5 0.45 0.1);
48         joint
```

```
        {
            type            composite;
            joints
            (
                {
                    type Py;        //translation in y-direction
                }
                {
                    type Ry;        //rotation in y-direction
                }
            );
        }

        patches         (floatingObject);
        innerDistance   0.05;
        outerDistance   0.35;
    }
}
}
```

- The rigid body needs a specification of the inertia
- For that purpose the body can be specified as `cuboid`, `sphere`, `masslessBody` or `compositeBody` (if defined by several bodies)

The source code (therefore the constructors with the List of Private data declaration) can be found in `$FOAM_SOURCE/rigidBodyDynamics/bodies` directory

- Size and mass can be given or calculated via the `#calc` expressions
- Initial position of the object must be specified with a transformational tensor `transform` followed by the initial position

The tensor components are describing cosines of rotations in the appropriate directions. Details can be found *e.g.* on `https://en.wikipedia.org/wiki/Cartesian_tensor` In short, tensor with main diagonal filled with number 1 specifies no rotation against the global CS

Rigid body Joints

- joints are connection points between the rigid body with a prescribed degree of freedom
- Joints can be translational or rotational in a prescribed direction

- For example `Px` specifies prismatic joint for translation along the x-axis
- `Pa` specifies prismatic joint for translation along the specified arbitrary axis.
- `Rxyz` specifies spherical joint for rotation about the x/y/z-axes using Euler-angles in the order x, y, z
- Details can be found only in the source code and Doxygen
- In Doxygen under the Mesh motion/Rigid body dynamics/Joints/... section
- Each body is connected to other body via keyword `parent`
- Bo0dy can be connected to the background with a specific parent name `root`
- Joints in RBM do apply only kinematic restrain to the movement
- It is not possible to use them to impose an extra force

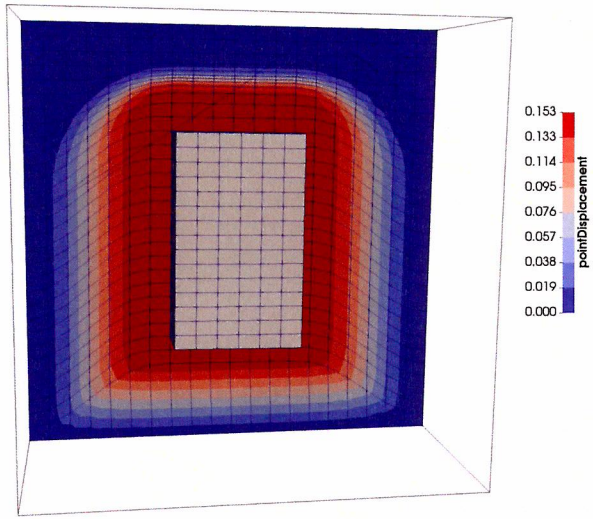
pointDisplacement file

```
dimensions      [0 1 0 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    stationaryWalls
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    atmosphere
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    floatingObject
    {
        type            calculated;
        value           uniform (0 0 0);
    }
}
```

- Patch on the rigid body is set to calculated as the displacement is not known
- Similar setup of the case could be done with 6DOF solver

- When the mesh deformation is too large, alternative approach must be used

- When the mesh deformation affects the mesh quality, different approach to mesh movement must be adopted: overset mesh introduced in version v1706 is an obvious choice



Too large distortion

- Problem with 6DOF or RBM solutions is, we cannot run the `moveDynamicMesh` before the solver run itself, as the movement is imposed by the fields themselves
- We will use the `floatingObject` tutorial to see the unsuccessful run
- Let us make the inertia of the object much lighter by setting density from 500 to 200$^{kg}/_{m^3}$ in `constant/dynamicMeshDict`

- This change is blueprinted into the whole case, as the tutorial uses dynamic code to set the inertia tensor
- When we run the case and write results every 0.01s we may see the mesh distortion before the computation crashes
- In the log file you can see the non-convergence of pressure if you look at the continuity error
- In the last time-step the MULES algorithm fails completely to maintain the boundedness of the scheme

Log output

- Log output from the solver shows:

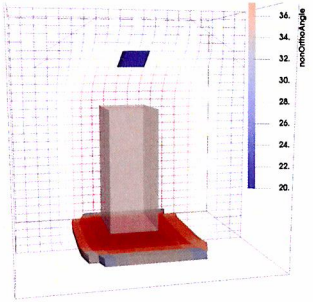
```
Rigid-body motion of the floatingObject
    Centre of rotation: (0.5 0.527828111118 0.1)
    Orientation: (0.999975336987 0 0.00702320571491 0 1 0 -0.00702320571491 0 0.999975336987)
//=>    Linear velocity: (0 48.3264117829 0)
//=>    Angular velocity: (0 -96.1993387859 0)
Execution time for mesh.update() = 0.04 s
GAMGPCG:  Solving for pcorr, Initial residual = 1, Final residual = 9.17203055925e-07, No Iterations 4
time step continuity errors : sum local = 3.53153820566e-09, global = -3.61767855026e-10, cumulative = -0.00193837401686
smoothSolver:  Solving for alpha.water, Initial residual = 7.59203649521e-05, Final residual = 6.7797658773e-10, No
       Iterations 2
Phase-1 volume fraction = 0.531100255824  Min(alpha.water) = -0.0112760779966  Max(alpha.water) = 1.00322236454
Applying the previous iteration compression flux
MULES: Correcting alpha.water
MULES: Correcting alpha.water
MULES: Correcting alpha.water
MULES: Correcting alpha.water
Phase-1 volume fraction = 0.531100255873  Min(alpha.water) = -0.0111584212163  Max(alpha.water) = 1.00322236454
//=> GAMG:  Solving for p_rgh, Initial residual = 0.612132840246, Final residual = 0.00271256749181, No Iterations 2
time step continuity errors : sum local = 5.65745177855e-05, global = 1.98729219574e-05, cumulative = -0.0019185010949
PIMPLE: iteration 2
forces forces:
    Not including porosity effects
    ...
```

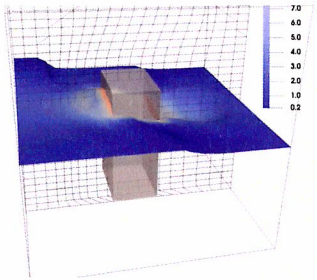
- Log output from the solver shows:

```
...
Rigid-body motion of the floatingObject
//=>    Centre of rotation: (0.5 -133989.47426 0.1)
    Orientation: (0.527071758457 0 -0.849820781951 0 1 0 0.849820781951 0 0.527071758457)
//=>    Linear velocity: (0 -90467635458.5 0)
//=>    Angular velocity: (0 95108668914.4 0)
Execution time for mesh.update() = 0.02 s
GAMGPCG:  Solving for pcorr, Initial residual = 1, Final residual = 0.0246735016701, No Iterations 101
time step continuity errors : sum local = 102.042177942, global = -0.871253258535, cumulative = -0.873171759629
smoothSolver:  Solving for alpha.water, Initial residual = 0.958645535709, Final residual = 1.78353734651e-09, No
       Iterations 22
//-> Phase-1 volume fraction = 0.226423975512  Min(alpha.water) = -16110.8223433  Max(alpha.water) = 39.8120014987
Applying the previous iteration compression flux
MULES: Correcting alpha.water
MULES: Correcting alpha.water
MULES: Correcting alpha.water
MULES: Correcting alpha.water
Phase-1 volume fraction = 55964655017.2  Min(alpha.water) = -1.94217901511e+31  Max(alpha.water) = 1.94075577712e+31
```

Mesh distortion

- Picture on the left shows the mesh distortion below the object
- Cells with non-orthogonality higher then 20 degrees
- On the right hand side the water surface and high velocity reaching up to $19^{m}/_{s}$ (we could see also the high Courant number in the log from the simulation)
- Remedy could be refine the mesh or extend the domain, relax the acceleration and lower the time step; If the distortion is too large for mesh to handle the only option is to use overset mesh introduced in the next section





A.3 Creating Zones and Sets

Working with Zones and Sets

- Cells and faces can be organised into sets or zones
- There are various reasons:
  - Specify the region to apply source term via `fvOptions` (*e.g.* porousity region)or `MRFProperties`
  - Creating a region to post-process the data via function object
  - Define a section to create solid and fluid regions with `splitMeshRegions` utility for CHT simulation
  - To specify part of the mesh for other mesh operations (splitting, extracting, patch creation)

- Set is collection of labels; Is destroyed by mesh renumbering
- Zone is indirect list which is fixed to the given zone
- Set or zone can be created
  - Interactively using `setSet` CLI utility
  - Using utility `topoSet` typically on structured meshes
  - Specifying the zone or set in `snappyHexMeshDict` for arbitrary shapes
  - Importing the mesh via *e.g.* fluent format, which can translate also sets

Interactive set creation

- Interactive manipulation with a mesh is possible via `setSet`
- Utility provides its own CLI but can be run in batch mode as well
- Utility expects to be run in `case` directory to be able to read the mesh

- Commands has the following structure

```
cellSet|faceSet|pointSet <setName> <action> <source>
```

- ... where the action could be one of:

list – prints the contents of the set

clear – clears the set

invert – inverts the set

remove – remove the set

new <source> – sets to set to the source set

add <source> – adds all elements from the source set

delete <source> – deletes

subset <source> – combines current set with the source set

- CellZones can be created via

```
cellZoneSet c0Zone new setToCellZone c0
```

- `setSet` utility can be used to create a faceSet:

```
faceSet amiFaces new patchToFace "AMI.*"
```

- which has an equivalent in `topoSetDict` for `topoSet`

```
17 actions
18 (
19     {
20         name    amiFaces;
21         type    faceSet;
22         action  new;
23         source  patchToFace;
24         sourceInfo
25         {
26             name "AMI.*";
27         }
28     }
29 );
```


Tip for parallel run

- Collecting AMI to a faceSet can be usefull for example to keep the AMI on a single processor during the parallel run
- New faceSet can be used in `decomposeParDict` to preserve region with AMI in a single domain:

```
singleProcessorFaceSets
    (
        //- Keep all of faceSet on a single processor. This puts all cells
        //  connected with a point, edge or face on the same processor.
        //  (just having face connected cells might not guarantee a balanced
        //  decomposition)
        // The processor can be -1 (the decompositionMethod chooses the
        // processor for a good load balance) or explicitly provided (upsets
        // balance)
        type    singleProcessorFaceSets;
        singleProcessorFaceSets ((amiFaces -1));
```

Source definition

- Sources to `setSet` or `topoSet` utilities can be listed in inheritance diagram in Doxygen by searching for keyword `topoSetSource`
- ... or found in `$FOAM_SRC/meshTools/sets`
- To understand how to use these sources, search for the Private data declaration in an `*.H` file, or Constructor definition in `*.C` file

```
54 // Private data
55
56     //- Add usage string
57     static addToUsageTable usage_;
58
59     //- First point on cylinder axis
60     vector p1_;
61
62     //- Second point on cylinder axis
63     vector p2_;
64
65     //- Radius
66     scalar radius_;
```

- `cylinderToCell` source can be used in the `setSet` as

```
Command>cellSet c0 new cylinderToCell (0 0 0) (1 1 1) 0.1
    Set:c0  Size:0  Action:new
    Adding cells with centre within cylinder, with p1 = (0 0 0),
    p2 = (1 1 1) and radius = 0.1
    Writing c0 (size 7381) to "constant/polyMesh/sets/c0" and to
    vtk file "VTK/c0/c0_0.vtk"
```

Utility `topoSet`

- This mesh manipulation utility has similar capabilities as `setSet`
- Utility reads the `system/topoSetDict` file
- To understand the options and various source definitions, the `topoSetDict` is prepared in `$FOAM_UTILITIES/mesh/manipulation/topoSet` utility source code directory

Sets in `snappyHexMesh`

- The disadvantage of sets defined by `topoSet` or `setSet` is the dependence on the mesh lines; typically these utilities are used on structured mesh
- The mesh should conform the arbitrary shape of the zone or set
- Mesh generator `snappyHexMesh` can create the mesh aware of the set or zone
- To specify the region either STL surface can be used (or VTK, OBJ ...) or other geometrical object defined in the `geometry` section of `snappyHexMeshDict`