Using OpenFOAM-1.6-ext's 'DynamicTopoFvMesh and Mesquite Motion Solver' libraries to solve prescribed boundary motion problems.

Richard Kenny, CAE Solutions.

Tools: 1. *OpenFOAM 1.6-ext* (supported by Wikki Ltd.)

2. "DynamicTopoFvMesh and Mesquite Motion Solver" libraries (supported by University of Massachusetts, www.ecs.umass.edu/~smenon/)

Where to find *OpenFOAM 1.6-ext?*

1) Use the control version software ‘git’

a) Create a local clone (or ‘copy’) of the distribution

git clone git://openfoam-extend.git.sourceforge.net/gitroot/openfoam-extend/OpenFOAM-1.6-ext

b) Update distribution with:

foam; git pull

OR,

2) Use FOCUSスパコン (www.j-focus.or.jp)

a) Coupled ‘mesh motion’ and CFD calculations are ‘heavy’ so, run in parallel using FOCUS’s considerable resources.

b) *OpenFOAM 1.6-ext :* installed 2011/08 by CAE Solutions.

Available at: */home1/share/OpenFOAM/OpenFOAM-1.6-ext*

"DynamicTopoFvMesh and Mesquite Motion Solver" libraries.

- The "Mesquite mesh motion library" was originally developed at Sandia Laboratories cf.

  http://www.cs.sandia.gov/optimization/knupp/

  and contains optimization algorithms (based upon minimization of specified objective functions subject to certain constraints) for smooth mesh motion. Mesh conditioning checks are performed each time-step.

- The implementation of the above in *OpenFOAM*(1.6-ext) was undertaken at the University of Massachusetts cf.

  http://www.ecs.umass.edu/~smenon

  which also maintains a 'git' distribution at

  https://github.com/smenon/dynamicTopoFvMesh

  The result is the 'mesh motion' class

  mesquiteMotionSolver

  and the 'topological' changer class called

  DynamicTopoFvMesh

- As mesh points move an iterative procedure involving *Delaunay Triangulation* and mesh cell insertion/deletion is employed to generate a well-conditioned final mesh. Works in both 2D and 3D with parallel capability.

- Unlike other mesh-motion approaches available in *OF* the *mesquiteMotionSolver* and related libraries allow an arbitrary range of motion.

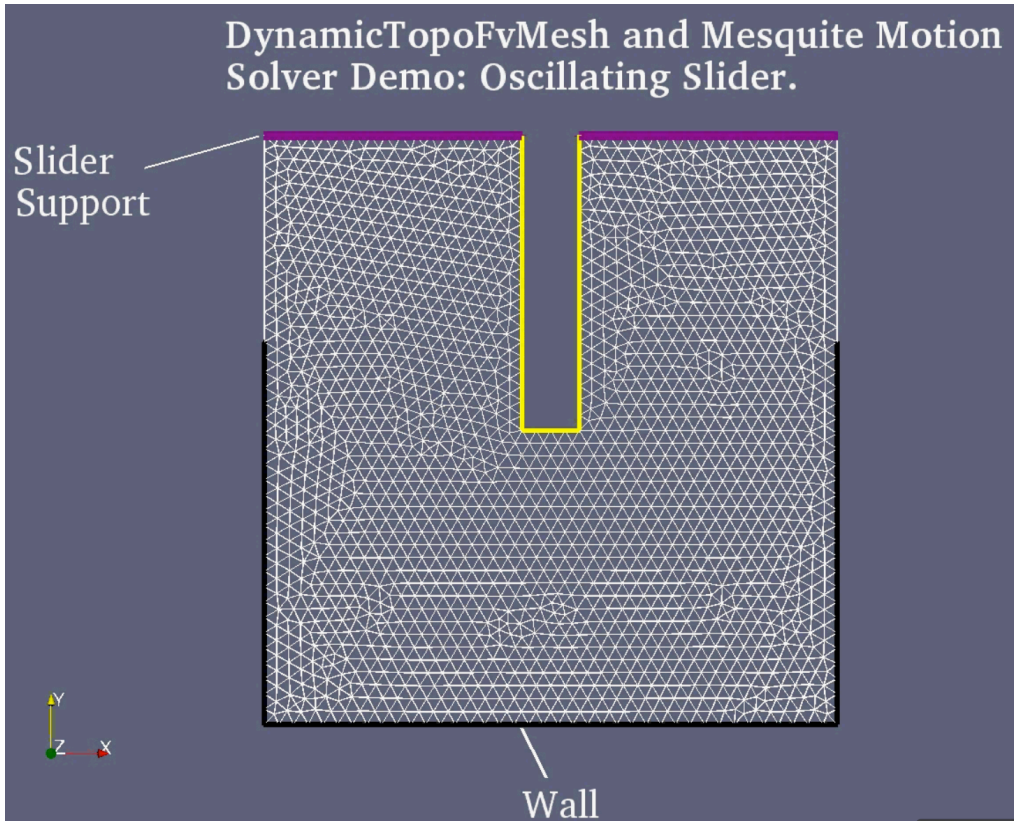
Simple 2D Case: Slider Motions

Prescribed Slider (yellow patch) Motions:

- oscillatory *translational*.
- oscillatory *rotational*
- oscillatory *translation/rotational*

Mesh Requirements

- 2D *OF* 'prism mesh' which is one mesh cell thick (conveniently constructed using the open source mesher tool Gmsh cf.
  http://geuz.org/gmsh/
  and
  http://openfoamwiki.net/index.php/2D_Mesh_Tutorial_using_GMSH).



Dictionary *constant/dynamicMeshDict* Settings:

Some principal entries:

```
dynamicFvMeshLibs      ("libdynamicTopoFvMesh.so"); //Load the dynamicTopoFvMesh library
 mesquiteOptions
{
   //- Constrain surface mesh-motion on a specified cylinder (cf. tutorial circCylinder3d).
   // - Adapt to current case geometry – requires some coding in the 'Mesquite Motion Solver' library.
   cylindricalConstraints
   {
      ..
    }

   //- Specify fixedValue patches for the motionSolver
   fixedValuePatches
   {
      slider
      {
         type        simpleHarmonicDisplacement;// PointPatch Boundary type that prescribes the motion.
         ..                                     // Functions or tabulated (x,y,z,t) data may be supplied.
      }
   }
}
```

Dictionary *constant/dynamicMeshDict* Settings (continued):

```
//- Some options for dynamicTopoFvMesh
dynamicTopoFvMesh
{
   // Toggle edgeRefinement on/off
   edgeRefinement      yes;// Helps maintain well-conditioned mesh on boundaries
   refinementOptions
   {
      collapseRatio   0.5;// criteria for cell collapse
      bisectionRatio  1.5;// criterion for cell insertion
      growthFactor    1.0;// Expansion factor from edge to interior

      //- By default, existing boundary edge-lengths are used for length-scales.
      //- Fix lengthscale on patches requiring special (customized) intervention.
      fixedLengthScalePatches
      {
       slider       0.00028;
       sliderSupport 0.00028;
      }
..
}
```

Sample Log output: Mesh checks are performed every time-step.

Point usage OK.

Upper triangular ordering OK.

Topological cell zip-up check OK.

Face vertices OK.

Face-face connectivity OK.

Mesh topology OK.

Boundary openness (0 5.32833e-19 1.41909e-15) Threshold = 1e-06 OK.

Max cell openness = 2.02687e-16 OK.

Max aspect ratio = 5.32057 OK.

Minumum face area = 7.02522e-09. Maximum face area = 1.15972e-07. Face area magnitudes OK.

Min volume = 5.60545e-13. Max volume = 5.79861e-12. Total volume = 4.75e-09. Cell volumes OK.

Mesh non-orthogonality Max: 45.9194 average: 12.5429 Threshold = 70

Non-orthogonality check OK.

Face pyramids OK.

Max skewness = 0.693341 OK.

Mesh geometry OK.

Mesh OK.

N.B. 'mesh motion Courant Number' < 0.5 for stable evolution.

Observations:

1. The ‘mesquiteMotionSolver’ and ‘dynamicTopoFvMesh’ classes produce well-conditioned (2D/ 3D tet) meshes for *straightforward* (and prescribed) boundary motions of arbitrary amplitude. (See movies for this case and ‘slider’ demos). Extension to hex meshes is currently under development.

2. Motions producing *unwanted* surface distortion require ‘user intervention’ by means of a specially-constructed boundary class. This may not be trivial.

3. For successful Delaunay triangulation, mesh points should move less than the average cell width over the domain (*mesh Courant Number < 0.5)*. This condition needs to be combined with the CFD Courant number criterion.

4. In theory, moving surfaces may be specified using *stl* format. No demos are known for this case and the practicalities currently remain unknown too.

5. CFD with mesh motion is ‘heavy’. Fortunately, both the ‘mesquiteMotionSolver’ and ‘dynamicTopoFvMesh’ classes are ‘parallel aware’. But, ‘parallel computations’ require any user-customized boundary-constraints to be similarly ‘parallel-aware’!

6. With *interFoam,* improve mass conservation error by sub-cycling over *alpha* equation.