# icoFsiFoam and interFsiFoam

**Constructing solvers for weakly coupled problems**

**using OpenFOAM-1.5-dev**

## Fluid–structure interaction

Fluid–structure interaction (FSI): A multiphysics problem.
Level of dynamic coupling is important.

- Weakly coupled
  - Small deformations
  - Permits partitioned approach
- Strong dynamical coupling
  - highly flexible structures
  - Similar densities between fluid and solid if solid stiffness is low
  - Demands a coupled approach

## FSI: mesh

Mesh construction is important

- Need dynamic mesh due to solid deformation
- Cell partitioning at the FSI boundary should give similar sizes of adjacent volumes
- Finite volume (FV) meshes in OpenFOAM are cell–centered
  This causes a problem at the FSI boundary: less accurate mesh motion predictions
- Vertex based meshes are optimal for FSI problems

## icoFsiFoam

Partitioned transient FSI solver for incompressible flow interacting with a solid of linear elasticity, causing small deformations in the solid.
Performs the partitioned solver–loop:

- Pressure is set on the FSI boundary
- Traction on the solid boundary is updated
- Solid deformation is solved using *stressedFoam* algorithm
- Dynamic mesh is updated accordingly
- Fluid domain is solved with a SIMPLE loop using additional pressure correction loops

The *icoFsiFoam* solver uses finite element mesh at the traction boundary between solid and fluid for the boundary motion, then interpolates between the domains.
Accepts different mesh sizes between solid and fluid domains - but too large difference will cause convergence problems!

## myIcoFsiFoam

Constructing your own FSI solver

- Create directory and copy files
- I chose the `$FOAM_RUN` directory, but you may pick a different one if you like.

```
run
mkdir myIcoFsiFoam
cp $FOAM_SOLVERS/incompressible/icoFoam/icoFoam.C myIcoFsiFoam/myIcoFsiFoam.C
cp $FOAM_SOLVERS/incompressible/icoFoam/icoFoam.C myIcoFsiFoam/solveFluid.H
cp $FOAM_SOLVERS/incompressible/icoFoam/createFields.H .
cp $FOAM_SOLVERS/stressAnalysis/stressedFoam/stressedFoam.C solveSolid.H
cp $FOAM_SOLVERS/stressAnalysis/stressedFoam/createFields.H createStressFields.H
cp $FOAM_SOLVERS/stressAnalysis/stressedFoam/read* .
cp $FOAM_SOLVERS/stressAnalysis/stressedFoam/calculateStress.H .
cp -r $FOAM_SOLVERS/stressAnalysis/stressedFoam/Make .
cp -r $FOAM_SOLVERS/stressAnalysis/icoFsiFoam/tractionDisplacement .
cp $FOAM_SOLVERS/stressAnalysis/icoFsiFoam/setPressure.H .
cp $FOAM_SOLVERS/stressAnalysis/icoFsiFoam/setMotion.H .
cp $FOAM_SOLVERS/stressAnalysis/icoFsiFoam/readCouplingProperties.H .
cp $FOAM_SRC/OpenFOAM/include/createMesh.H createStressMesh.H
```

## myIcoFsiFoam.C

Use your editor of choice to

- Remove main solver loop between line
  `#include CourantNo.H`
  and the `runTime.write()`-line.

- Add the following includes (from `icoFsiFoam.C`) to the header

```
#include "dynamicFvMesh.H"
#include "tractionDisplacementFvPatchVectorField.H"
#include "patchToPatchInterpolation.H"
#include "tetFemMatrices.H"
#include "faceTetPolyPatch.H"
#include "tetPolyPatchInterpolation.H"
#include "fixedValueTetPolyPatchFields.H"
#include "pointFields.H"
#include "volPointInterpolation.H"
```

## myIcoFsiFoam.C

- Rename `createMesh.H` to `createDynamicFvMesh.H` and add the following includes to the `main`–function before the time loop

```
#   include "createStressMesh.H"
#   include "readMechanicalProperties.H"
#   include "readThermalProperties.H"
#   include "createStressFields.H"
#   include "readCouplingProperties.H"
#   include "readTimeControls.H"
```

- Add the following includes between the time outputs after `CourantNo.H`:

```
#       include "readTimeControls.H"
#       include "setDeltaT.H"
//    Main solver code:
#       include "setPressure.H"
#       include "solveSolid.H"
#       include "setMotion.H"
#       include "solveFluid.H"
```

## solveFluid.H - createFields.H

`solveFluid.H`:

- Remove all header- and footer-information, retain only the main solver loop interior.
  Ie. what was stripped from the `myIcoFsiFoam.C` file.
- Change `continuityErrs.H` to `movingMeshContinuityErrs.H`.
- Remember to put braces {} around the code to avoid namespace confusion in compiler.

`createFields.H`:

- Insert

```
dimensionedScalar rhoFluid
(
    transportProperties.lookup("rho")
);
```

  between reading of `nu` and `p`

## solveSolid.H - createStressFields.H

`solveSolid.H`:

- As above, remove all header- and footer-information and retain only the main solver loop interior including and between `#include readStressedFoamControls.H` and `#include calculateStress.H`
- Change all instances of `U` to `Usolid`.
- Change all instances of `T` to `Tsolid`.
- Remember to put braces {} around the code to avoid namespace confusion in compiler.

`createStressFields.H`:

- Replace every instance of `mesh` with `stressMesh`
- Define the volume vector field `Usolid` instead of `U`
- Define the volume scalar field `Tsolid` instead of `T`

## readMechanicalProperties.H - readThermalProperties.H - calculateStress.H

`readMechanicalProperties.H`:

- Replace `mesh` with `stressMesh` to specify the solid mesh.
- Replace every instance of `nu` with `nuS` to separate it from the fluid `nu`.

`readThermalProperties.H`:

- On line 9, replace `mesh` with `stressMesh` to specify solid domain.

`calculateStress.H`:

- Replace all instances of `U` and `T` with `Usolid` and `Tsolid`, respectively.
- Replace `mesh` with `stressMesh` in the stress meshes since we are in the solid domain.

## createStressMesh.H - readStressedFoamControls.H

`createStressMesh.H`:

- Replace `mesh` with `stressMesh` on line 4 to separate the solid mesh from the fluid mesh.
- Replace `Foam::fvMesh::defaultRegion,` with `"solid",` to specify a different region than the default, since the default region will be used for the fluid domain.
- Append the following lines to create an interpolator between the meshes:

```
Foam::pointMesh pStressMesh(stressMesh);

Foam::volPointInterpolation cpi
(
    stressMesh,
    pStressMesh
);
```

`readStressedFoamControls.H`:

- Change entry `mesh` to `stressMesh` to read from correct mesh (solid mesh).

## Make directory files

`Make/files`:

- Change all instances of `stressedFoam` to `myIcoFsiFoam`
- Replace `FOAM_APPBIN` with `FOAM_USER_APPBIN` environment variable.

`Make/options`:

- Edit the `EXE_INC = \` entry by adding the following:
  - `-ItractionDisplacement \`
  - `-I\$(LIB_SRC)/dynamicFvMesh/lnInclude \`
  - `$(W_DECOMP_INC) \`
  - `tetDecompositionFiniteElement/lnInclude`
- Edit the `EXE_LIBS = \` entry by adding the following:
  - `-ldynamicFvMesh \`
  - `$(W_DECOMP_LIBS) \`

## Working FSI solver

Our FSI solver *myIcoFsiSolver* should now

- be ready for compilation with `wmake`
- handle solid deformation and influence the flow field by altered geometry
- tackle thermal stresses in the solid (though no thermal coupling with the fluid)
- output stress components

... as long as there is a valid case with weak dynamical coupling ...

## CASE: flappingConsoleSmall

- This is a test case made for the *icoFsiFoam*
- To use it with our new solver we need to add the `solid/constant/thermalProperties` dictionary from *stressedFoam*
- Inspect the directory structure and dictionaries to see what things look like ; )
- This is not necessarily the only way to structure the cases for our solver, but is a consistent and intuitive method.
  Our solver saves the solid states and searches for the solid dictionaries in a `solid` subdirectory in the relevant fluid directories.

## CASE: flappingConsoleSmall with thermals

By changing the `controlDict` to compute thermal stresses and setting some apropriate boundary conditions to an initial field `solid/0/T` we can try to see if this alters the behavior much…The supplied case

## Solver: myInterFsiFoam

Use the same approach as above to create a three–field solver by couple *interFoam* with *stressed-Foam*

- Copy the entire `interFoam` directory (including any subdirectories) to `$FOAM_RUN/myInterF` and rename `interFoam.C` to `myInterFsiFoam.C`
- Copy all relevant files from `myIcoFsiFoam` and rename main solver `.C` file.
- Extract main fluid solver loop algorithm into seperate file called `solveFluids.H`.
- Call all files in similar order as in `myIcoFsiFoam.C` inside time loop of `myInterFsiFoam.C`
- Import all necessary headers (FEM, moving mesh, etc.) from `myIcoFsiFoam.C` into `myInterFsiFoam.C`.
- Edit `myInterFsiFoam/Make/files`: Rename solver, update app-dir. env.variable, include `tractionDisplacement` directory.
- Edit `myInterFsiFoam/Make/options`: Add traction displacement, dynamicFvMesh and all FE decomposition references from `myIcoFsiFoam/Make/options`.

## Solver: myInterFsiFoam

- Change `setPressure.H` line 11:
  `rhoFluid.value();` should read `rho;`
- Change all occurances of `rho` to `rhoSolid` in the files

  ```
  \verb|readThermalProperties|
  \verb|tractionDisplacementFvPatchVectorField.C|
  \verb|readMechanicalProperties|
  \verb|calculateStress.H|
  ```

- Compile and cross fingers ; )

## CASE: softDamBreak

- The damBreak geometry is well suited for an FSI extension. No need to alter the fluid geometry
- The supplied case has a solid part in the dam (identical to the one used in *flappingConsoleSmall*), and a slightly refined mesh.
- Directory structure is altered to comply with our solver algorithm
- Boundary patches must be updated, and dictionaries must be consistent.
- We use the `controlDictionary` from `damBreak` with some modifications

Dam is not really soft, and crashes with too low `E` due to too similar densities between one of the fluids and the solid material. In other words: Dynamic coupling is strong when the solid is not stiff, and our solver is no longer applicable.

## The End

Thank you for your time!

Karl Jacob Maus