Application of dynamic meshes to potentialFreeSurfaceFoam to solve for 6DOF floating body motions

Guilherme Moura Paredes

Faculty of Engineering,
University of Porto,
Porto, Portugal

2012-08-27

Objective

The objective is to modify the solver **potentialFreeSurfaceFoam** to get it to work with moving meshes.

potentialFreeSurfaceFoam is a new solver, distributed with OpenFOAM 2.1.x, which solves free surface flows by approximating the free surface profile with a wave field. Everything is solved in a static grid.

Theory of moving meshes

The continuity equation, eq. 1:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho \vec{v})}{\partial \vec{x}} = 0 \tag{1}$$

$\vec{v}$ - flow velocity; $\vec{x}$ - position vector; ρ - fluid density; t is time.
Integrating eq. 1 over a control volume with moving boundaries:

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega} \rho \, \mathrm{d}\Omega - \int_{S} \frac{\mathrm{d}\vec{r}}{\mathrm{d}t} \cdot \vec{n} \, \mathrm{d}S + \int_{S} \rho \vec{v} \cdot \vec{n} \, \mathrm{d}S \tag{2}$$

Ω - volume of the control volume; S - surface of the control volume; $\vec{r}$ - position of the boundaries.
Setting

$$\frac{\mathrm{d}\vec{r}}{\mathrm{d}t} = \vec{v_b} \tag{3}$$

We get

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega} \rho \, \mathrm{d}\Omega + \int_{S} \rho \, (\vec{v} - \vec{v_b}) \cdot \vec{n} \, \mathrm{d}S \tag{4}$$

Setting up

potentialFreeSurfaceDyMFoam will be constructed based on **potentialFreeSurfaceFoam**

`OF21x`

if `$WM_PROJECT_USER_DIR` hasn't got the same structure as `$WM_PROJECT_DIR`, then

```
cd $WM_PROJECT_DIR
cp -r --parents applications/solvers/incompressible/\
  potentialFreeSurfaceFoam $WM_PROJECT_USER_DIR
cd $WM_PROJECT_USER_DIR/applications/solvers/incompressible
mv potentialFreeSurfaceFoam potentialFreeSurfaceDyMFoam
cd potentialFreeSurfaceDyMFoam
```

otherwise

```
cd $WM_PROJECT_USER_DIR/applications/solvers/incompressible
cp -r $FOAM_SOLVERS/incompressible/\
  potentialFreeSurfaceFoam potentialFreeSurfaceDyMFoam
cd potentialFreeSurfaceDyMFoam
```

Files

First, remove the files resulting from the compilation of **potentialFreeSurfaceDyMFoam**:

```
wclean
```

Since the new solver is named **potentialFreeSurfaceDyMFoam**, rename `potentialFreeSurfaceFoam.C`

```
mv potentialFreeSurfaceFoam.C potentialFreeSurfaceDyMFoam.C
```

New files

Two extra files are needed for solvers with dynamic meshes: `correctPhi.H` and `readControls.H`. These files can be copied from the **pimpleDyMFoam** source code, since they have a structure very close to the one desired for **potentialFreeSurfaceDyMFoam**:

```
cp -r $FOAM_SOLVERS/incompressible/pimpleFoam/\
  pimpleDyMFoam/correctPhi.H .
cp -r $FOAM_SOLVERS/incompressible/pimpleFoam/\
  pimpleDyMFoam/readControls.H .
```

Code structure

`potentialFreeSurfaceDyMFoam/` has now the following structure:

```
potentialFreeSurfaceDyMFoam/
├── potentialFreeSurfaceDyMFoam.C
├── readControls.H
├── correctPhi.H
├── createFields.H
├── UEqn.H
├── pEqn.H
└── Make/
    ├── options
    └── files
```

Changes

The file `Ueqn.H` has no reference to the flux, `phi`, so there is no need to change it.

The file `createFields.H` has a reference to `phi`, but only to create the field and not to manipulate it. So, there is no need to change it to implement dynamic meshes. It will require other changes.

All the other files will need changes.

potentialFreeSurfaceDyMFoam.C

- in line 40, add

  ```
  #include "dynamicFvMesh.H"
  ```

- in line 51, replace

  ```
  #include "createMesh.H"
  ```

  with

  ```
  #include "createDynamicFvMesh.H"
  ```

- in line 64, move the code

  ```
  #include readTimeControls.H
  ```

  to line 52, after

  ```
  #include "createDynamicFvMesh.H"
  ```

- in line 64 add

  ```
  #include "readControls.H"
  ```

potentialFreeSurfaceDyMFoam.C

- in lines 66 and 67 add

```
// Make the fluxes absolute
fvc::makeAbsolute(phi, U);
```

- in line 40, add

```
mesh.update();

if (mesh.changing() && correctPhi)
{
  #include "correctPhi.H"
}

// Make the fluxes relative to the mesh motion
fvc::makeRelative(phi, U);

if (mesh.changing() && checkMeshCourantNo)
  {
    #include "meshCourantNo.H"
  }
```

pEqn.H

- in lines 11 to 14, replace

```
phi = (fvc::interpolate(U) & mesh.Sf())
+ fvc::ddtPhiCorr(rAU, U, phi);
adjustPhi(phi, U, p_gh);
```

with

```
phi = (fvc::interpolate(U) & mesh.Sf());
if (ddtPhiCorr)
 {
   phi += fvc::ddtPhiCorr(rAU, U, phi);
 }
if (p.needReference())
 {
  fvc::makeRelative(phi, U);
  adjustPhi(phi, U, p);
  fvc::makeAbsolute(phi, U);
 }
```

- in line 48 add

```
// Make the fluxes relative to the mesh motion
fvc::makeRelative(phi, U);
```

correctPhi.H

pimpleDyMFoam computes total pressure, `p`, but **potentialFreeSurfaceFoam** computes dynamic pressure, `p_gh`. The instances of variable `p` in `correctPhi.H` must be modified to `p_gh`.

A simple find and replace all command will not work, because there are several commands that use the letter p in `correctPhi.H`. The change must be done case by case.

Replace this:	with this:
`p.boundaryField()`	`p_gh.boundaryField()`
`forAll(p.boundaryField()`	`forAll(p_gh.boundaryField()`
`if (p.boundaryField()`	`if (p_gh.boundaryField()`
`"pcorr", p.dimensions()`	`"pcorr", p_gh.dimensions()`
`setReference(pRefCell, pRefValue)`	`setReference(p_ghRefCell, p_ghRefValue)`

correctPhi.H

potentialFreeSurfaceFoam uses the value of `rAU` interpolated at the cell faces, `rAUf`. This variable must be declared and the instances of variable `rAU` changed to `rAUf`.

- in line 58, change

```
fvm::laplacian(rAU, pcorr) == fvc::div(phi)
```

  to

```
fvm::laplacian(rAUf, pcorr) == fvc::div(phi)
```

- in line 53 add

```
dimensionedScalar rAUf("(1|A(U))", dimTime, 1.0);
```

Make/files

- replace all instances of `potentialFreeSurfaceFoam` with `potentialFreeSurfaceDyMFoam`
- change the compiled solver destination from

```
EXE = $(FOAM_APPBIN)/potentialFreeSurfaceFoam
```

to

```
EXE = $(FOAM_USER_APPBIN)/potentialFreeSurfaceDyMFoam
```

Make/options

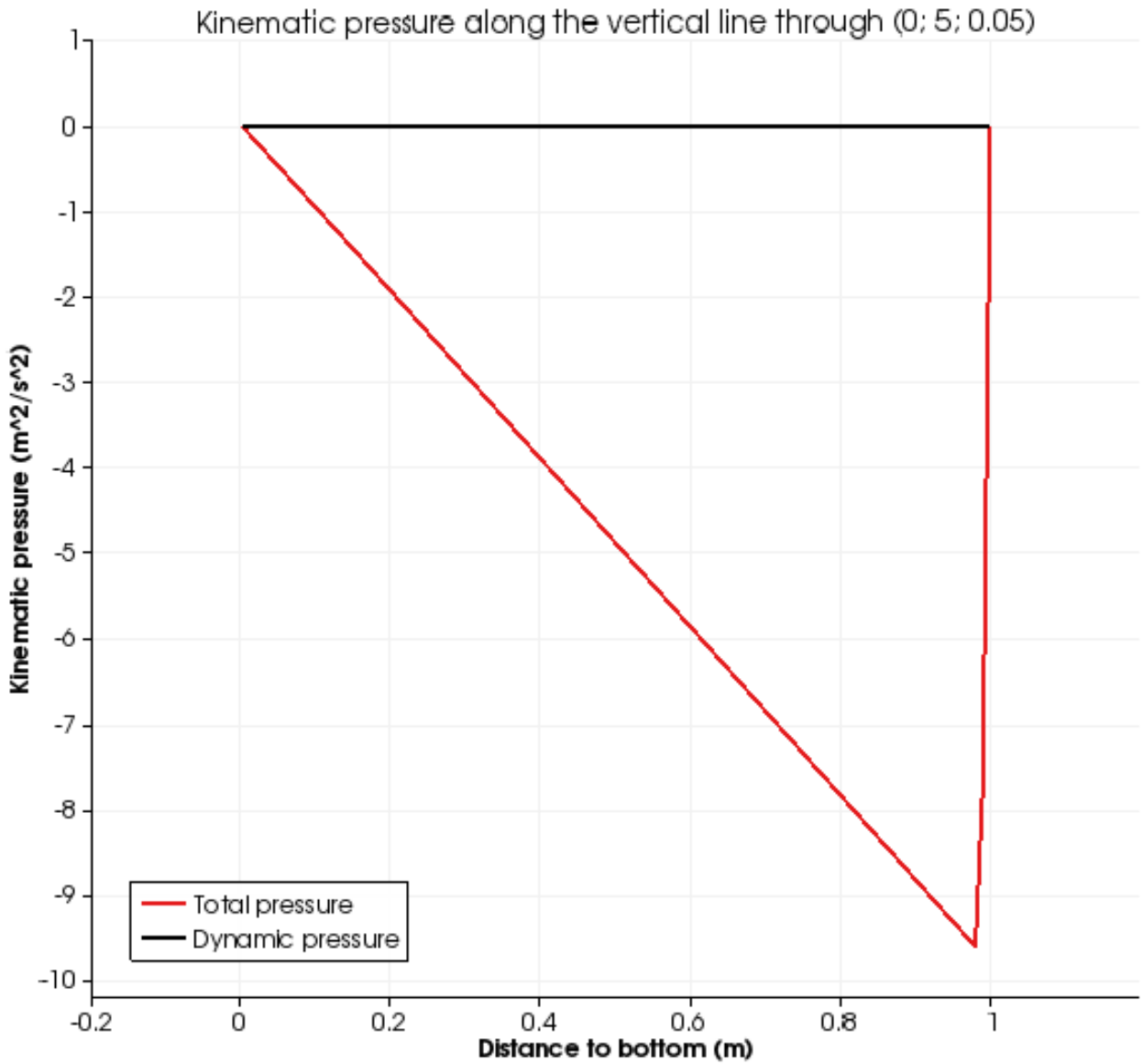
- in line 2 and 3, under `EXE_INC = \`, add

```
-I$(LIB_SRC)/dynamicMesh/lnInclude \
-I$(LIB_SRC)/dynamicFvMesh/lnInclude \
```

- in lines 10 and 11, under `EXE_LIBS = \` add

```
-ldynamicFvMesh \
-ltopoChangerFvMesh \
```

A possible bug



createFields.H

- in line 76, change the code

```
dimensionedVector("zero", dimLength, vector::zero)
```

to

```
dimensionedVector("one", dimLength, vector::one)
```

- in lines 79 and 80 comment the code:

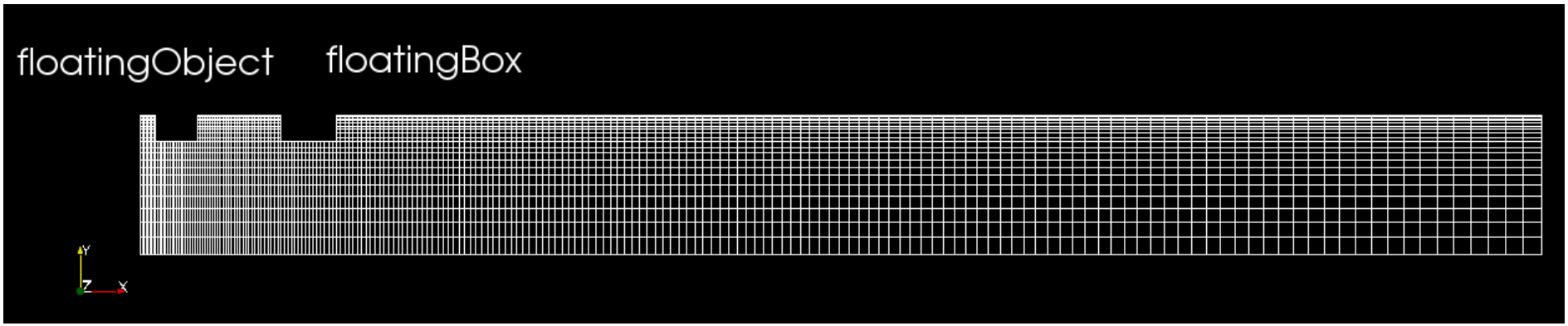
```
/*refLevel.boundaryField()[freeSurfacePatchI]
    == mesh.C().boundaryField()[freeSurfacePatchI];*/
```

Execute

```
wmake
```

Preparing the case

This case is a 2D simulation of a box that "moves" and generates waves (`floatingObject`) on a fluid and another box floating and moving with the waves (`floatingObject`).

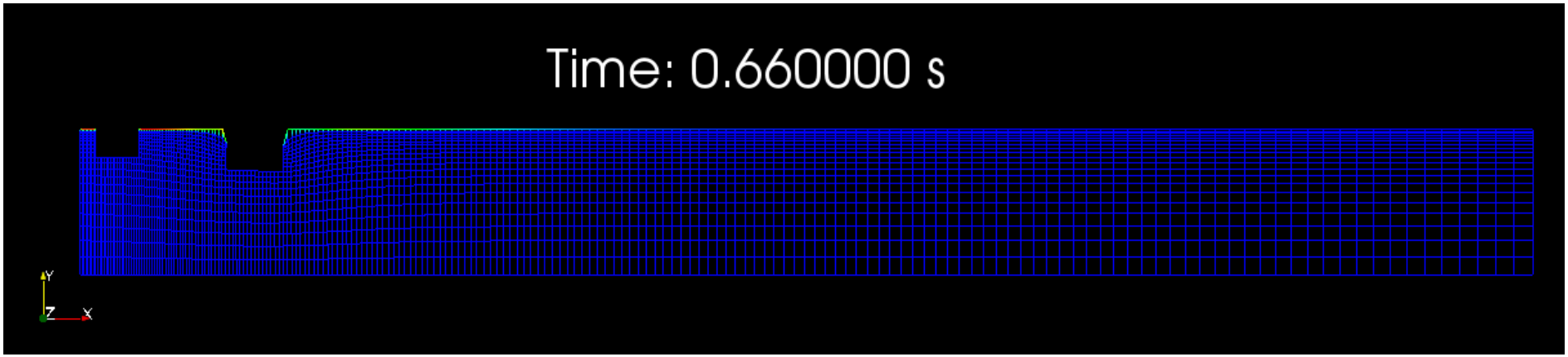


Unpack the case `oscillatingDyMBox` and go to its folder.

Test case - oscillatingDyMBox

Run the case, just by executing the `Allrun` script:

```
./Allrun
```



Modifying the case

- In `0.org/U`, in the patch `floatingObject`, we can modify the frequency and amplitude of the movement of the box that generates waves, by changing the field `amplitude` and `frequency`
- In `0.org/pointDisplacement` we can change the mass and inertia of the `floatingBox` (and other characteristics) and apply some constraints and restraints.
- In `system/topoSetDict` we can change the shape and position of the box that generates waves, by changing the coordinates of the field `box`. This coordinates define the geometry of the moving box. The same can be done for the other box, `floatingBox`, in `system/topoSetDict2`.