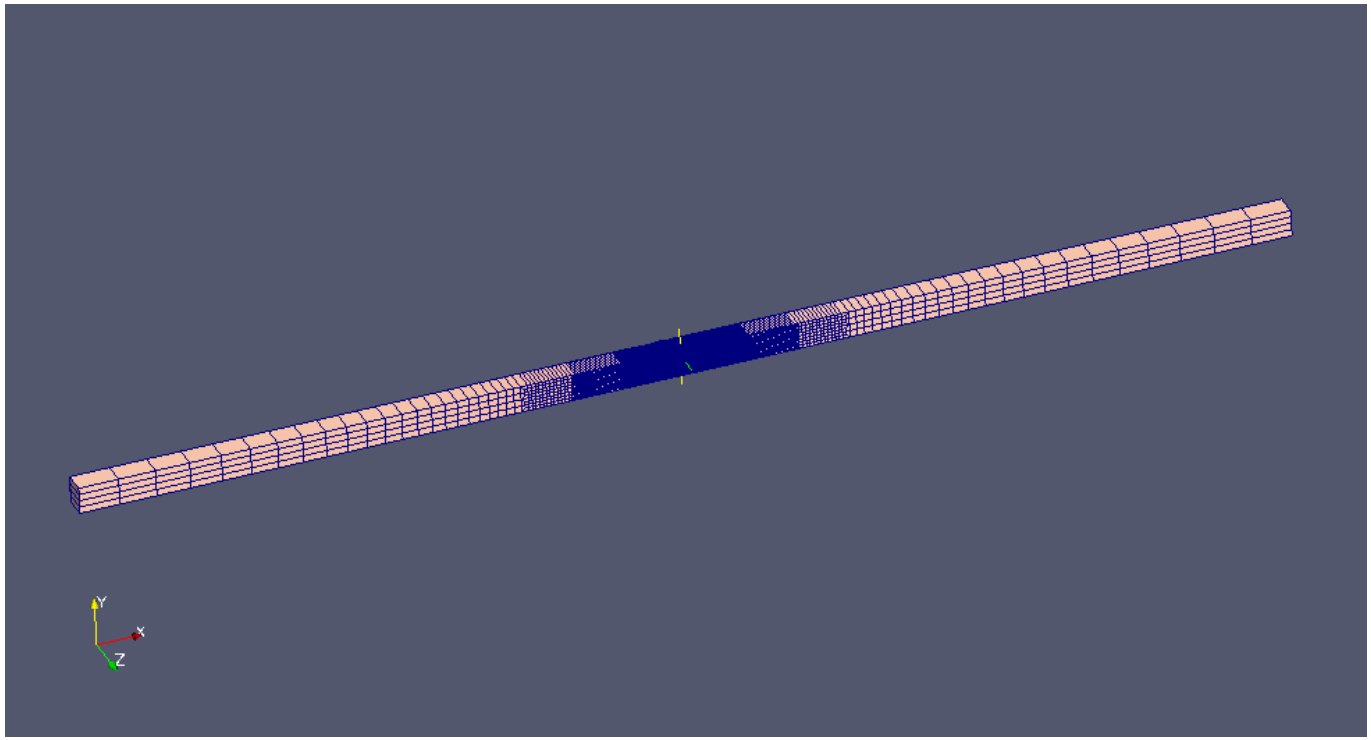
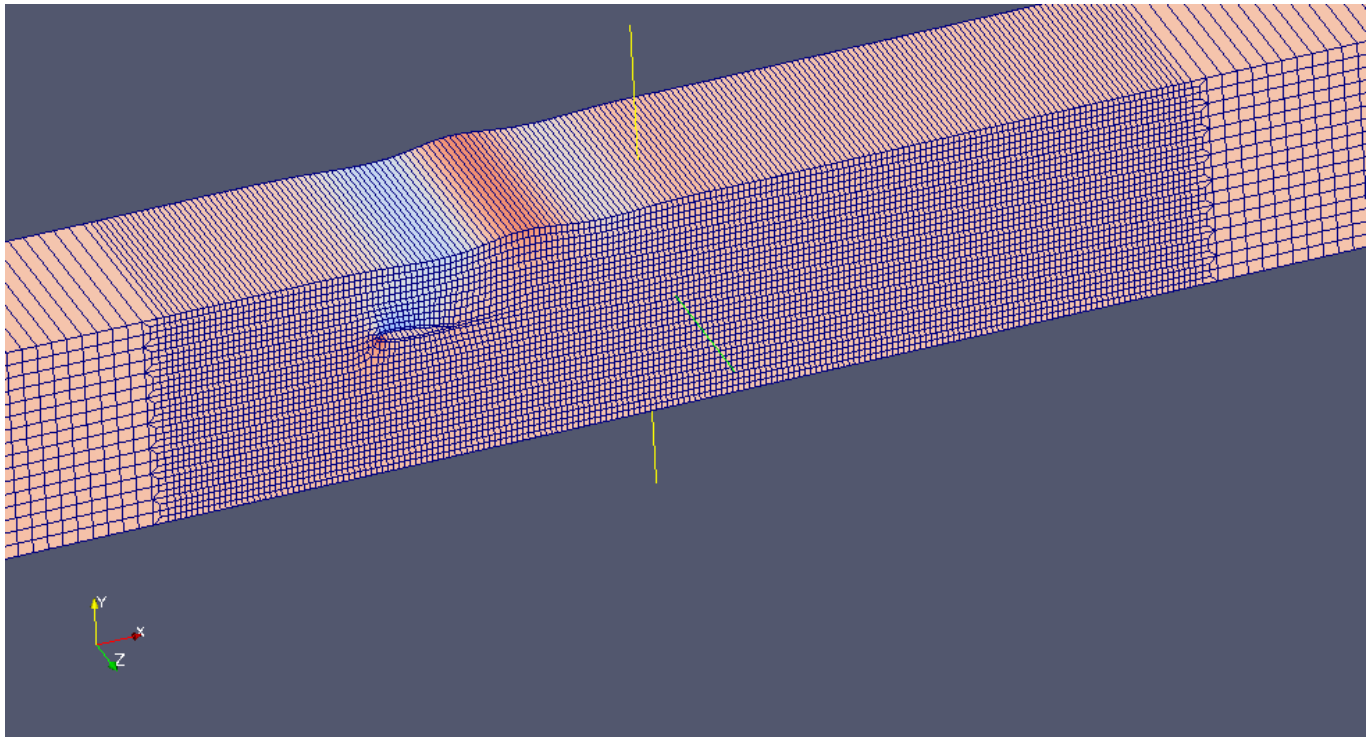
Agenda

- hydrofoil
- modDamBreak
- interTrackFoam

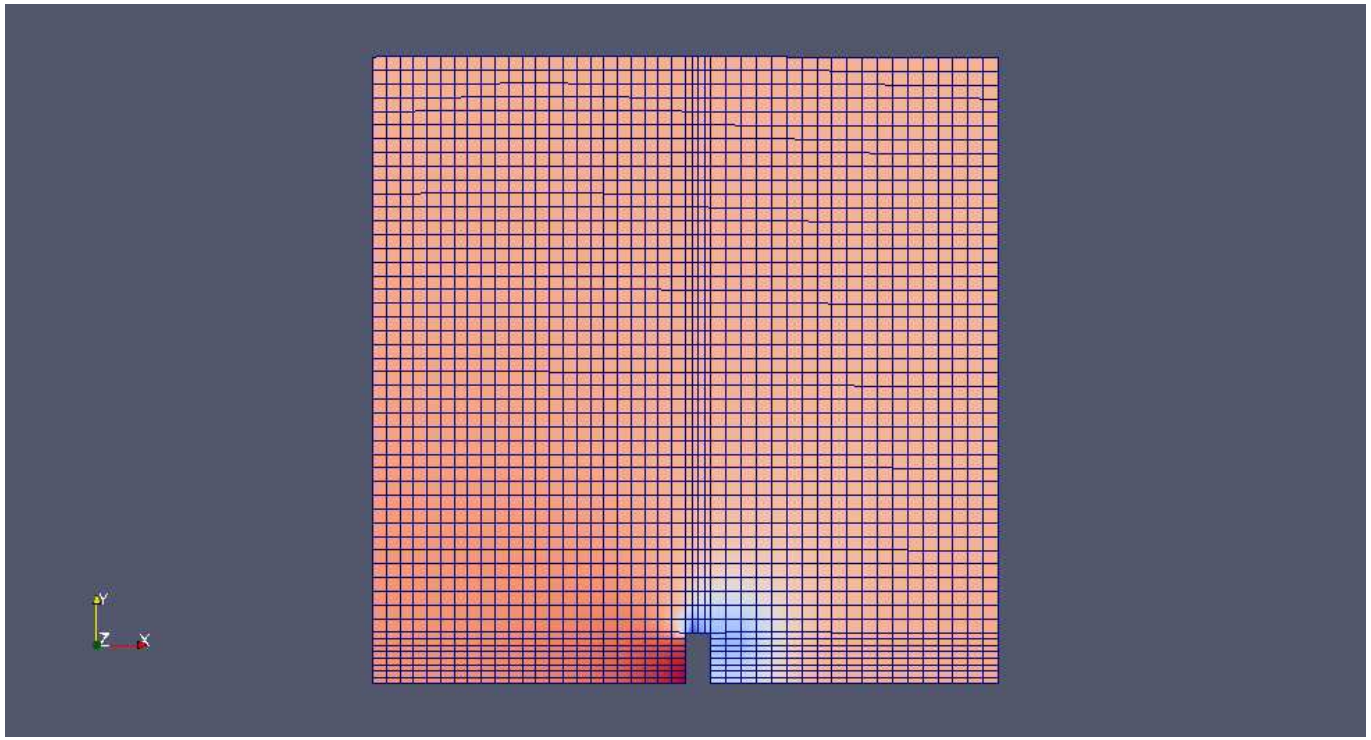
hydrofoil domain



Close-up of hydrofoil



modDamBreak



Procedure creating modDamBreak

```
run
cp -r $FOAM_TUTORIALS/interTrackFoam/hydrofoil .
rm -rf hydrofoil/constant/polyMesh
cp -r $FOAM_TUTORIALS/interFoam/damBreak/constant/\
polyMesh hydrofoil/constant/polyMesh
mv hydrofoil modDamBreak
rm -rf modDamBreak/constant/polyMesh/boundary
```

Procedure creating modDamBreak

- Change boundary conditions for $motionU$, U and p
  - Finite-area discretisation, $makeFaMesh$

Time directory, 0

- $motionU$
- U
- p

Constant directory, *constant*

- *dynamicMeshDict*
- *faMesh*
- *freeSurfaceProperties*
- *polyMesh*
- *transportProperties*

Constant directory, $dynamicMeshDict$

```
twoDMotion          yes;

solver              laplaceFaceDecomposition;

diffusivity         patchEnhanced;

distancePatches     1 (freeSurface);

frozenDiffusion     yes;


pseudoSolid
{
    poissonsRatio           0.3;
    nCorrectors             3;
    convergenceTolerance    1e-9;
};
```

Constant directory, *solvers*

Mesh motion solvers

- *laplaceFaceDecomposition*
- *pseudoSolidFaceDecomposition*
- *RBFMotionSolver*

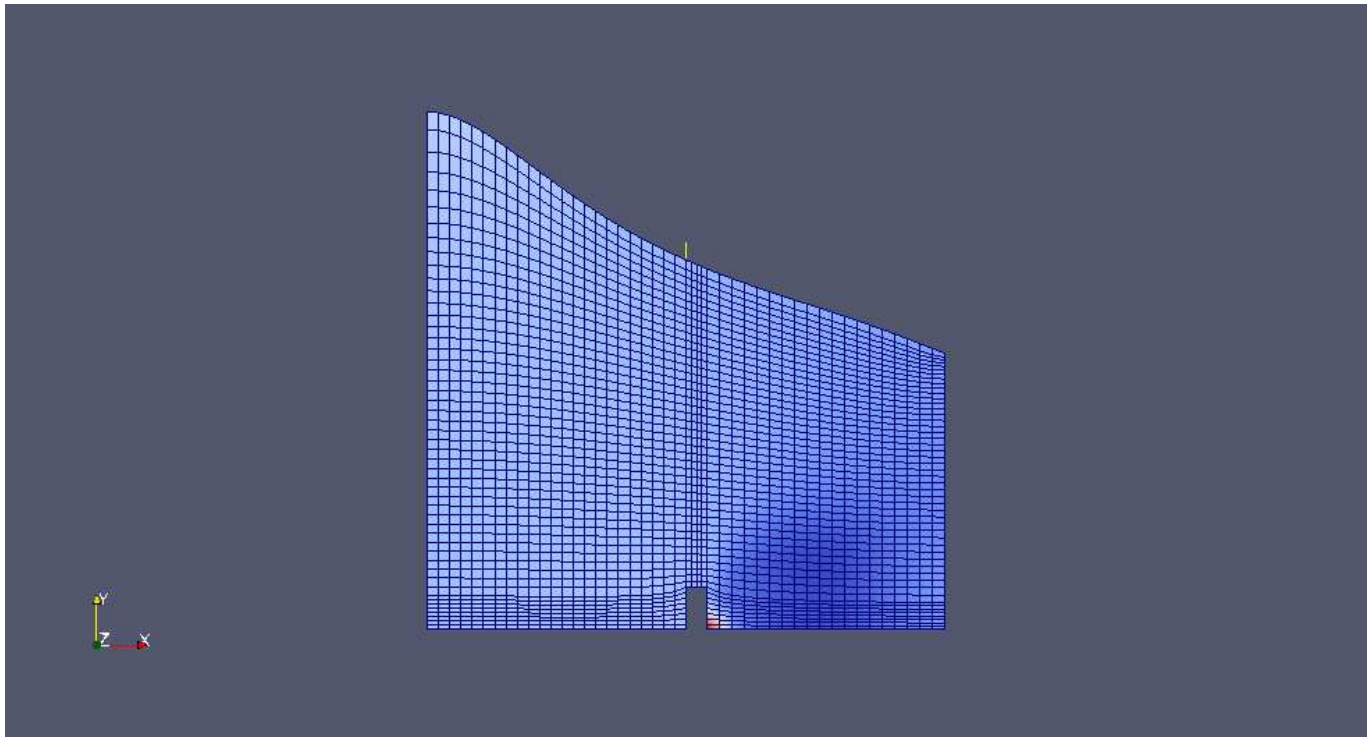
solver, *laplaceFaceDecomposition*

Diffusivity options for *laplaceFaceDecomposition* solver

diffusivity	type
distancebased	*linear*
	quadratic
	exponental
	patchEnhanced
qualitybased	*uniform*
	distortionEnergy
	deformationEnergy
other	*file*

Table 1: The diffusivity options

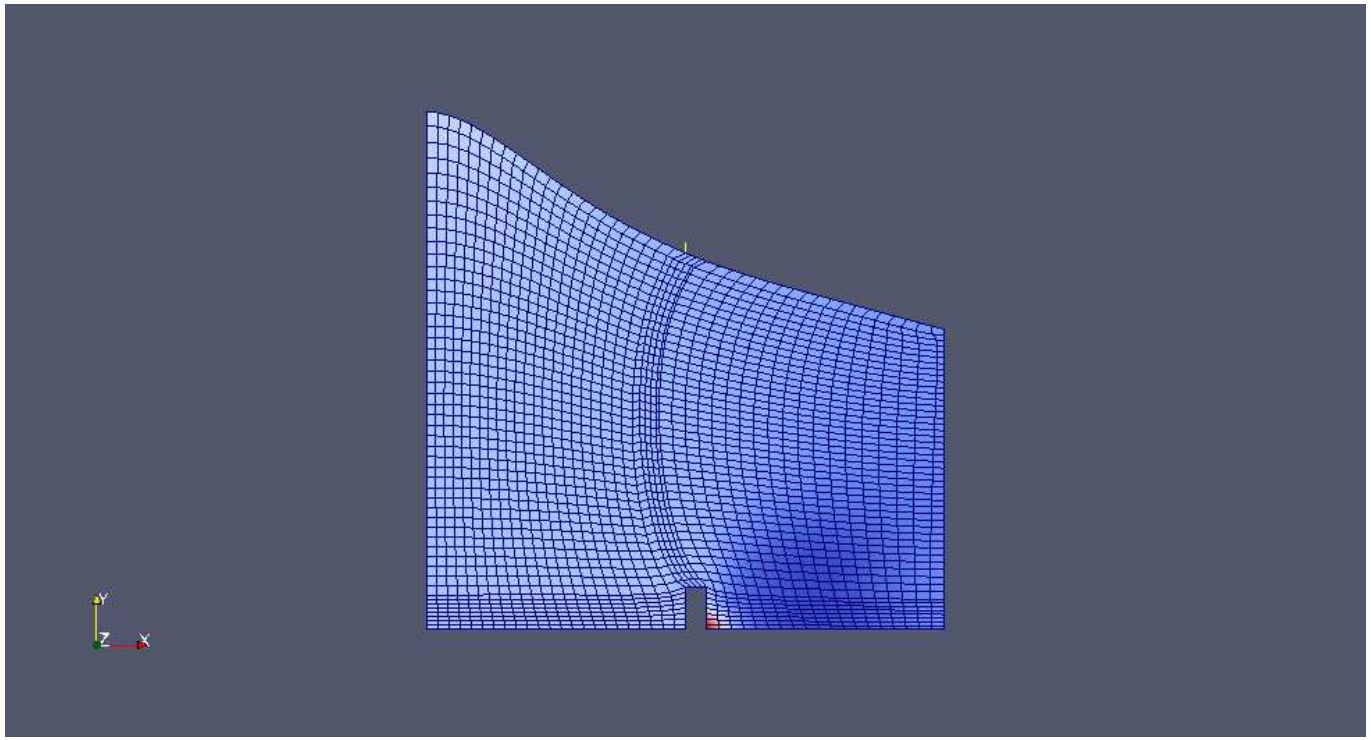
solver, $laplaceFaceDecomposition$



solver, *pseudoSolidFaceDecomposition*

```
{
    poissonsRatio           0.3;
    nCorrectors             3;
    convergenceTolerance    1e-9;
};
```

solver, $pseudoSolidFaceDecomposition$



solver, $RBFMotionSolver$

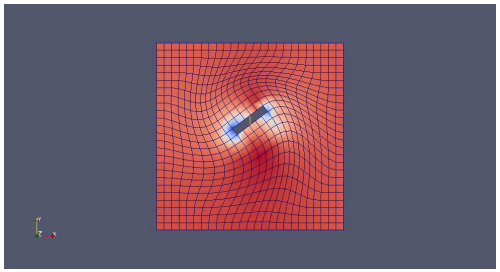


Figure 1: large.

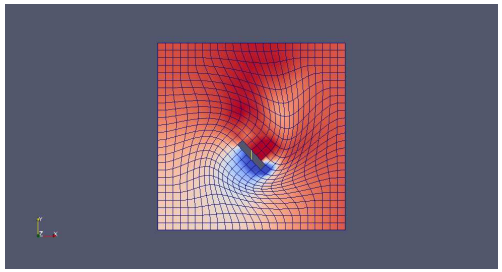


Figure 2: large.

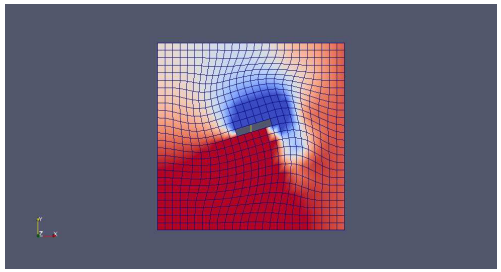


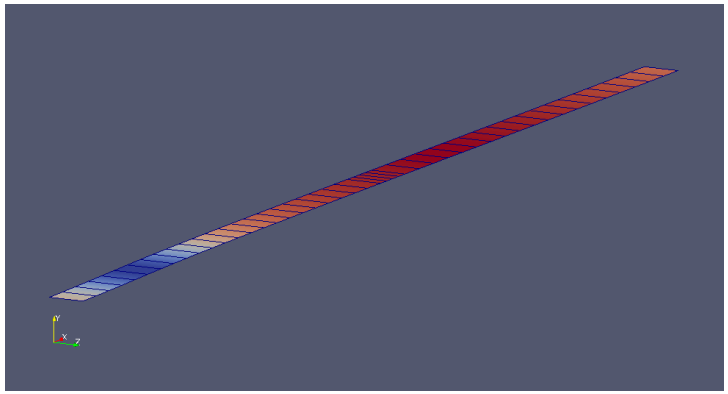
Figure 3: close.

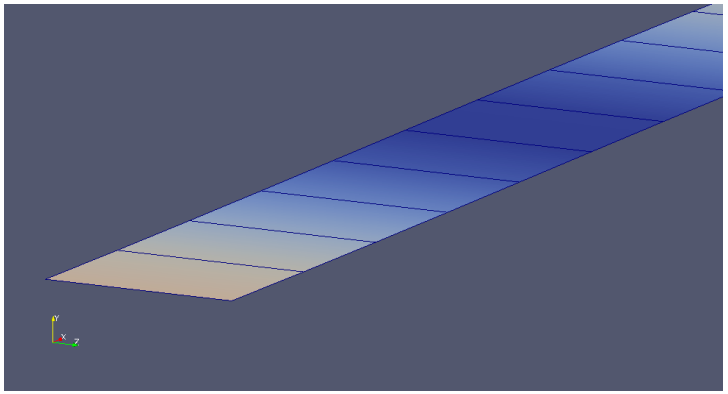
faMeshDefintion

```
polyMeshPatches  1( freeSurface );

boundary
{
    left
    {
        type                patch;
        ownerPolyPatch      freeSurface;
        neighbourPolyPatch  left;
    }

    right
    {
        type                patch;
        ownerPolyPatch      freeSurface;
        neighbourPolyPatch  right;
    }
```

```
    frontAndBackPlanes
    {
        type                empty;
        ownerPolyPatch      freeSurface;
        neighbourPolyPatch  frontAndBackPlanes;
    }
}
```

$faMeshDefinition$



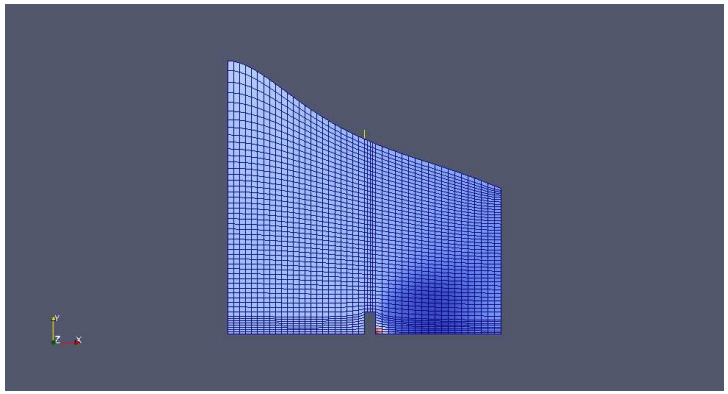


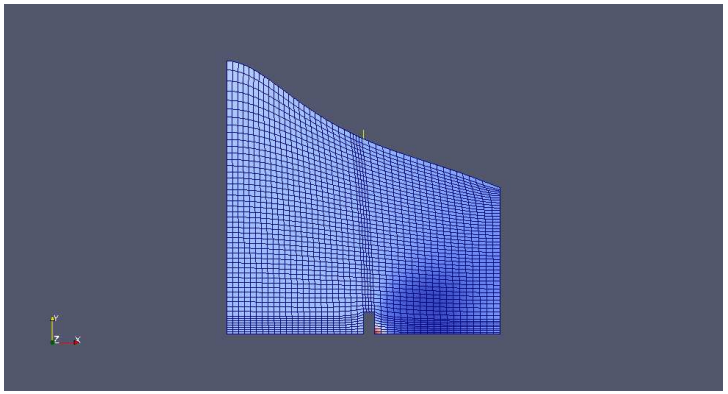
freeSurfaceProperties

```
twoFluids               no;

normalMotionDir         no;

freeSurfaceSmoothing    no;

cleanInterface          yes;

muFluidA                muFluidA    [ 1 -1 -1 0 0 0 0 ]     0;

muFluidB                muFluidB    [ 1 -1 -1 0 0 0 0 ]     1.5e-5;

rhoFluidA               rhoFluidA   [ 1 -3  0 0 0 0 0 ]     1000.0;

rhoFluidB               rhoFluidB   [ 1 -3  0 0 0 0 0 ]     1.0;

surfaceTension          surfaceTension [ 1 -2 0 0 0 0 0 ] 0.0;
```

```
g                       g [ 0 1 -2 0 0 0 0 ] (0 -9.81 0);

fixedFreeSurfacePatches 1 ( inlet );

surfactantProperties
{
    bulkConc            bulkConc            [ 0 -3  0 0  1 0 0 ] 1.0e-2;

    saturatedConc       saturatedSurfConc   [ 0 -2  0 0  1 0 0 ] 5.0e-6;

    adsorptionCoeff     adsorptionCoeff     [ 0  3 -1 0 -1 0 0 ] 40.0;

    desorptionCoeff     desorptionCoeff     [ 0 -3  0 0  1 0 0 ] 8.93e-2;

    bulkDiffusion       bulkDiffusion       [ 0  2 -1 0  0 0 0 ] 1.0e-9;

    diffusion           diffusion           [ 0  2 -1 0  0 0 0 ] 1.0e-9;

    temperature         temperature         [ 0  0  0 1  0 0 0 ] 293.0;
}
```

normalMotionDir





System

- $contolDict$
- $faSolution$
- $faSchemes$
- $fvSolution$
- $fvSchemes$
- $tetFemSolution$

interTrackFoam

```
#include "fvCFD.H"
#include "motionSolver.H"
#include "freeSurface.H"
#include "OFstream.H"

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

int main(int argc, char *argv[])
{
#   include "setRootCase.H"
#   include "createTime.H"
#   include "createMeshNoClear.H"
#   include "createFields.H"
#   include "initContinuityErrs.H"


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
    Info << "\nStarting time loop\n" << endl;

    for (runTime++; !runTime.end(); runTime++)
    {
        Info << "Time = " << runTime.value() << endl;

#       include "readInterfaceSIMPLEControls.H"
#       include "CourantNo.H"

        interface.updateDisplacementDirections();

        interface.moveMeshPointsForOldFreeSurfDisplacement();

        interface.smooth();

        // --- SIMPLE loop
        for (label timeCorr=0; timeCorr<=nTimeCorr; timeCorr++)
        {
            p.storePrevIter();

            interface.correctBoundaryConditions();
```

```
tmp<fvVectorMatrix> UEqn
(
    fvm::ddt(rho, U)
  + fvm::div(phiNet, U)
  - fvm::laplacian(mu, U)
);

UEqn().relax();

solve(UEqn() == - fvc::grad(p));

volScalarField AU = UEqn().A();

U = UEqn().H()/AU;
U.correctBoundaryConditions();

UEqn.clear();

phi = (fvc::interpolate(U) & mesh.Sf());
```

```
        // Non-orthogonal pressure corrector loop
        for (label nonOrth=0; nonOrth<=nNonOrthCorr; nonOrth++)
        {
            tmp<fvScalarMatrix> pEqn
            (
                fvm::laplacian(1.0/AU, p) == fvc::div(phi)
            );

            pEqn().setReference(0, 0.0);
            pEqn().solve();

            if (nonOrth == nNonOrthCorr)
            {
                phi -= pEqn().flux();
            }
        }

#       include "continuityErrs.H"

        // Explicitly relax pressure for momentum corrector
        p.relax();
```

```
            // Momentum corrector
            U -= fvc::grad(p)/AU;
            U.correctBoundaryConditions();

            // Move mesh
            interface.movePoints();

            // Update motion fluxes
            phiNet = fvc::interpolate(rho)*(phi - fvc::meshPhi(rho, U));

#           include "freeSurfaceContinuityErrs.H"
        }

        runTime.write();

        Info << "ExecutionTime = "
            << scalar(runTime.elapsedCpuTime())
            << " s\n" << endl << endl;
    }
```

```
    Info << "End\n" << endl;

    return(0);
}
```