How to implement an application

Example: Electric conduction in a rod surrounded by air

Governing equations

Maxwell's equation:

$$\nabla \times E = 0 \tag{9}$$

where E is the electric field strength.

$$\nabla \cdot B = 0 \tag{10}$$

where B is the magnetic flux density.

$$\nabla \times H = J \tag{11}$$

where H is the magnetic field strength and J is current density.

Charge continuity:

$$\nabla \cdot J = 0 \tag{12}$$

Ohm's law:

$$J = \sigma E \tag{13}$$

where σ is the electric conductivity.
Constitutive law:

$$B = \mu_0 H \tag{14}$$

where μ_0 is the magnetic permeability of vacuum.

Combining Equations (1)-(6) and assuming Coulomb gauge condition ($\nabla \cdot A = 0$) leads to a Poisson equation for the magnetic potential and a Laplace equation for the electric potential...

Governing equations in OpenFoam

Magnetic potential:

$$\nabla^2 A = \mu_0 \sigma (\nabla \phi) \qquad (15)$$

OpenFOAM representation:

```
solve
    (
    fvm::laplacian(A) ==
    sigma*muMag*(fvc::grad(ElPot))
    );
```

Electric potential:

$$\nabla \cdot [\sigma(\nabla \phi)] = 0 \qquad (16)$$

OpenFOAM representation:

```
solve
    (
    fvm::laplacian(sigma,ElPot)
    );
```

We see that A depends on ϕ, but not vice-versa.

Implementing the rodFoam solver

Create the basic files in your user directory:

```
cd $WM_PROJECT_USER_DIR
mkdir -p applications/solvers/electromagnetics/rodFoam
cd applications/solvers/electromagnetics/rodFoam
foamNewSource App rodFoam
tree
```

We see:

```
.
|-- Make
|   |-- files
|   `-- options
`-- rodFoam.C
```

Make sure that the binary file ends up in your user directory:

```
sed -i s/FOAM_APPBIN/FOAM_USER_APPBIN/g Make/files
```

Add a few lines to rodFoam.C

We need a mesh to discretize our equations on, and we need to initialize properties and fields. After `#include "createTime.H"`, add:

```
#include "createMesh.H"     #In the OpenFOAM installation
#include "createFields.H"   #Must be implemented - see next slides
```

Continue adding (after the above), our equations:

```
    solve ( fvm::laplacian(sigma, ElPot) );
    solve ( fvm::laplacian(A)==sigma*muMag*(fvc::grad(ElPot)) );
```

Add some additional things that can be computed when we know `A` and `ElPot`:

```
    B = fvc::curl(A);
    Je = -sigma*(fvc::grad(ElPot));
```

We also want to write out the results to a new time directory.
Continue adding:

```
    runTime++;
    sigma.write();
    ElPot.write();
    A.write();
    B.write();
    Je.write();
```

The createFields.H file (1/6)

We need to construct and initialize `muMag`, `sigma`, `Elpot`, `A`, `B`, and `Je`.
Edit the `createFields.H` file.

Read `muMag` from a dictionary:

```
Info<< "Reading physicalProperties\n" << endl;
IOdictionary physicalProperties
(
    IOobject
    (
        "physicalProperties",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ,
        IOobject::NO_WRITE
    )
);
dimensionedScalar muMag
(
    physicalProperties.lookup("muMag")
);
```

The createFields.H file (2/6)

Construct `volScalarField sigma`:

```
Info<< "Reading field sigma\n" << endl;
volScalarField sigma
(
    IOobject
    (
        "sigma",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

The createFields.H file (3/6)

Construct `volScalarField` `Elpot`:

```
volScalarField ElPot
(
    IOobject
    (
        "ElPot",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

The createFields.H file (4/6)

Construct `volVectorField` `A`:

```
Info<< "Reading field A\n" << endl;
volVectorField A
(
    IOobject
    (
        "A",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

The createFields.H file (5/6)

Construct and initialize `volVectorField B`:

```
Info << "Calculating magnetic field B \n" << endl;
volVectorField B
(
    IOobject
    (
        "B",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    fvc::curl(A)
);
```

The createFields.H file (6/6)

Construct and initialize `volVectorField` `Je`:

```
volVectorField  Je
(
    IOobject
    (
        "Je",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    -sigma*(fvc::grad(ElPot))
);
```

Compile the solver

We have implemented a solver, which is compiled by:

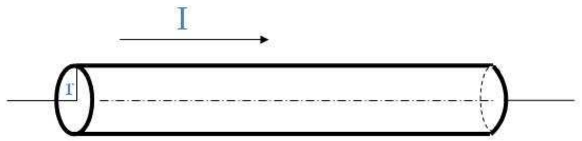
```
wmake
```

If successful, the output should end something like:

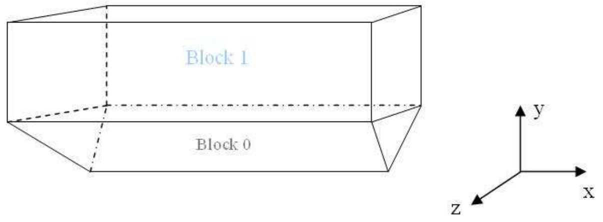
```
-o /chalmers/users/hani/OpenFOAM/hani-2.1.x/platforms/linux64GccDPOpt/bin/rodFoam
```

We now need a case to use the solver on. It is provided to you, since it is too much to describe in slides.

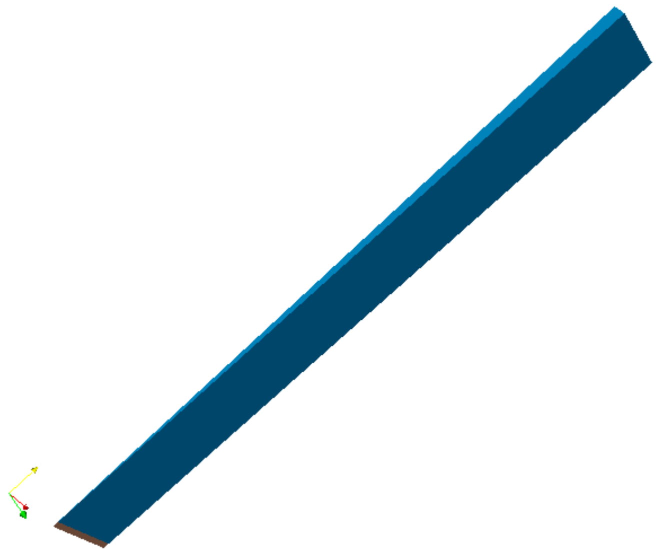
Geometry and mesh, the rodFoamCase case



Electric rod.



Computational domain



In paraFoam
A 2D axi-symmetric case, with a wedge mesh

Boundary and initial conditions

- We solve for the magnetic potential `A` (A) and the electric potential `ElPot` (ϕ), so we need boundary conditions:

	block 0, sides	block 1, sides	block1, top
A	$\nabla A = 0$	$\nabla A = 0$	$A = 0$
ϕ	$\phi_{left} = 707, \phi_{right} = 0$	$\nabla \phi = 0$	$\nabla \phi = 0$

  and we initialize the fields to zero.

- The internal field of the electric conductivity sigma (σ) is nonuniform:

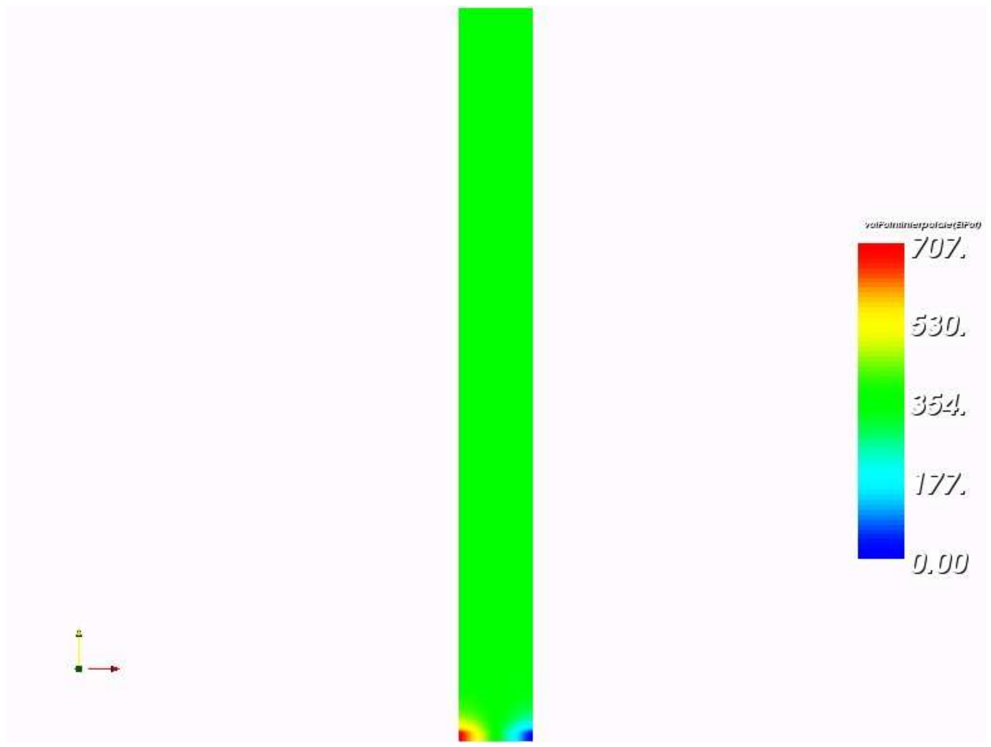
$$\sigma = \begin{cases} 2700 & \text{if } x < R \text{ where R -radius of the block 1} \\ 1e-5 & \text{otherwise} \end{cases}$$

  so we use a `volScalarField` and `setFields` to set the internal field.

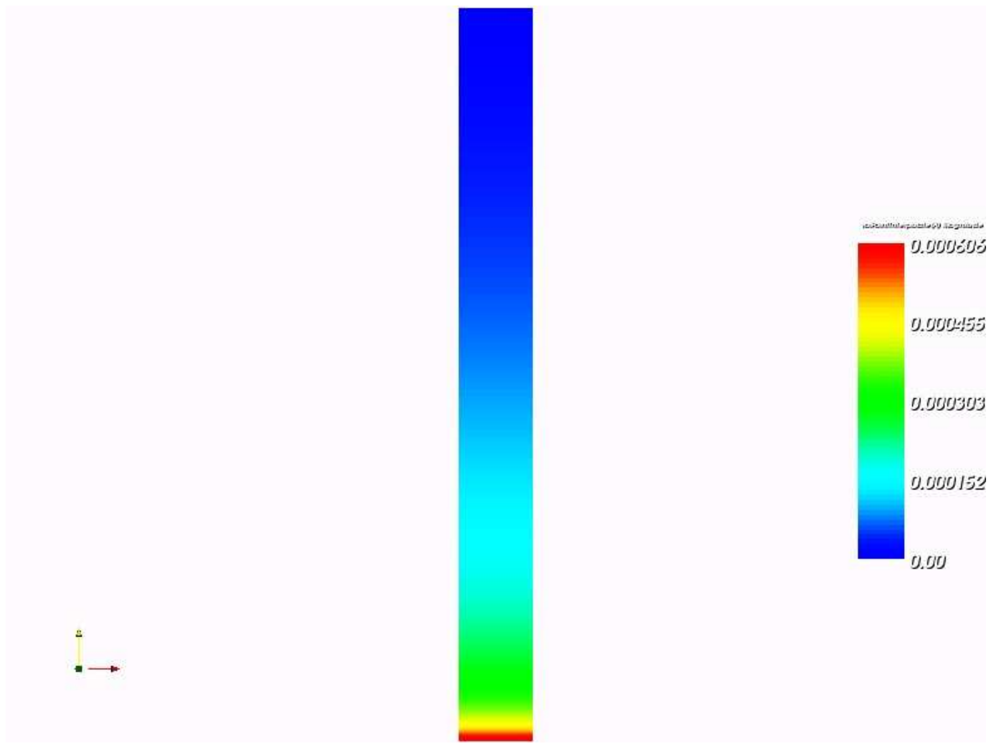
- The magnetic permeability of vacuum (μ_0) is read from the `constant/physicalProperties` dictionary.

Run and view the results in paraFoam

```
./Allrun 2>&1 | tee log_Allrun
```



Electric potential (ϕ)



Magnitude of magnetic potential vector (A)

Validation of components of A and B using Gnuplot

- The `Allrun` script also ran `sample` using dictionary `system/sampleDict`
- For this we need to extract the components:

```
foamCalc components A
foamCalc components B
```

- The results are validated with the analytical solution using Gnuplot:

```
gnuplot rodComparisonAxBz.plt
```

- Visualize using:

```
gv rodAxVSy.ps
gv rodBzVSy.ps
```

Analytic solution

- Analytic solution for x component of magnetic potential vector A

$$A_x = \begin{cases} A_x(0) - \frac{\mu_0 J x^2}{4} & \text{if } r < R, \\ A_x(0) - \frac{\mu_0 J R^2}{2}[0.5 + ln(r/R)] & \text{otherwise} \end{cases}$$

  where $A_x(0) = 0.000606129$, $J = 19.086e + 7$ is the current density and R is the radius of the electric rod.

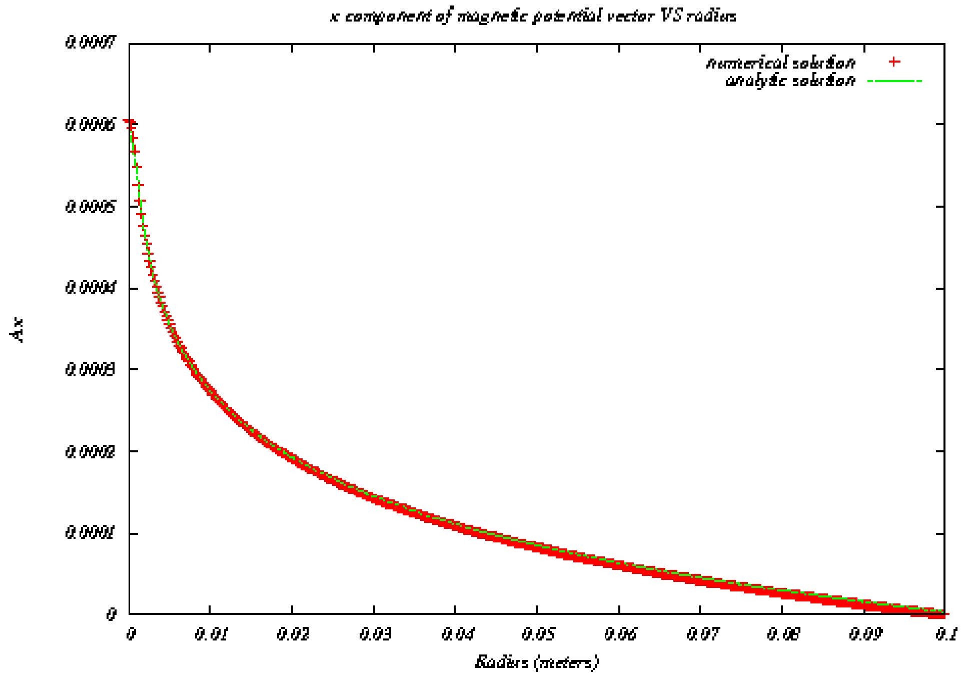
- Analytic solution for z component of magnetic field B

$$B_z = \begin{cases} \frac{\mu_0 J x}{2} & \text{if } r < R, \\ \frac{\mu_0 J R^2}{2r} & \text{otherwise} \end{cases}$$

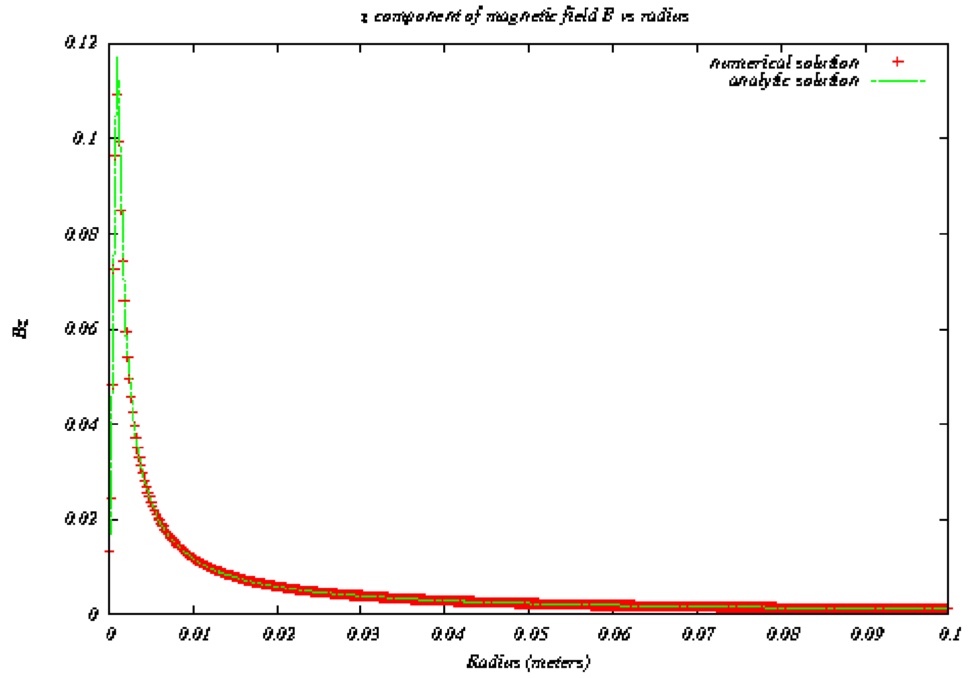
  where $J = 19.086e + 7$ is the current density and R is the radius of the electric rod.

- Have a look in `rodComparisonAxBz.plt` to see how to plot a function in Gnuplot.

Validation



x-component of magnetic potential vector A vs radius of the domain.



z-component of the magnetic field B vs radius of the domain

How to modify an existing application

- The applications are located in the `$WM_PROJECT_DIR/applications` directory (equivalent to `$FOAM_APP`. Go there using alias `app`).
- Copy an application that is similar to what you would like to do and modify it for your purposes. In this case we will make our own copy of the `icoFoam` solver and put it in our `$WM_PROJECT_USER_DIR` with the same file structure as in the OpenFOAM installation:

```
foam
cp -r --parents applications/solvers/incompressible/icoFoam $WM_PROJECT_USER_DIR
cd $WM_PROJECT_USER_DIR/applications/solvers/incompressible
mv icoFoam passiveScalarFoam
cd passiveScalarFoam
wclean
mv icoFoam.C passiveScalarFoam.C
```

- Modify `Make/files` to:

```
passiveScalarFoam.C
EXE = $(FOAM_USER_APPBIN)/passiveScalarFoam
```

- Compile with `wmake` in the `passiveScalarFoam` directory. `rehash` if necessary.
- Test that it works on the `cavity` case...

Test on cavity case

We will quickly visit the run directory to test...

```
pushd $FOAM_RUN #so that we can easily go back to the current directory
rm -r cavity
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity .
blockMesh -case cavity
passiveScalarFoam -case cavity
```

After checking that it worked, go back to the `passiveScalarFoam` directory:

```
popd #brings you back to the directory where you typed the pushd command
```

Add a passive scalar transport equation (1/3)

- Let's add, to `passiveScalarFoam`, the passive scalar transport equation

$$\frac{\partial s}{\partial t} + \nabla \cdot (\mathbf{u}\; s) = 0$$

- We must modify the solver:
  - Create `volumeScalarField s` (do the same as for `p` in `createFields.H`, since both are scalar fields)
  - Add the equation `solve(fvm::ddt(s) + fvm::div(phi, s));` before `runTime.write();` in `passiveScalarFoam.C`.
  - Compile `passiveScalarFoam` using `wmake`
- We must modify the case - next slide ...

Add a passive scalar transport equation (2/3)

- We must modify the case:
  - Use the `icoFoam/cavity` case as a base:
    ```
    run
    cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity passiveCavity
    cd passiveCavity
    ```
  - Copy the `0/p` file to `0/s` and modify `p` to `s` in that file. Choose approprate dimensions for the scalar field (not important now).
  - In `fvSchemes`, add (if you don't, it will complain):
    ```
        div(phi,s)        Gauss linearUpwind Gauss;
    ```
  - In `fvSolution`, copy the solution settings from `U` (since the equations for velocity and `s` are similar), and just change `U` to `s`. (if you use `PCG`, as for `p`, it will complain - try it yourself!)
- We must initialize and run the case - next slide ...

Add a passive scalar transport equation (3/3)

- We must initialize `s`:
  - `cp $FOAM_TUTORIALS/multiphase/interFoam/laminar/damBreak/system/setFieldsDict system`
  - Set `defaultFieldValues`:
    `volScalarFieldValue s 0`
  - Modify the bounding box to:
    `box (0.03 0.03 -1) (0.06 0.06 1);`
  - Set `fieldValues`:
    `volScalarFieldValue s 1`
- Run the case:
  `blockMesh`
  `setFields`
  `passiveScalarFoam >& log`
  `paraFoam` - mark `s` in Volume Fields, color by `s` (cell value) and run an animation.
- You can see that although there is no diffusion term in the equation, there is massive diffusion in the results. This is due to mesh resolution, numerical scheme etc. The `interfoam` solver has a special treatment to reduce this kind of diffusion.

Add particles to the interFoam/damBreak case

Add the solidParticleCloud class to the interFoam/damBreak tutorial by doing the following, and you will have some nice animation to view.

Copy the interFoam solver, clean up, re-name and compile:

```
cd $WM_PROJECT_DIR
cp -r --parents applications/solvers/multiphase/interFoam $WM_PROJECT_USER_DIR
cd $WM_PROJECT_USER_DIR/applications/solvers/multiphase
mv interFoam solidParticleInterFoam
cd solidParticleInterFoam
rm -r Allw* interDyMFoam LTSInterFoam MRFInterFoam porousInterFoam
wclean
rm -rf Make/linux*
mv interFoam.C solidParticleInterFoam.C
sed -i.orig s/interFoam/solidParticleInterFoam/g Make/files
sed -i s/FOAM_APPBIN/FOAM_USER_APPBIN/g Make/files
wmake
```

At this point you can check that the code still works for the damBreak tutorial.

Add particles to the interFoam/damBreak case

Now we will add functionality from the `solidParticleCloud` class.
Modify `solidParticleInterFoam.C`:

Include the class declarations in `solidParticleCloud.H`.
After `#include "twoPhaseMixture.H`, add:

```
#include "solidParticleCloud.H"
```

Create a solidParticleCloud object.
After `#include "setInitialDeltaT.H"`, add:

```
solidParticleCloud particles(mesh);
```

Move the particles.
Before `runTime.write();`, add:

```
particles.move(g);
```

Add particles to the interFoam/damBreak case

We need to add some libraries when we compile.
Make sure that `Make/options` looks like this:

```
EXE_INC = \
    -I$(LIB_SRC)/transportModels \
    -I$(LIB_SRC)/transportModels/incompressible/lnInclude \
    -I$(LIB_SRC)/transportModels/interfaceProperties/lnInclude \
    -I$(LIB_SRC)/turbulenceModels/incompressible/turbulenceModel \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/lagrangian/basic/lnInclude \
    -I$(LIB_SRC)/lagrangian/solidParticle/lnInclude \
    -I$(LIB_SRC)/meshTools/lnInclude

EXE_LIBS = \
    -ltwoPhaseInterfaceProperties \
    -lincompressibleTransportModels \
    -lincompressibleTurbulenceModel \
    -lincompressibleRASModels \
    -lincompressibleLESModels \
    -lfiniteVolume \
    -llagrangian \
    -lsolidParticle
```

Compile:

```
wmake
```

Add particles to the interFoam/damBreak case

We need to set up a case, based on the original `damBreak` case:

```
run
cp -r $FOAM_TUTORIALS/multiphase/interFoam/ras/damBreak solidParticleDamBreak
cd solidParticleDamBreak
```

Initialize the particles:

```
mkdir -p 0/lagrangian/defaultCloud
```

add files for diameter (`d`), positions (`positions`) and velocity (`U`)...

...and set the particle properties in `constant/particleProperties`...

Add particles to the interFoam/damBreak case

Diameter file (`0/lagrangian/defaultCloud/d`):

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       scalarField;
    location    "0";
    object      d;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


2
(
2.0e-3
2.0e-3
)

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

Add particles to the interFoam/damBreak case

Positions file (`0/lagrangian/defaultCloud/positions`):

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       Cloud<solidParticle>;
    location    "0";
    object      positions;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

2
(
(1e-2 0.58 0.005) 0
(2e-2 0.58 0.005) 0
)


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

Add particles to the interFoam/damBreak case

Velocity file (`0/lagrangian/defaultCloud/U`):

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       vectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


2
(
(1.7e-1 0 0)
(1.7 0 0)
)


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

Add particles to the interFoam/damBreak case

Particle properties file (`constant/particleProperties`):

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      particleProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

rhop rhop [ 1 -3  0  0  0  0  0] 1000;
e    e    [ 0  0  0  0  0  0  0] 0.8;
mu   mu   [ 0  0  0  0  0  0  0] 0.2;

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

Add particles to the interFoam/damBreak case

Run and animate using `foamToVTK` and `paraview`:

```
blockMesh
setFields
solidParticleInterFoam 2>&1 | tee log_solidParticleInterFoam
foamToVTK
paraview
```

- File/open: VTK/solidParticeDamBreak_..vtk
- File/open: VTK/lagrangian/defaultCloud/defaultCloud_..vtk
- For the solidParticleDamBreak object: Display: Opacity 0,3. Color By: alpha1 (cell values)
- For the defaultCloud object: Create box glyphs (length: 10/10/10, Scale Mode off) to visualize the particles.
- Run the animation and enjoy...