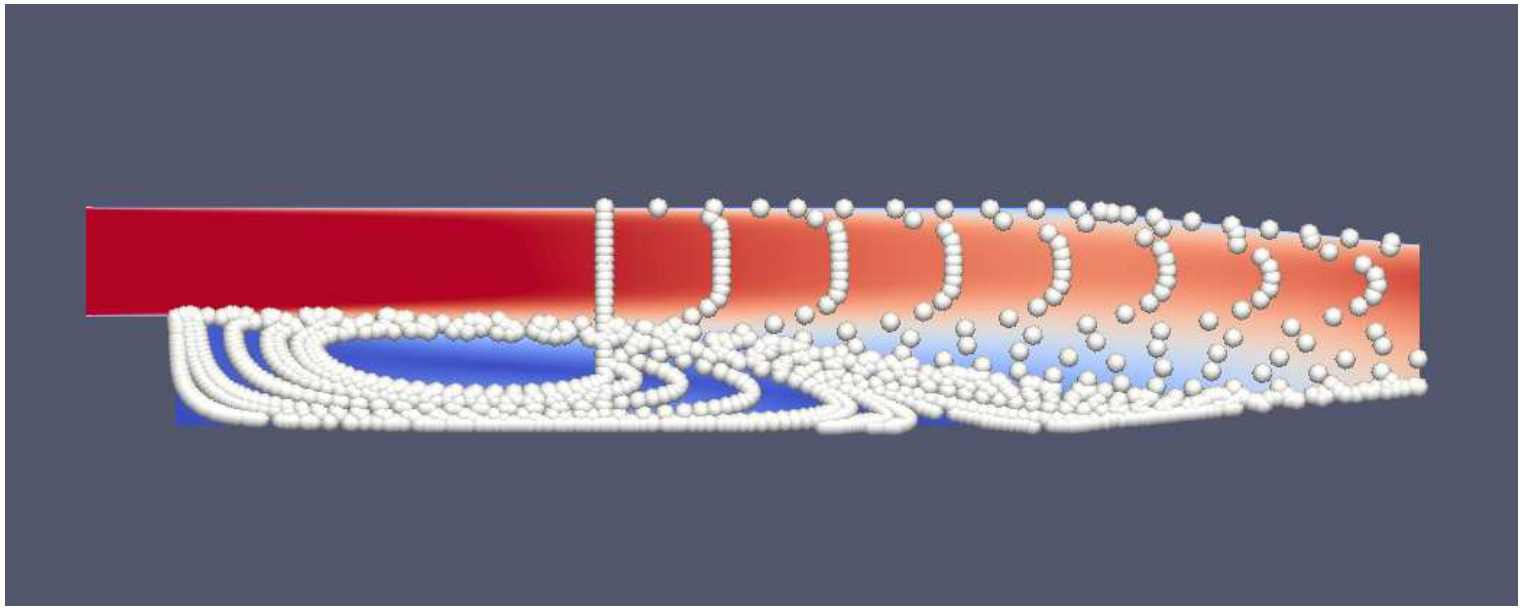
Introduction to particle injection in OpenFoam



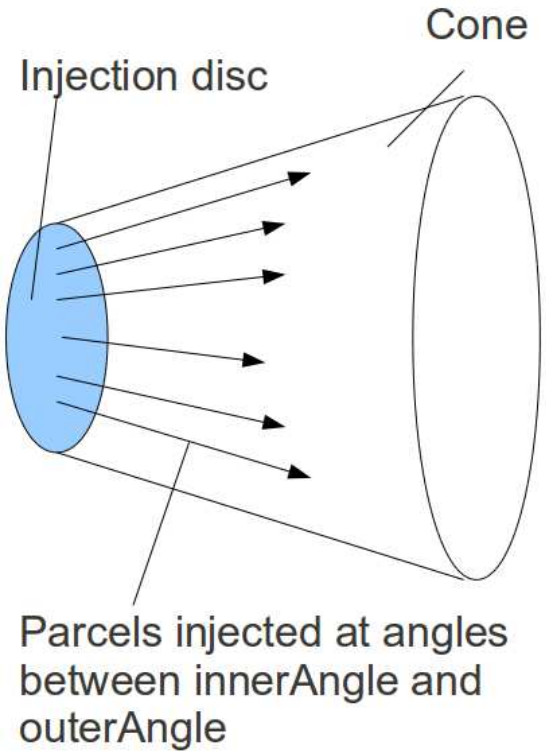
Agenda

- Short introduction of particle injection with unitInjector and hollowConeInjector in dieselSpray.
- Tutorial of particle injection in solidParticle.

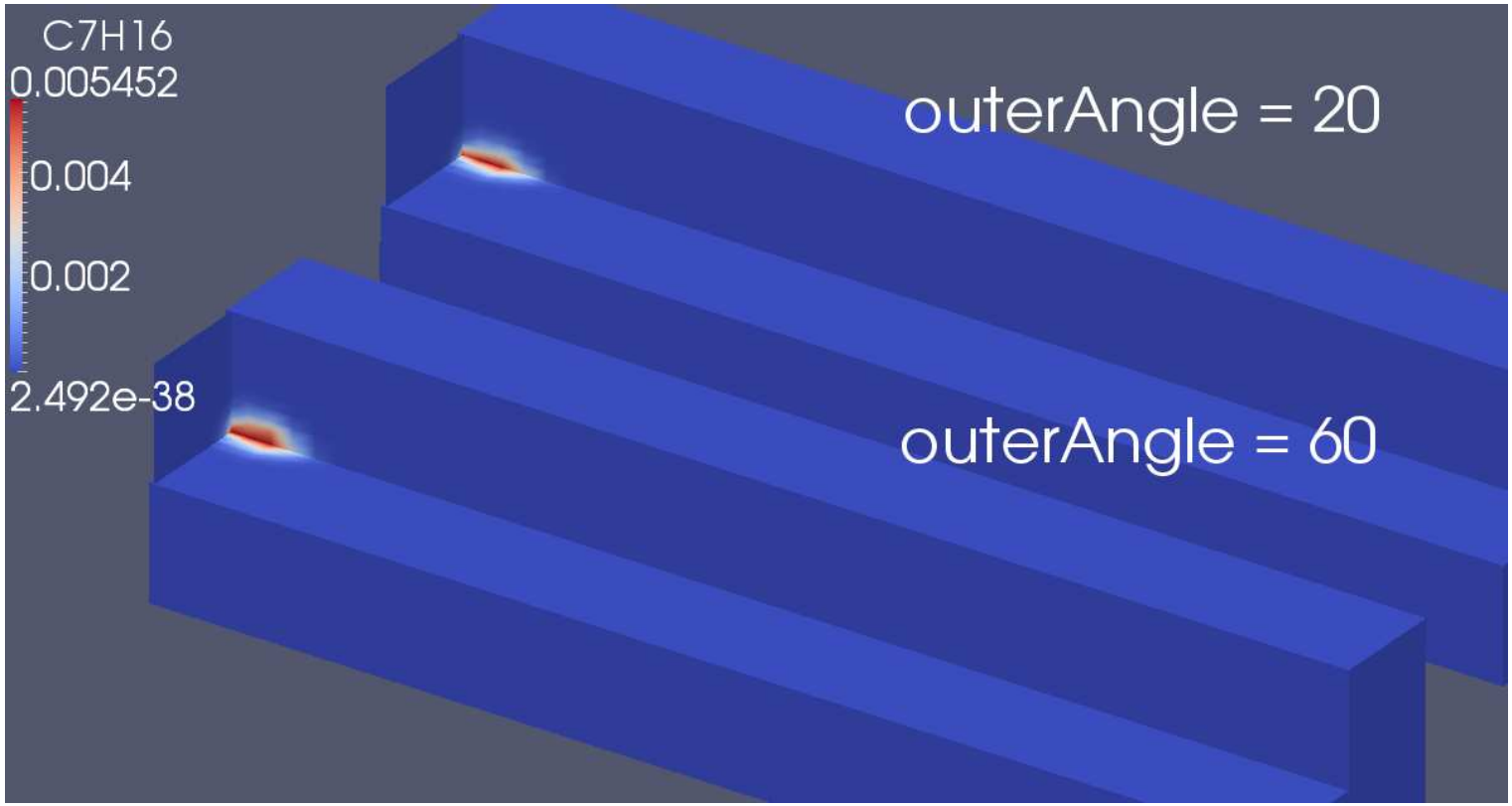
unitInjector and hollowConeInjector

- Injector type and injector model needed.
- unitInjector and hollowConeInjector concerned here.
- Injector type unitInjector.
  - Position
  - Direction
  - Diameter
  - Cd
  - Mass
  - Specie mass fraction
  - Mass flow
  - Temperature
- Injector model hollowConeInjector
  - Parcel size distribution
  - Inner and outer cone angle

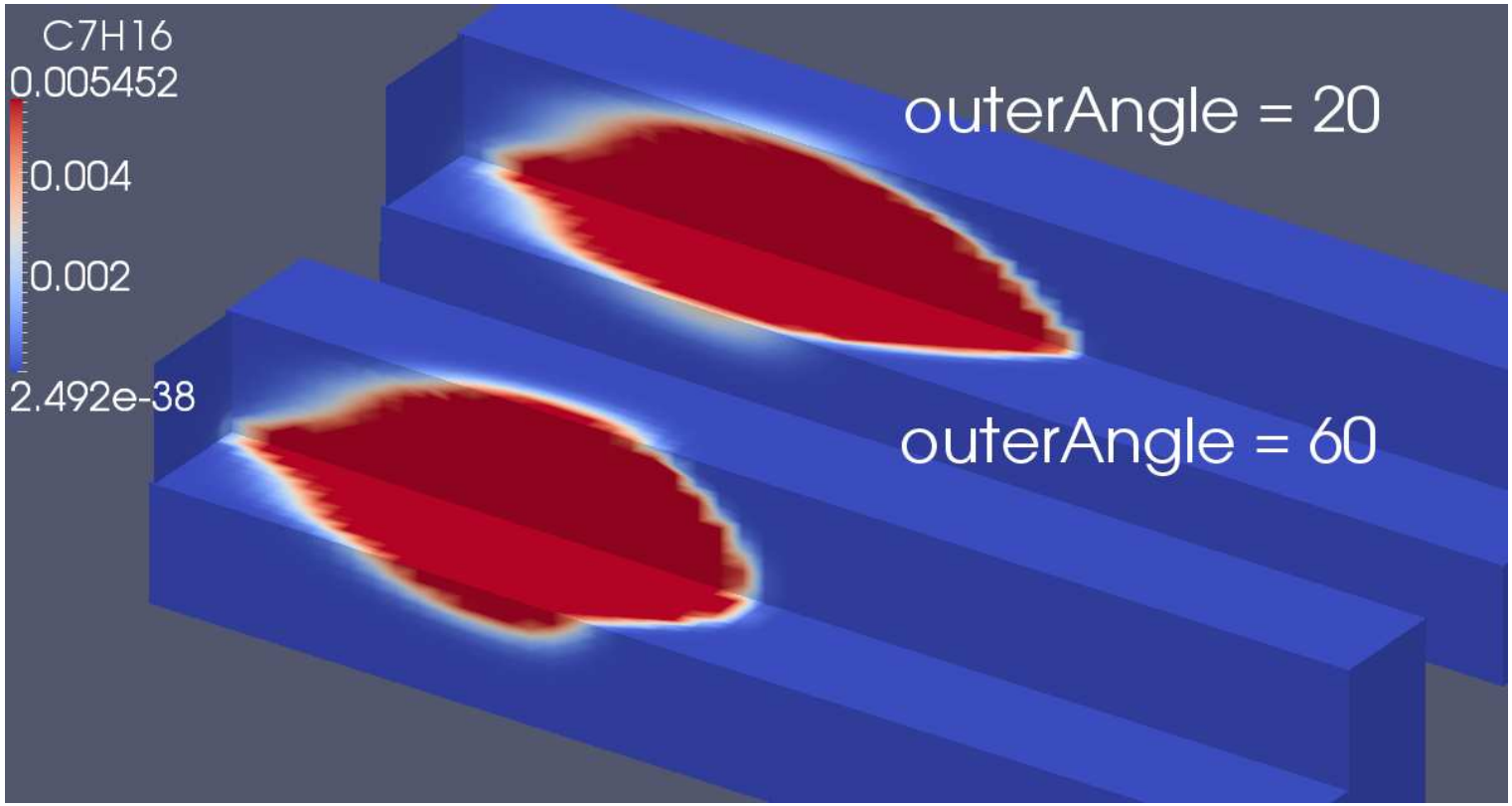
Basic injection visualisation



Example from aachenBomb case



Example from aachenBomb case



Tutorial - compilation settings

Create a directory for the new class

```
mkdir $WM_PROJECT_USER_DIR/applications/solvers/injectorSolidParticleFoam
cd $WM_PROJECT_USER_DIR/applications/solvers/injectorSolidParticleFoam
```

Copy the needed files to the new directory

```
cp -r $FOAM_SRC/lagrangian/solidParticle/* .
cp $FOAM_SOLVERS/incompressible/simpleFoam/* .
```

Rename the files corresponding to the new class name

```
 mv solidParticle.C injectorSolidParticle.C
mv solidParticle.H injectorSolidParticle.H
 mv solidParticleI.H injectorSolidParticleI.H
mv solidParticleIO.C injectorSolidParticleIO.C
 mv solidParticleCloud.C injectorSolidParticleCloud.C
mv solidParticleCloud.H injectorSolidParticleCloud.H
 mv solidParticleCloudI.H injectorSolidParticleCloudI.H
mv simpleFoam.C injectorSolidParticleFoam.C
```

Tutorial - compilation settings

Make sure that the code within the files also agrees with the new names.

```
sed -i s/solid/injectorSolid/g injectorSolidParticle.C
 sed -i s/solid/injectorSolid/g injectorSolidParticle.H
sed -i s/solid/injectorSolid/g injectorSolidParticleI.H
 sed -i s/solid/injectorSolid/g injectorSolidParticleIO.C
sed -i s/solid/injectorSolid/g injectorSolidParticleCloud.C
 sed -i s/solid/injectorSolid/g injectorSolidParticleCloud.H
sed -i s/solid/injectorSolid/g injectorSolidParticleCloudI.H
 sed -i s/simpleFoam/injectorSolidParticleFoam/g injectorSolidParticleFoam.C
```

Edit `Make/files`, make sure it looks like:

```
injectorSolidParticle.C
injectorSolidParticleIO.C
injectorSolidParticleCloud.C
injectorSolidParticleFoam.C

EXE = $(FOAM_USER_APPBIN)/injectorSolidParticleFoam
```

Tutorial - compilation settings

Also change `Make/options` to

```
EXE_INC = \
    -I$(LIB_SRC)/turbulenceModels \
    -I$(LIB_SRC)/turbulenceModels/incompressible/RAS/RASModel \
    -I$(LIB_SRC)/transportModels \
    -I$(LIB_SRC)/transportModels/incompressible/singlePhaseTransportModel \
    -I$(LIB_SRC)/finiteVolume/lnInclude  \
    -I$(LIB_SRC)/lagrangian/basic/lnInclude

EXE_LIBS = \
    -lincompressibleRASModels \
    -lincompressibleTransportModels \
    -lfiniteVolume  \
    -llagrangian \
    -lincompressibleTurbulenceModel
```

Tutorial - injectorSolidParticleFoam.C

Among the include statements, make sure to include

```
#include "injectorSolidParticleCloud.H"
```

Just after `int main(int argc, char *argv[])`, state

```
#include "readGravitationalAcceleration.H"
```

Directly after this section insert

```
injectorSolidParticleCloud particles(mesh);
```

Just before `runTime.write();`, add

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

Tutorial -injectorSolidParticleCloud.C

Insert beneath `Cloud<injectorSolidParticle>::move(td);`

```
// Injector 1
//Set injection position (z=0 if 2d)
scalar posy=0.015;
scalar posz=0;
scalar posx=-0.0203;
vector pos = vector(posx,posy,posz);
//Set initial velocity vector
vector vel=vector(0,0,0);
//Particle diameter
scalar d = 1e-3;
// Find cell at specified injection position and add particle here
label cellI=mesh_.findCell(pos);
if(cellI>=0) {
injectorSolidParticle* ptr= new injectorSolidParticle(*this,pos,cellI,d,vel);
Cloud<injectorSolidParticle>::addParticle(ptr);
}
```

Tutorial -createFields.H

At the top of `createFields.H`.

```
 Info<< "\nReading transportProperties\n" << endl;
    IOdictionary transportProperties
    (
        IOobject
        (
            "transportProperties",
            runTime.constant(),
            mesh,
            IOobject::MUST_READ,
            IOobject::NO_WRITE,
            false
        )
    );

dimensionedScalar rhotP(transportProperties.lookup("rho"));

volScalarField rho
            (
```

```
                    IOobject
                    (
                        "rho",
                        runTime.timeName(),
                        mesh,
                        IOobject::NO_READ,
                        IOobject::NO_WRITE
                    ),
                    mesh,
                    dimensionedScalar("rho", rhotP.dimensions(), rhotP.value())
                )
;
```

In the class directory do

```
wclean
wmake
```

Tutorial - example case

Implement the new class on the pitzDaily case.

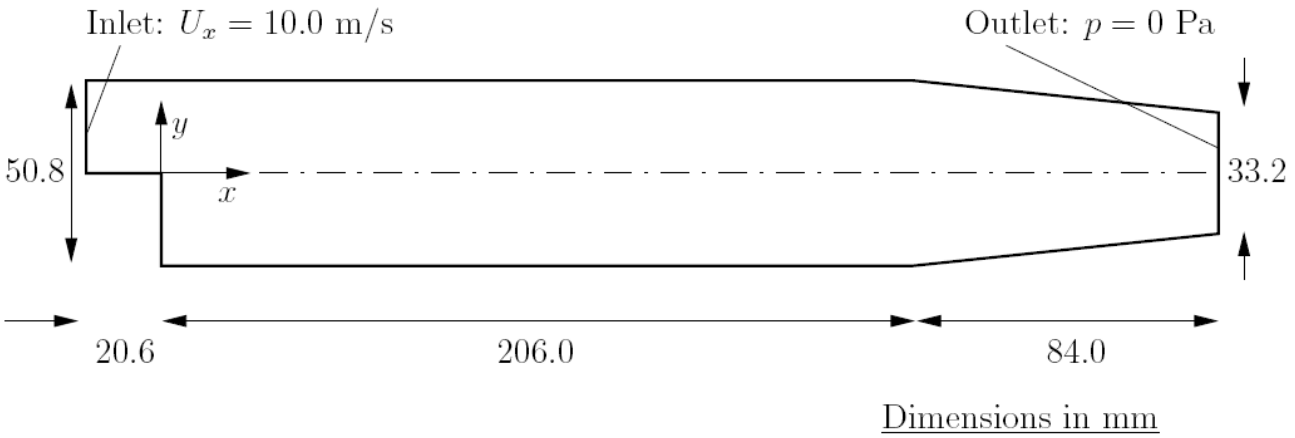


Figure 1: Geometry of the pitzDaily tutorial case.

Copy and rename the case.

```
cd $FOAM_RUN
cp -r $FOAM_TUTORIALS/incompressible/simpleFoam/pitzDaily ./pitzDailyInject
cd pitzDailyInject
```

Tutorial - example case

Solve the flow

```
blockMesh
simpleFoam
```

- Copy the folder `lagrangian` to `./1000`
- Copy the file `particleProperties` to `./constant`
- Copy the file `g` to `.constant`.

Edit `system/controlDict` and change the following parameters as

```
startTime       1000;
endTime         1000.15;
deltaT          0.003;
writeInterval   1;
timePrecision   8;
```

Tutorial - example case

Edit `/constant/transportProperties` and after
`nu nu [ 0 2 -1 0 0 0 0 ] 1e-05;` insert

```
rho rho [ 1 -3 0 0 0 0 0 ] 1;
```

Run, convert and preprocess by

```
injectorSolidParticleFoam
foamToVTK
paraview
```

Tutorial - two injectors

```
cd $WM_PROJECT_USER_DIR/applications/solvers/injectorSolidParticleFoam
```

In `injectorSolidParticleCloud.C` add

```
// Injector 2
//Set injection position (z=0 if 2d)
scalar posx2=0.15;
scalar posz2=0;
scalar posy2=-0.0243;
vector pos2 = vector(posx2,posy2,posz2);
//Set initial velocity vector
vector vel2=vector(0,0,0);
//Particle diameter
scalar d2 = 1e-3;
label cellI2=mesh_.findCell(pos);
if(cellI2>=0) {
injectorSolidParticle* ptr= new
injectorSolidParticle(*this,pos2,cellI2,d2,vel2);
Cloud<injectorSolidParticle>::addParticle(ptr);}
```

Tutorial - two injectors

Recompile

```
wclean
wmake
```

Go to the case directory again

```
cd $FOAM_RUN/pitzDailyInject
```

Edit `.system/controlDict` and change the end time to

```
endTime         1000.42;
```

Before rerunning the case, delete the previous particle track results

```
rm -rf 1000.*
rm -rf VTK
```

Solve with the recompiled solver, convert and postprocess

```
injectorSolidParticleFoam
foamToVTK
paraview
```

Tutorial - evenly distributed injector

```
cd $WM_PROJECT_USER_DIR/applications/solvers/injectorSolidParticleFoam
```

In `injectorSolidParticleCloud.C` scroll down to the section where the injector is defined. Remove entirely the second injector and before `//Injector 1` insert

```
for (label i=0; i<=25; i++) {
```

Don't forget to end the loop with `}`
Change the y-pos to

```
scalar posy=-0.0248+0.002*i;
```

Set the x-pos as

```
scalar posx=0.1;
```

Recompile the class. Solve, convert and preprocess the case as earlier (don't forget to remove earlier results).