Tutorial Lagrangian Particle Tracking

Aurélia Vallier

Division of Fluid Mechanics
Department of Energy Sciences
Lund Institute of Technology, Lund Sweden

OUTLINE

- **Introduction to Lagrangian particle tracking**
- **LPT in OpenFOAM**
- **Improvement**
- **Test case**

LPT

Euler frame: the flow field
(U, p, density, viscosity...)

Lagrangian frame: the particles
(velocity, position, diameter, density,...)

Eulerian frame

$$\nabla \cdot \mathbf{U} = 0$$

$$\rho \frac{\partial \mathbf{U}}{\partial t} + \rho(\mathbf{U} \cdot \nabla \mathbf{U}) = -\nabla p + \mu \nabla^2 \mathbf{U} + \rho g - \sum_N \mathbf{f_P}$$

```
fvVectorMatrix UEqn
    (
        fvm::ddt(rho, U)
      + fvm::div(phi, U)
      + turbulence->divDevRhoReff(U)
     ==
        rho*g
      + dieselSpray.momentumSource()
    );

    if (momentumPredictor)
    {
        solve(UEqn == -fvc::grad(p));
    }
```

Lagrangian frame

$$\mathbf{f_P} = \frac{m_P}{\Delta V} \frac{(\mathbf{U_p})_{t_{out}} - (\mathbf{U_p})_{t_{in}}}{t_{out} - t_{in}}$$

```
vector oMom = m()*U();
// update the parcel properties (U, T, D)
updateParcelProperties(dt,sDB,celli,face());
vector nMom = m()*U();
// Update the Spray Source Terms
sDB.sms()[celli]   += oMom - nMom;
```

```
tsource().internalField() = sms_/runTime_.deltaT().value()/mesh_.V();
```

Lagrangian frame

$$m_p \frac{d\mathbf{U_P}}{dt} = \mathbf{F_D} + m_p \mathbf{g} = -m_p \frac{\mathbf{U_P} - \mathbf{U}}{\tau_P} + m_p \mathbf{g}$$

$$\rightarrow \mathbf{U_P^{\Delta t+1}} = \frac{\mathbf{U_P^{\Delta t}} + \mathbf{U_{@P}} \frac{dt}{\tau_p} + \mathbf{g}dt}{1 + \frac{dt}{\tau_p}}$$

```
vector Up = sDB.UInterpolator().interpolate(position(), celli, facei)+ Uturb();
scalar timeRatio = dt/tauMomentum;
U() = (U() + timeRatio*Up + sDB.g()*dt)/(1.0 + timeRatio);
```


Lagrangian frame

$$\frac{d\mathbf{x_P}}{dt} = \mathbf{U_P}$$

```
// set the lagrangian time-step
  scalar dt = min(dtMax, tEnd);
// Track and adjust the time step if the trajectory is not completed
  dt *= trackToFace(position() + dt*U_, sDB);
// Decrement the end-time acording to how much time the track took
  tEnd -= dt;
  // Set the current time-step fraction.
   stepFraction() = 1.0 - tEnd/deltaT;
```


LPT in OpenFOAM

DieselFoam : solver for diesel spray and combustion
- evaporation, collision...
- parcel (computational particles)

IcoLagrangianFoam : simplified version of dieselFoam,
- no evaporation, no collision...
- wrong drag model

SolidParticle : new library for one way coupling LPT

Improvement

- Injection of several particles per time step

- Volume fraction of particles (volScalarField)
  → we don't need to save the properties of each particle

Injector of particles

```
void Foam::solidParticleCloud::inject(solidParticle::trackData &td) {
 for (label nbP=1; nbP<=td.spc().nbInjByDt(); nbP++) {
    vector tmp=(random().vector01()- vector(0.5,0.5,0.5))*2;
    vector center=td.spc().center();
    vector r0=td.spc().r0();
    scalar posx=tmp.x()*r0.x();
    scalar posy=tmp.y()*r0.y();
    scalar posz=tmp.z()*r0.z();
    vector pos=center+vector(posx,posy,posz);
    vector tmpv=
 vector(random().GaussNormal(),random().GaussNormal(),random().GaussNormal())/sqrt(3.);
    vector vel=tmpv*td.spc().velprime()+td.spc().vel();
    label cellI=mesh_.findCell(pos);
    if(cellI>=0)
     {
     solidParticle* ptr= new solidParticle(*this,pos,cellI,td.spc().d(),vel);
     Cloud<solidParticle>::addParticle(ptr);
     }
  }
}
```


Volume fraction of particles

$$volfrac = nbP.V_p/\Delta V$$

In soliParticle.C, at the end of the function `move`

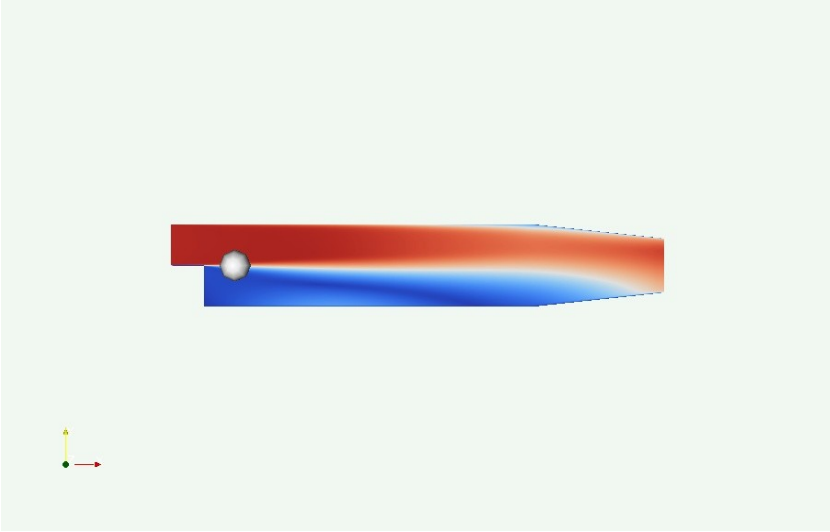
```
if (td.keepParticle)
 {
  label cellnew = cell();
  td.spc().nbPVp()[cellnew]+=(4/3*3.14*pow(d_,3)/8);
 }
```

In solidParticleFoam.C, call the function volFrac()

```
particles.move(g);
volfrac= particles.volFrac();                    //added
runTime.write();
```

The function volFrac() returns `nbPVp_/mesh_.V();`

Test case



1- Copy and compile the solver
mySolidParticleSimpleFoam

2- Copy the tutorial pitzDaily

3- Add the file 0/volFracorg
(cp 0/p 0/volFracorg and change dimension to [0 0 0...])

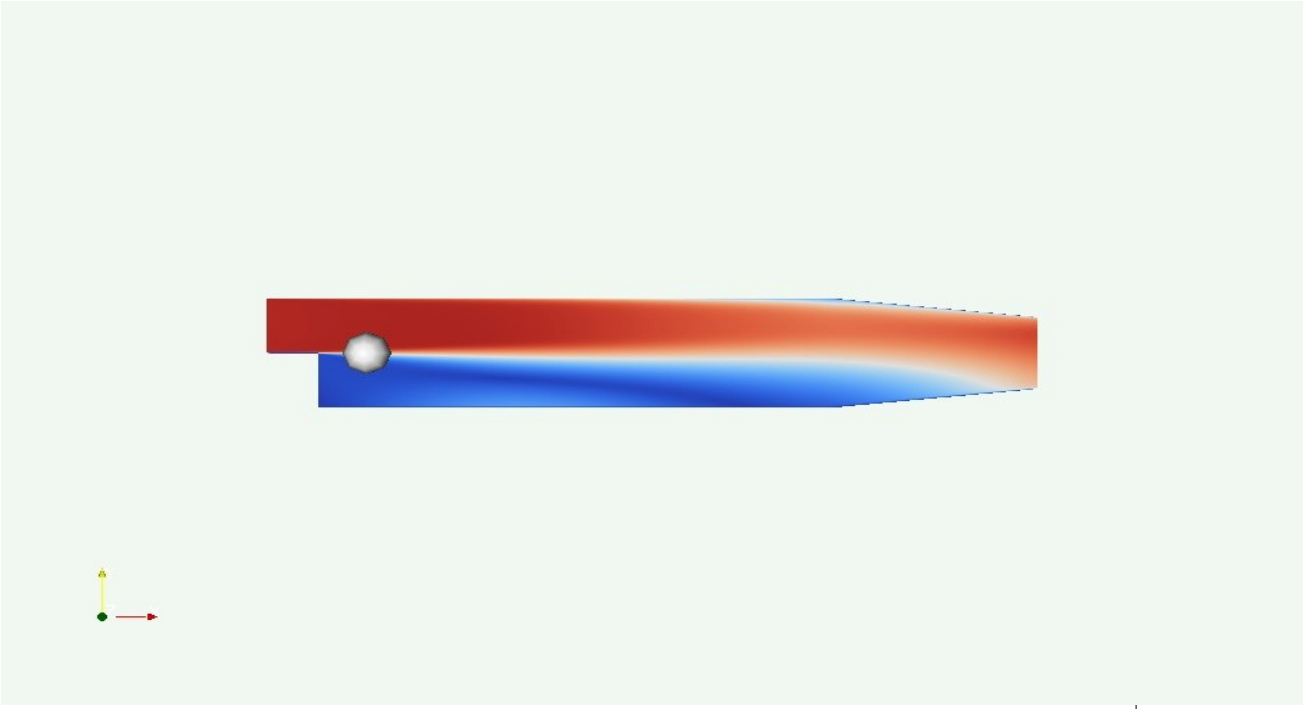
4- Add the file constant/g

```
dimensions      [0 1 -2 0 0 0 0];
value           ( 0 0 0 );
```

5- Add density in the file constant/transportProperties

```
rho             rho [ 1 -3 0 0 0 0 0 ] 1000;
```

6- Add the file constant/particleProperties



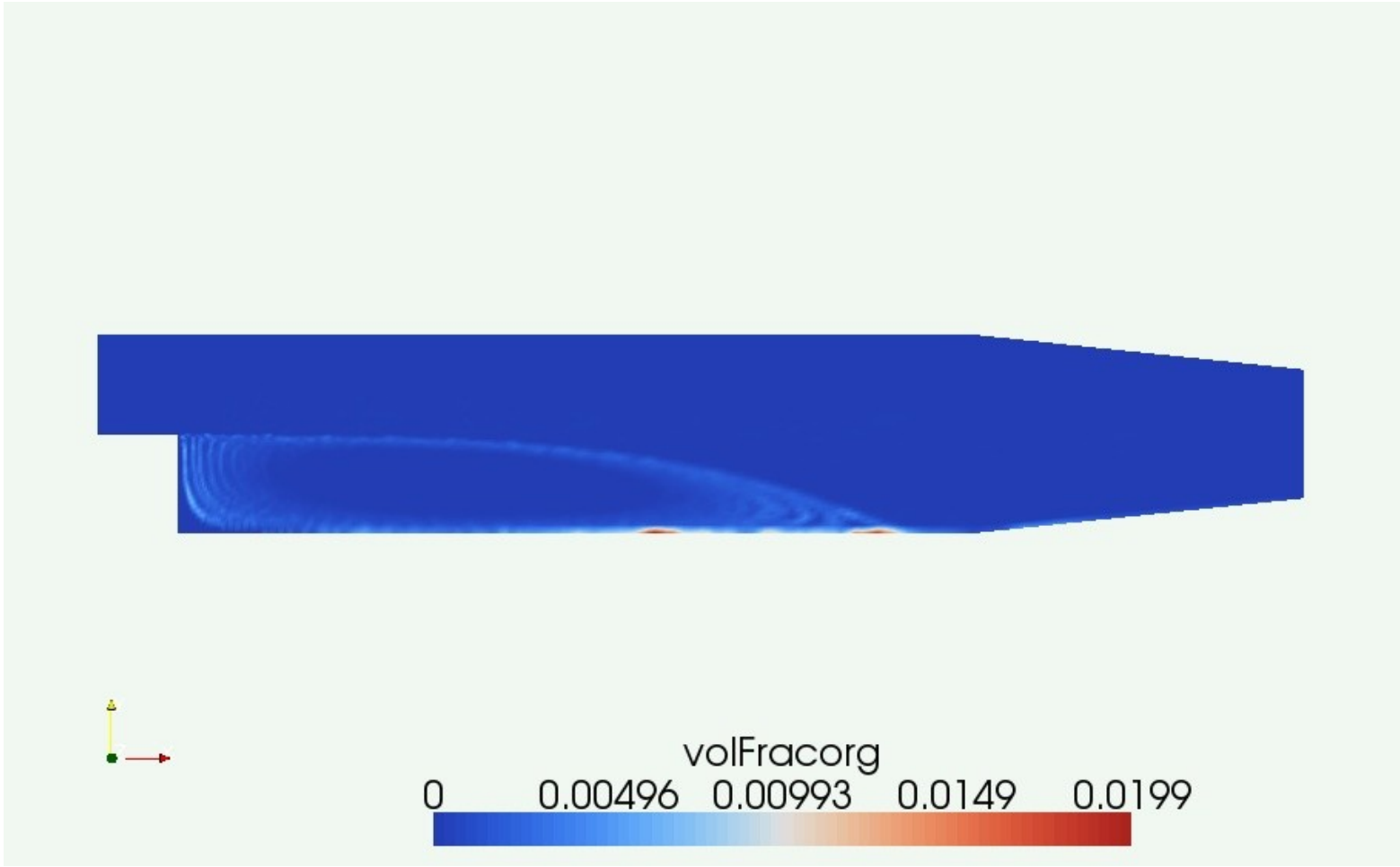
delta x=0.311
max(Ux)=10.1
→ a particle can go through in 0.03s

constant/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;
nbInjByDt nbInjByDt [ 0  0  0  0  0  0  0] 500;
center center [ 0  1  0  0  0  0  0] (0.02 0.0 0.0);
r0 r0 [ 0  1  0  0  0  0  0] (0.01 0.01 0.0);
d d [ 0  1  0  0  0  0  0] 0.01; //5e-5;
vel vel [ 0  1 -1   0  0  0  0] (0  0 0);
velprime velprime [ 0  0  0   0  0  0  0] 0 ;
tInjStart tInjStart [ 1 0   0  0  0  0  0] 1e-5;
tInjEnd tInjEnd [ 1 0   0  0  0  0  0] 5000;
```

system/controlDict

```
startFrom       startTime;

startTime       1000;

stopAt          endTime;

endTime         1001;

deltaT          0.003;

writeControl    timeStep;

writeInterval   1;
```



Distribution of the volume fraction of particles at t=1001

Thank you