Particles simulation in incompressible flow using regularized hydrodynamics equations in OpenFOAM v1912

Instructors: Tatiana V. Stenina, Aleksadr V. Ivanov
Authors: T.V. Stenina, A.V. Ivanov, M.V. Kraposhin, I.N. Sibgatullin
Training level: Intermediate
Session type: Lecture with examples
Software stack: OpenFOAM v1912

`https://github.com/unicfdlab`

Plan of training course

Before we start...

If you are listener of the course, you should:

- have basic knowledge of OpenFOAM
- know basic commands for Linux terminal
- **have preinstalled OpenFOAM v1912 on your laptop OR ability to boot from USB**
- have Internet connection

Training course materials

- Course location:
  `https://github.com/unicfdlab/TrainingTracks`
- Folder QHDFoam-OFv1912

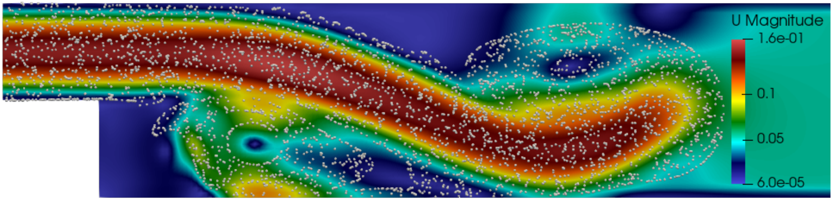
Folder	**Description**
cases	Cases that will be used to demonstrate QHD solver's work during the track
materials	This presentation and other materials that were used in this course

Full version of the solver is available at
`https://github.com/unicfdlab/QGDsolver`

Key points of training course

The following points will be considered:

- a description of the basic principles of the solver (particlesQHDFoam);
- setting the input parameters (initial and boundary conditions);
- setting particles cloud properties
- running tutorials for OpenFOAM v1912.



Part I
Theoretical part

particlesQHDFoam: how it works

Governing equations of Eulerian phase

- Continuity equation:

$$\nabla \cdot \left(\vec{U} - \vec{W}\right) = 0, \quad \vec{W} = \tau \left((\vec{U} \cdot \nabla)\vec{U} + \frac{1}{\rho_0}\nabla\tilde{p} - \beta\vec{g}\tilde{T}) \right)$$

- Momentum equation:

$$\frac{\partial \vec{U}}{\partial t} + \nabla \cdot \left(\left(\vec{U} - \vec{W}\right) \otimes \vec{U} \right) + \frac{1}{\rho_0}\nabla\tilde{p} = \frac{1}{\rho_0}\nabla \cdot \hat{\Pi} + \beta\vec{g}\tilde{T}$$

- Scalar (temperature) transport equation:

$$\frac{\partial T}{\partial t} + \nabla \cdot \left(\left(\vec{U} - \vec{W}\right) T \right) - \nabla \cdot \left(\tau\vec{U} \left(\vec{U} \cdot \nabla\right) T \right) - \nabla \cdot \left(\frac{\mu}{\rho_0 Pr}\nabla T \right) = 0$$

- Incompressible EoS and regularized stress tensor:

$$\rho = \rho_0 \left(1 + \beta\tilde{T}\right), \quad \hat{\Pi} = \rho\vec{U}\otimes\vec{W} + \hat{\Pi}_{NS}, \quad \hat{\Pi}_{NS} = \mu \left[(\nabla \otimes \vec{U}) + (\nabla \otimes \vec{U})^T \right]$$

Governing equations of Lagrangian-Particle-Tracking

Calculation of particles motion:

$$\frac{d\vec{x}}{dt} = \vec{U}_p, \quad m_p \frac{d\vec{U}_p}{dt} = \sum_i \vec{F}_i$$

We could consider different forces acting on the particle, e.g., drag, gravitational and buoyancy forces, pressure forces

$$m_p \frac{d\vec{U}_p}{dt} = \sum_i \vec{F}_i = \vec{F}_D + \vec{F}_G + \vec{F}_P + \ldots$$

particlesQHDFoam = QHDFoam + particles

Purpose of the solver

To solve equation of incompressible fluid with equations of particles clouds motion and heat transfer

Changes to the foundation solvers

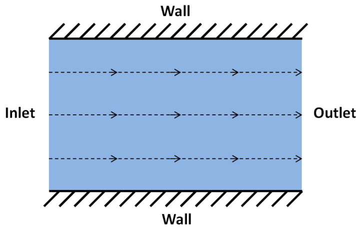
The Lagrangian particle tracking is accounted in new solver through changes in main () procedure.
Source code:

```
#include "basicThermoCloud.H"

...

parcels.evolve();
```

Boundary conditions

Types of boundary conditions:

- wall
- inlet
- outlet



Within the framework of regularized conditions, these boundary conditions are described mathematically on the following slides.

Keywords in OpenFOAM

There are prepared keywords for these boundary conditions in OpenFOAM.
For example, we want to set a specific pressure gradient for the wall:

```
wall
{
    type     qhdFlux;
    value    $internalField;
}
```

Or we want to set a fixed velocity for the inlet:

```
inlet
{
    type     fixedValue;
    value    uniform (1 0 0);
}
```

Keywords in OpenFOAM

slip	This boundary condition provides a slip constraint: $\vec{n}\vec{U}_n = 0$, $\frac{\partial U_\tau}{\partial n}\tau = 0$
noSlip	This boundary condition fixes the velocity to zero at walls, similar to fixedValue $= 0$
fixedValue	This boundary condition provides a fixed value constraint, and is the base class for a number of other boundary conditions
zeroGradient	This boundary condition applies a zero-gradient in normal direction condition
qhdFlux	This boundary condition is a specific condition for ∇p: $\vec{n}\cdot\vec{U} - \tau(\vec{U}\cdot\nabla\vec{U} - \frac{1}{\rho_0}\nabla p)\cdot\vec{n} = 0$
totalPressure	This boundary condition provides a total pressure condition

Empty

This boundary condition provides an 'empty' condition for reduced dimensions cases, i.e. 1- and 2-D geometries. Apply this condition to patches whose normal is aligned to geometric directions which are not involved in simulation.

Regularization parameter τ

Value of τ ($[\tau] = s$) coefficient is selected to be equal or less than some characteristic hydrodynamic time using characteristic velocity magnitude U, kinematic viscosity ν, grid step Δx or other parameters:

- For simple cases, τ could be estimated from dimensionless numbers (like Re, Gr or others): $\tau \approx \tau_0 Re^{-1}$, $\tau \approx \tau_0 Gr^{-1}$;
- Through the max CFL $Co^{max} = |\vec{U}|^{max} \Delta t / \Delta x$ number: $\tau \approx \frac{Co^{max} \Delta x}{|\vec{U}|^{max}} C_\tau$, where C_τ is a constant less than 1

Three stability criteria are used:

1. $\Delta t < C_\tau^{-1} \tau$, where $C_\tau \leq \frac{1}{2}$. In some cases C_τ could be set to 0.75
2. $Co = U \frac{\Delta t}{\Delta x} < Co^{max}$. The Co^{max} is usually about $0.1 - 0.2$ in most cases
3. $\tau |\vec{U}| \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}$

Regularization parameter τ

Let us determine the value of the regularization coefficient τ using the Reynolds number:

$$\tau = \tau_0 Re^{-1}, \quad Re = \frac{Ul}{\nu}.$$

If $\tau_0 = \frac{l}{U}$ then

$$\tau = \frac{\nu}{U^2},$$

where U is a value of characteristic speed, l is a characteristic size. The time integration step should not exceed τ, and is often chosen as:

$$\triangle t = \frac{\tau}{2}.$$

Part III
Practical part

How to set up cases

How to install QGDSolver

This is for OpenFOAM+ v1912, for other OpenFOAM version, different branches should be used.

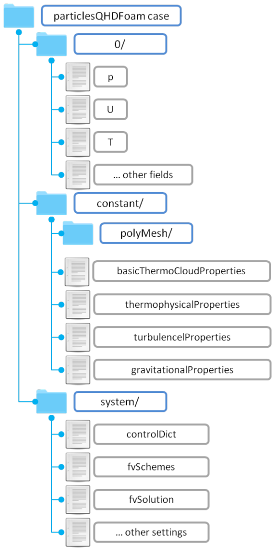
- Download QGDSolver directly from `https://github.com/unicfdlab/QGDsolver/tree/digitef-dev-1912`
  you can try short link: `https://clck.ru/QgNJy`
  or using git clone:

  ```
  git clone https://github.com/unicfdlab/QGDsolver.git
  git checkout digitef-dev-1912
  ```

- Install QGDSolver:

  ```
  ./Allwmake
  ```

particlesQHDFoam case structure



Initial and boundary conditions

Initial conditions are set in the folder "0". Three fields are mandatory to start a simulation: pressure "p", velocity "U" and temperature "T"

Fluid and cloud properties

Thermophysical fluid properties are set in "thermophysicalPropertis" dictionary. By default the turbulence modelling is turned off in the "turbulenceProperties" dictionary. Value and direction of gravity bulk field is set in "gravitationalProperties". Particle properties and settings described in the "basicThermoCloudProperties"

Numerical schemes

Numerical schemes settings are stored in "fvSchmes" and "fvSolution", time advancement control is in "controlDict"

Stages of solution

See folder QHDFoam-OFv1912

- prepare new case folder:

```
cp cases/backwardStep cases/backwardStepTraining -r
```

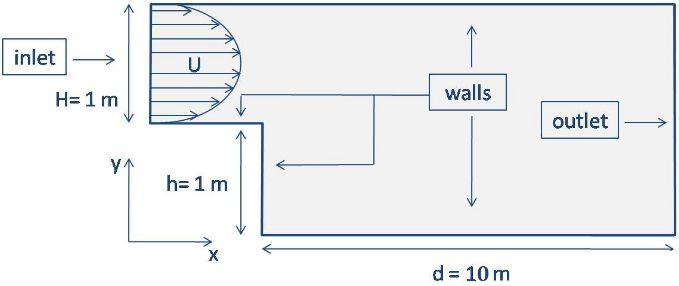
- mesh generation
- boundary conditions setup
- physical properties setup
- τ selection
- advancement time settings
- numerical schemes settings

backwardStepTraining

Case set up

See folder QHDFoam-OFv1912. The case is in the folder cases/

- 2D case (backward step)
- fixed profile for velocity at the inlet
- fluid: $\rho = 1000$, $\mu = 1$
- stable flow: $Re = 1 \div 1000$, $Re = \frac{H \cdot U}{\nu}$



Mesh generation

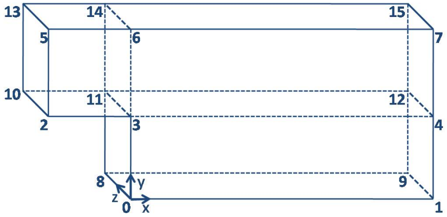
See file system/*blockMeshDict*

Set scale:

```
convertToMeters 1;
```

Set vertices:

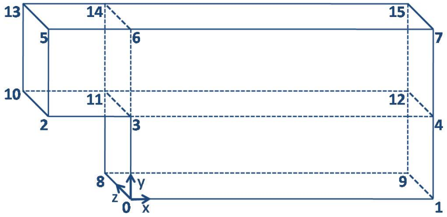
```
vertices
(
        (0 0 0)      // 0
        (10 0 0)     // 1
        (-1 1 0)     // 2
        (0 1 0)      // 3
        (10 1 0)     // 4
        (-1 2 0)     // 5
        (0 2 0)      // 6
        (10 2 0)     // 7
```



Mesh generation

Set vertices:

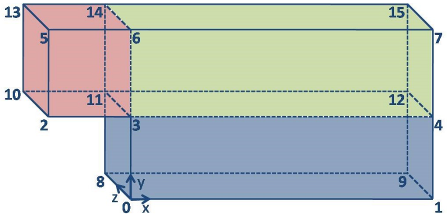
```
        (0 0 0.05)      // 8
        (10 0 0.05)     // 9
        (-1 1 0.05)     // 10
        (0 1 0.05)      // 11
        (10 1 0.05)     // 12
        (-1 2 0.05)     // 13
        (0 2 0.05)      // 14
        (10 2 0.05)     // 15
   );
```



Mesh generation

Create three boxes:

```
blocks
(
    hex (0 1 4 3  8  9  12 11) (200 20 1) simpleGrading (1 1 1)
    hex (2 3 6 5 10 11 14 13) (20 20 1) simpleGrading (1 1 1)
    hex (3 4 7 6 11 12 15 14) (200 20 1) simpleGrading (1 1 1)
);
```



Mesh generation

Describe boundaries:

```
boundary
(
    inlet
    {
            type patch;
            faces ( (2 5 13 10) );
    }
    outlet
    {
            type patch;
            faces ( (1 4 12 9) (4 7 15 12) );
    }
```

Mesh generation

```
    walls
    {
        type wall;
        faces ( (0 1 9 8) (0 3 11 8) (2 3 11 10)
                      (5 6 14 13) (6 7 15 14) );
    }
    symwall
    {
        type empty;
        faces ( (0 1 4 3) (2 3 6 5) (3 4 7 6)
                  (8 9 12 11) (10 11 14 13) (11 12 15 14) );
    }
);
```

Command: blockMesh

Boundary conditions

See folder 0/

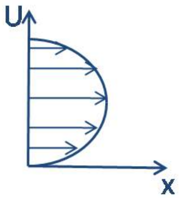
Name	U, m/s	p, Pa	T, K
inlet	fixedProfile	qhdFlux	fixedValue 300
outlet	zeroGradient	totalPressure	zeroGradient
walls	noSlip	qhdFlux	zeroGradient
symwall	empty	empty	empty

fixedProfile

See file 0/*U*

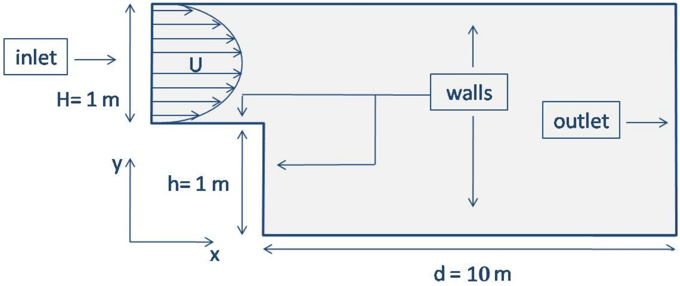
Set fixed Poiseuille profile for velocity at the inlet:

```
type        fixedProfile;
profile     table
(
        (0 (0 0 0))
        (0.1 (0.054 0 0))
        (0.2 (0.096 0 0))
        (0.3 (0.126 0 0))
        (0.4 (0.144 0 0))
        (0.5 (0.15 0 0))
        (0.6 (0.144 0 0))
        (0.7 (0.126 0 0))
        (0.8 (0.096 0 0))
        (0.9 (0.054 0 0))
        (1 (0 0 0))
)
```



fixedProfile

```
direction   (0 1 0); // direction in which the speed will change
origin      1; // the step to step back in "direction"from (0 0 0)
value       uniform (0 0 0);
```



totalPressure

Set total pressure at the outlet:

```
outlet
{
  type      totalPressure;
  p0        $internalField; // $p0 = p + 0.5\rho * \vec{U} \cdot \vec{U} = const,$
  rho       rho;
  value     $internalField;
}
```

Physical properties

See folder constant/
See file *thermophysicalProperties*
Set density:

```
equationOfState
{
    rho 1000;
}
```

Set dinamic viscosity:

```
transport
{
    mu      1;
    Pr      1;
    beta    0; // buoyancy factor
}
```

τ calculation

See file *thermophysicalProperties*

```
QGD
{
  implicitDiffusion true; // approximation of viscous terms
  QGDCoeffs constTau; // choice of calculation option for τ
  constTauDict
  {
      Tau 0.1; // $\tau \sim \tau_0 = \frac{\nu}{U^2} = \frac{10^{-3}}{10^{-2}} = 0.1$
  }
  pRefCell 0;
  pRefValue 0;
}
```

In a closed incompressible system such as the cavity, pressure is relative: it is the pressure range that matters not the absolute values. In cases such as this, the solver sets a reference level by pRefValue in cell pRefCell. In this example both are set to 0.

QGDCoeffs

We have different methods for τ calculation:

- H2bynuQHD
- HbyUQHD
- T0byGr
- **constTau**

H2bynuQHD

$$\tau = \frac{h^2}{\nu},$$

where h is a grid step, ν is a kinematic viscosity.

```
QGD
{
    implicitDiffusion true;
    QGDCoeffs H2bynuQHD;
    pRefCell 0;
    pRefValue 0;
}
```

HbyUQHD

$$\tau = \frac{h}{U},$$

where h is a grid step, U is a magnitude of characteristic velocity.

```
QGD
{
    implicitDiffusion true;
    QGDCoeffs HbyUQHD;
    HbyUQHDDict
    {
        UQHD 1; // value of characteristic speed
    }
    pRefCell 0;
    pRefValue 0;
}
```

T0byGr

$$\tau = \frac{T_0}{Gr},$$

where T_0 is a bulk temperature, Gr is a Grashof number.

$$Gr = \frac{g\beta(T_s - T_0)L^3}{\nu^2},$$

where g is acceleration of gravity, β is the coefficient of thermal expansion, $T_s$is the surface temperature, L is the characteristic length, ν is a kinematic viscosity.

```
QGD
{
    implicitDiffusion true;
    QGDCoeffs T0byGr;
    T0byGrDict
    {
        Gr 100; T0 300; //T0 = τ0
    }
    pRefCell 0;
    pRefValue 0;
}
```

constTau

$$\tau = constant$$

```
QGD
{
    implicitDiffusion true;
    QGDCoeffs constTau;
    constTauDict
    {
          Tau 1e-3;
    }
    pRefCell 0;
    pRefValue 0;
}
```

Physical properties

See file *gravitationalProperties*
Set acceleration of gravity:

```
g    g    [0 1 -2 0 0 0 0]    (0 0 0); // $\vec{g} = (0\ 0\ 0)$
```

See file *turbulenceProperties*
Set flow type:

```
simulationType    laminar; // uses no turbulence models
```

How to set up the particle cloud

See file *basicThermoCloudProperties*

This file consists of the following blocks:

- solution:
  how to solve equations with particles;
- constantProperties:
  Define the physical particle properties and relevant particle forces;
- injectionModels:
  how the particles fall into the computational domain.

How to set up the particle cloud

basicThermoCloudProperties::solution

```
solution
{
    active true; // Activate/de-activate the particles
    coupled true; // Enable/disable phase coupling
    transient yes; // Transient/steady-state solutionx
    cellValueSourceCorrection off; // Enable/disable correction
              of momentum transferred to the Eulerian phase
    maxCo 0.3; // max. Courant number
    interpolationSchemes // Choose interpolation schemes
    {
        rho cell;
        U cellPoint;
        thermo:mu cell;
        ...
    }
```

How to set up the particle cloud

basicThermoCloudProperties::solution

```
    integrationSchemes // integration schemes for the LPT
                                (Lagrangian Particle Tracking)
    {
        U Euler;
        T Euler;
    }
    sourceTerms // treatment of source terms
    {
        schemes
        {
            U explicit 1;
            h explicit 1;
        }
    }
}
```

How to set up the particle cloud

basicThermoCloudProperties::constantProperties

```
constantProperties
{
    rho0 7874; // Density
    T0 300; // Temperature
    Cp0 4500; // Heat capacity
    youngsModulus 1.3e5; // Young's module (elastic modulus)
    poissonsRatio 0.35; // Poisson's ratio
    subModels
    {
        particleForces
        {
            sphereDrag; // Drag force
            gravity; // Gravity/Buoyancy force
        }
    } }
```

How to set up the particle cloud

basicThermoCloudProperties:injectionModels

```
injectionModels
{
    model1
    {
      type patchInjection; // Injection model
      patch inlet; // injection patch name
      duration 100; // Total duration of injection
      parcelsPerSecond 100; // Injected parcels/particles per second
      massTotal 0;
      parcelBasisType fixed;
      flowRateProfile constant 1;
      nParticle 1;
      SOI 0; // Start-of-injection time (SOI)
      U0 (10 0 0); // Initial parcel/particle velocity (U_0)
```

How to set up the particle cloud

basicThermoCloudProperties:injectionModels

```
            sizeDistribution // Size distribution model
            {
              type fixedValue;
               fixedValueDistribution
               {
                      value 0.00007; // particle diameter
               }
            }
          }
      }
```

Advancement time settings

See file system/*controlDict* to create time settings:

- time step interval

```
deltaT    0.5e-3;
```

- write interval

```
writeInterval    1;
```

- CFL number and parameter C_τ^{-1} (any value less than 1)

```
writeControl           adjustableRunTime;
adjustableTimeStep     true;
maxCo                  0.5;
cTau                   0.3;
```

Advancement time settings

- Start time of calculations

```
startTime    0;
```

- End time of calculations

```
endTime    100;
```

Numerical schemes settings. Running

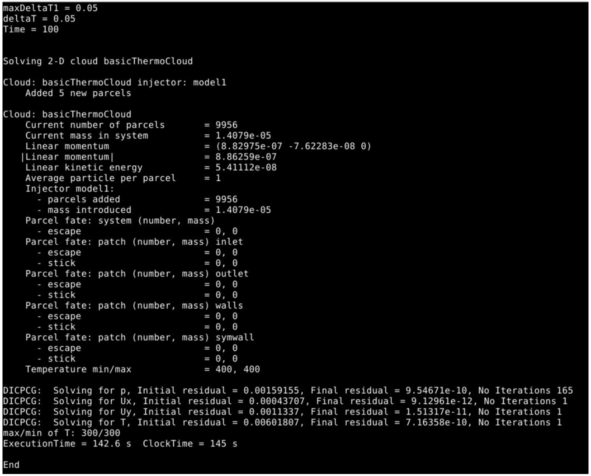
See file system/*fvSchemes* and system/*fvSolution*.
The user specifies the choice of finite volume schemes in the *fvSchemes* dictionary. In file *fvSchemes* you can see that we use only central difference scheme.
The specification of the linear equation solvers and tolerances and other algorithm controls is made in the *fvSolution* dictionary.

You can start application by ***particlesQHDFoam*** command.

Sequence of all commands is placed in the script file: ./**Allrun**.
Clean results: ./**Allclean**.

Results

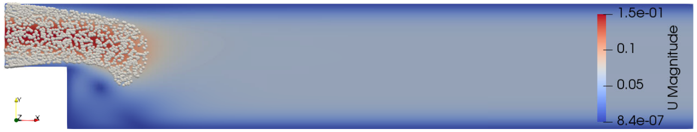
After the entering the command **particlesQHDFoam**, you will see on the screen:



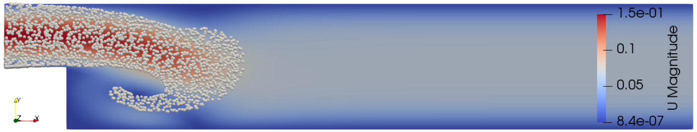
```
maxDeltaT1 = 0.05
deltaT = 0.05
Time = 100


Solving 2-D cloud basicThermoCloud

Cloud: basicThermoCloud injector: model1
    Added 5 new parcels

Cloud: basicThermoCloud
    Current number of parcels       = 9956
    Current mass in system          = 1.4079e-05
    Linear momentum                 = (8.82975e-07 -7.62283e-08 0)
   |Linear momentum|                = 8.86259e-07
    Linear kinetic energy           = 5.41112e-08
    Average particle per parcel     = 1
    Injector model1:
      - parcels added               = 9956
      - mass introduced             = 1.4079e-05
    Parcel fate: system (number, mass)
      - escape                      = 0, 0
    Parcel fate: patch (number, mass) inlet
      - escape                      = 0, 0
      - stick                       = 0, 0
    Parcel fate: patch (number, mass) outlet
      - escape                      = 0, 0
      - stick                       = 0, 0
    Parcel fate: patch (number, mass) walls
      - escape                      = 0, 0
      - stick                       = 0, 0
    Parcel fate: patch (number, mass) symwall
      - escape                      = 0, 0
      - stick                       = 0, 0
    Temperature min/max             = 400, 400

DICPCG:  Solving for p, Initial residual = 0.00159155, Final residual = 9.54671e-10, No Iterations 165
DICPCG:  Solving for Ux, Initial residual = 0.00043707, Final residual = 9.12961e-12, No Iterations 1
DICPCG:  Solving for Uy, Initial residual = 0.0011337, Final residual = 1.51317e-11, No Iterations 1
DICPCG:  Solving for T, Initial residual = 0.00601807, Final residual = 7.16358e-10, No Iterations 1
max/min of T: 300/300
ExecutionTime = 142.6 s  ClockTime = 145 s

End
```

Results

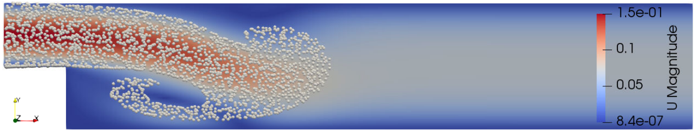
$Re = 100$



$t = 20$



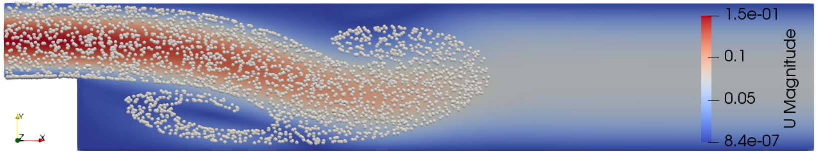
$t = 40$



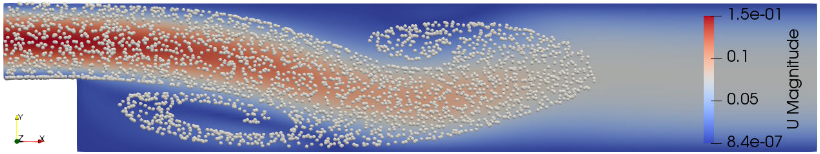
$t = 60$

Results

$Re = 100$



$t = 80$



$t = 100$

Results

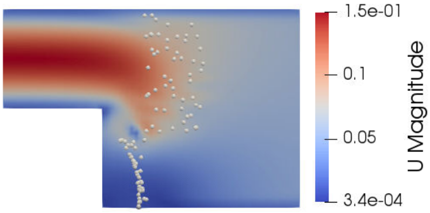
Let's do some changes

$Re = 300$

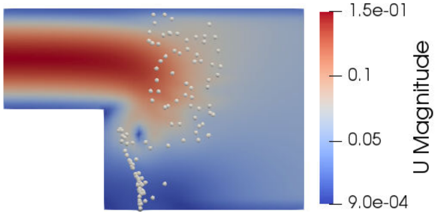
- inject particles at the 5^{th} second from patch outlet
- particles per second: 10;
- initial particle velocity $U_0 = (-700\ 0\ 0)$
- particle diameter $d = 0.01$;
- reduce the length of the channel to 2 m
- $\mu = 0.1$
- $\tau \sim \tau_0 = \dfrac{\nu}{U^2} = \dfrac{10^{-4}}{10^{-2}} = 10^{-2}$;
- $\Delta t = \frac{\tau}{2} = 0.5e - 3$;
- startTime = 0;
- endTime = 20;
- writeInterval = 2;

Results

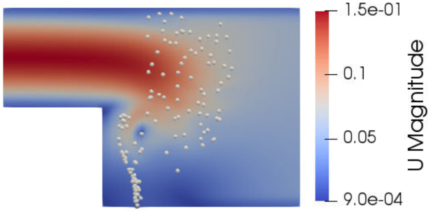
$Re = 1000$



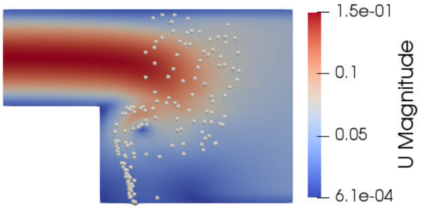
$t = 14$



$t = 16$



$t = 18$



$t = 20$

Results

Summary

- We look how particlesQHDFoam for OpenFOAM v1912 works;
- We learned how to set up basicThermoCloud for particlesQHDFoam;
- We studied how to set up and solve case step-by-step on the basic example.

Let's talk about training track.
Some questions?

Contacts

Telegram: https://t.me/qgd_qhd

GitHub: https://github.com/unicfdlab/libAcoustics/issues