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

Instructors: Tatiana V. Stenina, Aleksadr V. Ivanov
Training level: Intermediate
Session type: Lecture with examples
Software stack: OpenFOAM v1912

`https://github.com/unicfdlab`

Plan of training course

1. Training course materials
2. Introduction to regularized/QHD equations
   - Key points of training course
3. Theoretical part
   - Governing equations
   - Boundary conditions
   - Parameter τ
4. Practical part
   - How to install QHD solver
   - Stages of solution
   - Basic case
   - How to set up the particle cloud
   - Results
5. Summary

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`

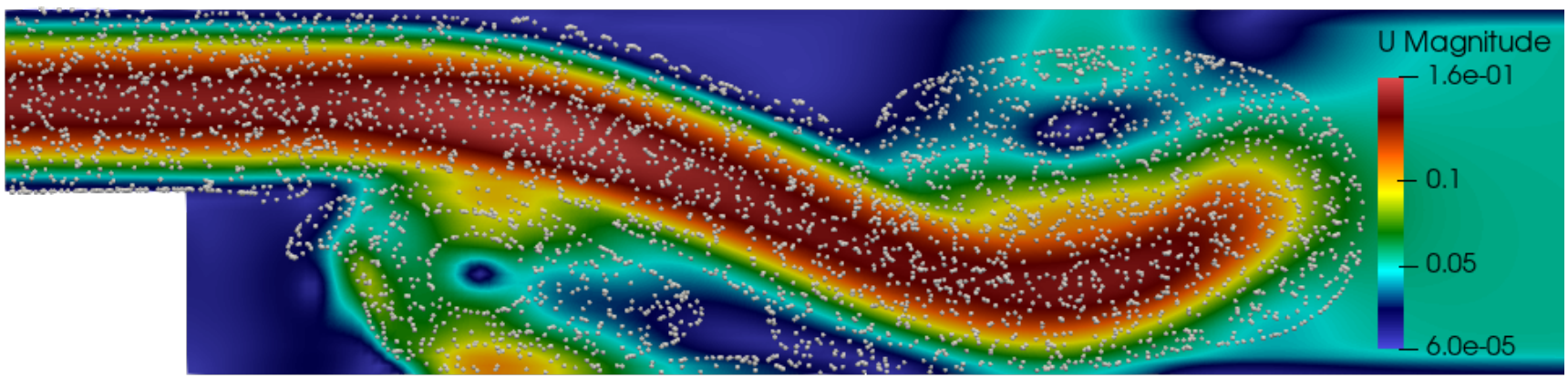
Part I
Introduction

What is regularized hydrodynamics equations

Key points of training course

The following issues 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 numerical calculations on test cases in OpenFOAM v1912.



Part I
Theoretical part

---

particlesQHD: how it works

Governing equations of Eulerian phase

- Continuity equation:

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

- Momentum equation:

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

- Scalar (temperature) transport equation:

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

- 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:

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

Heat balance equation:

$$m_p \frac{dh_p}{dt} = q_T$$

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

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

particlesQHDFoam = QHDFoam + particles

Purpose of the solver

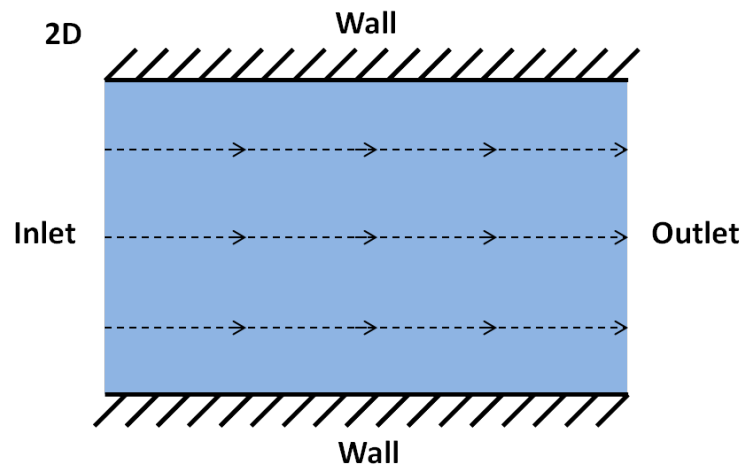
To solve equation of incompressible fluid with equations of particles clouds motion and heat transfer

Changes to the foundation solvers

The source code to solve the equation is:

```
#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 combinations in OpenFOAM.
For example, we want to set a specific pressure gradient for the wall:

```
wall
{
    type    qhdFlux;
}
```

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: $\vec{n}\vec{U} = 0$, $\frac{\partial U}{\partial n}\tau = 0$
noSlip	This boundary condition fixes the velocity to zero at walls, similar to fixedValue = 0
fixedValue	This boundary condition supplies a fixed value constraint, and is the base class for a number of other boundary conditions
zeroGradient	This boundary condition applies a zero-gradient condition from the patch internal field onto the patch faces
qhdFlux	This boundary condition is a specific condition for ∇p: $\vec{n} \cdot \vec{U} - \tau(\vec{U}\nabla\vec{U} - \frac{1}{\rho}\nabla p) \cdot \vec{n} = 0$
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 that do not constitue solution directions

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

Two stability criteria are used:

1. $\Delta t < C_\tau \tau$, 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 characteristic value of speed, l is a characteristic size.
The parameter τ has the order of the regularization coefficient τ_0.
The time integration step should not exceed τ, and is often chosen in the form:

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

Part III
Practical part

How to set up cases

How to install QGDSolver

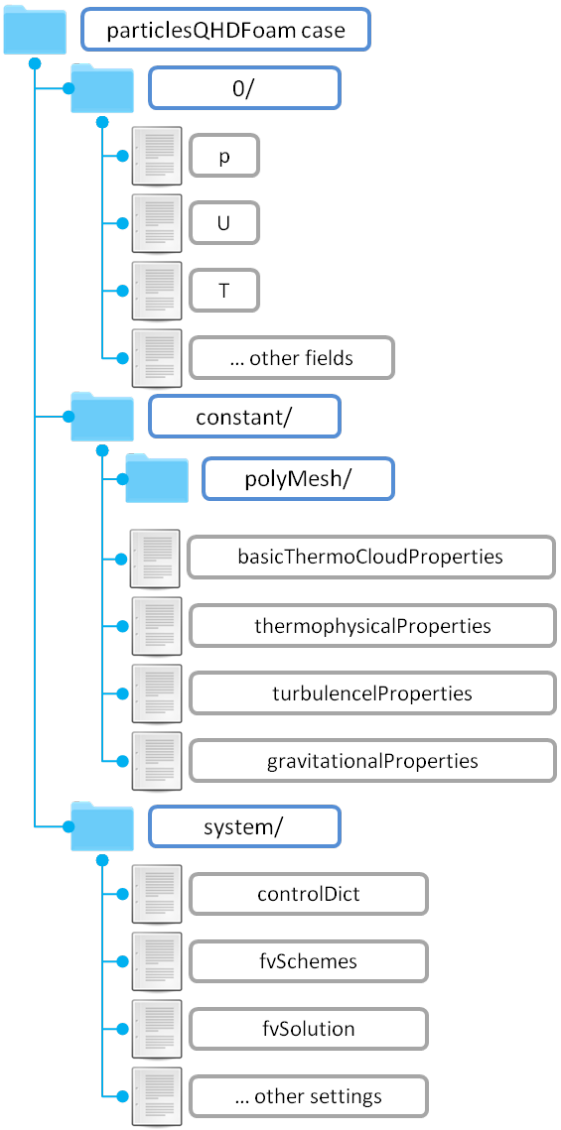
- Download QGDSolver directly from `https://github.com/unicfdlab/QGDsolver/tree/digitef-dev-1912`
  you can try short link: `https://clck.ru/QgNJy`
  or using git clone:

```
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

- prepare new case folder:

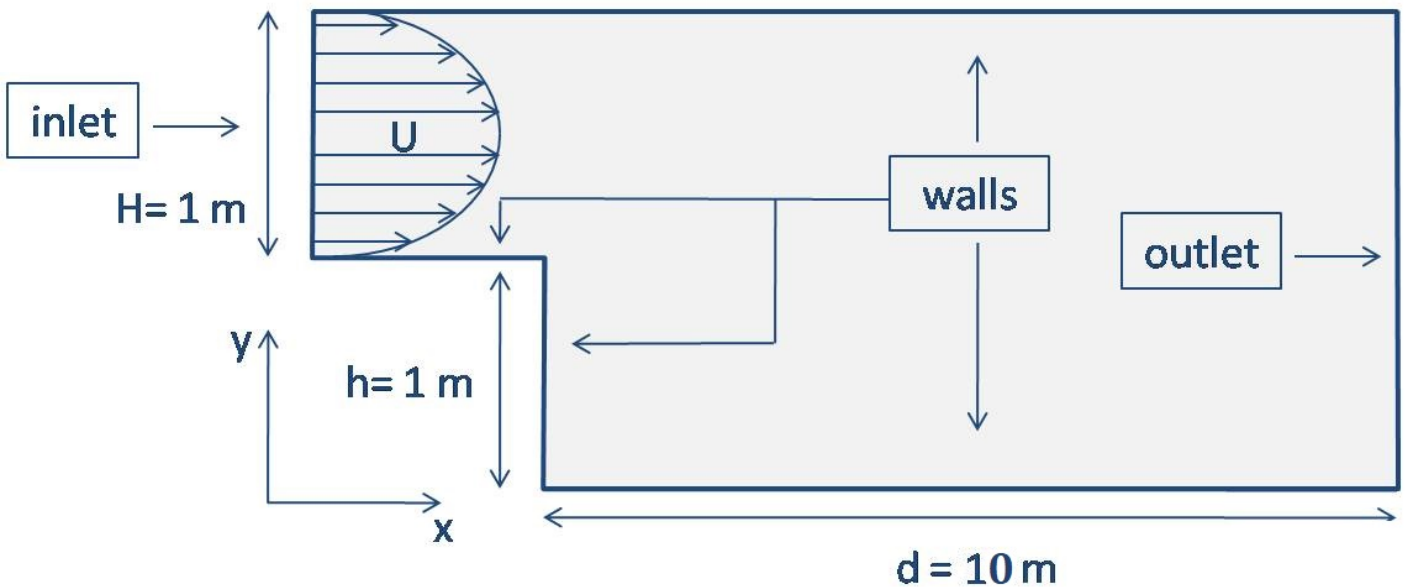
```
cp cases/backwardStep cases/backwardStep -r
```

- mesh generation
- set boundary conditions
- set physical properties
- τ selection
- time setting
- numerical schemes settings

Case set up

The case is in the folder cases/

- 2D case (backward step)
- fixed profile for velocity at the inlet
- fluid: water
- stable flow: $Re = 1 \div 1000$



Mesh generation

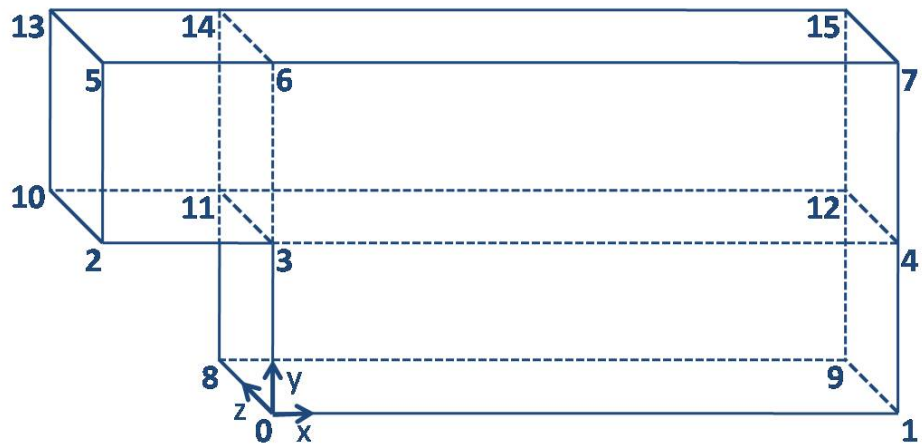
See file system/*blockMeshDict*

Set scale:

```
convertToMeters 1;
```

Set vertices:

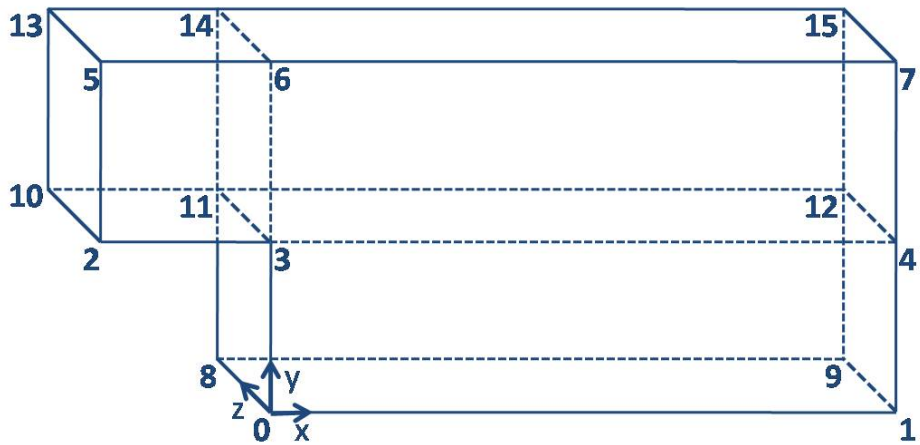
```
vertices
(
    (0 0 0)     // 0
    (25 0 0)    // 1
    (-1 1 0)    // 2
    (0 1 0)     // 3
    (25 1 0)    // 4
    (-1 2 0)    // 5
    (0 2 0)     // 6
    (25 2 0)    // 7
```



Mesh generation

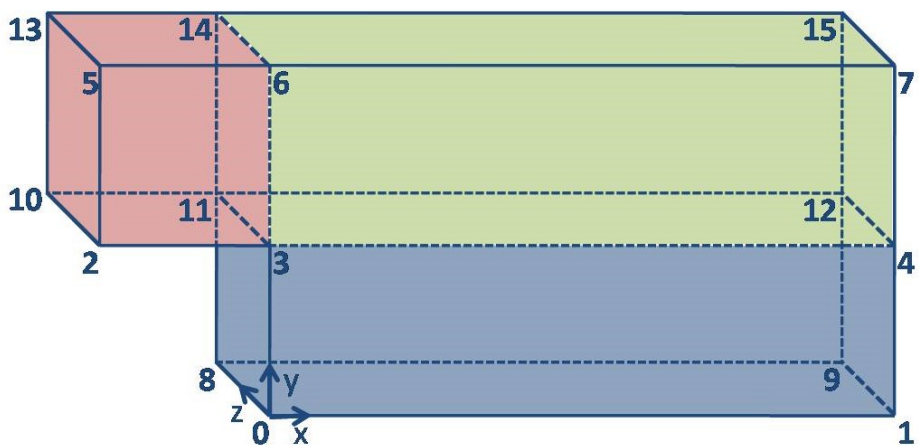
Set vertices:

```
        (0 0 0.05)      // 8
        (25 0 0.05)     // 9
        (-1 1 0.05)     // 10
        (0 1 0.05)      // 11
        (25 1 0.05)     // 12
        (-1 2 0.05)     // 13
        (0 2 0.05)      // 14
        (25 2 0.05)     // 15
);
```



Create three boxes:

```
blocks
(
    hex (0 1 4 3  8  9  12 11) (500 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) (500 20 1) simpleGrading (1 1 1)
);
```



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/

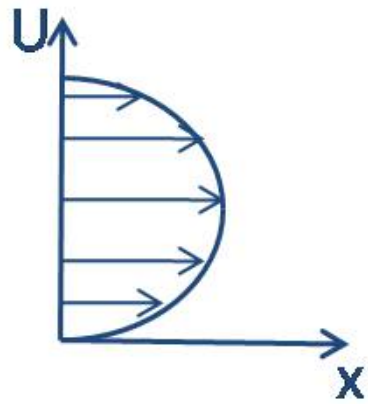
Name	U, m/s	p, Pa	T, K
inlet	fixedProfile	qhdFlux	fixedValue 300
outlet	zeroGradient	totalPressure	zeroGradient
walls	noSlip	qhdFlux	zeroGradient
symwall	empty	empty	empty

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);
origin       1;
value        uniform (0 0 0);
```

totalPressure

Set total pressure at the outlet:

```
outlet
{
  type      totalPressure;
  p0        $internalField;
  rho       rho;
  value     $internalField;
}
```

Physical properties

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

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

Set dinamic viscosity:

```
transport
{
    mu      1;
    Pr      0.73;
    beta    0.0;
}
```

τ calculation

See file *thermophysicalProperties*

```
QGD
{
    implicitDiffusion true;
    QGDCoeffs constTau;
    constTauDict
    {
        Tau 0.1; // τ ∼ τ0 = ν/U^2 = 10^-3/1^2 = 10^-3
    }
    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;
    }
    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;
    }
    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 -10 0);
```

See file *turbulenceProperties*
Set flow type:

```
simulationType    laminar;
```

How to set up the particle cloud basicThermoCloudProperties:

```
solution
{
    active                          true;
    coupled                         true;
    transient                       yes;
    cellValueSourceCorrection       off;
    maxCo                           0.3;
    interpolationSchemes
    {
        rho                         cell;
        U                           cellPoint;
        thermo:mu                   cell;
        ...
    }
    integrationSchemes
    {
        U                           Euler;
        T                           Euler;
    }
    sourceTerms
    {
        schemes
        {
            U                       explicit 1;
            h                       explicit 1;
        }
    }
}
```

- Activate/de-activate the particles
- Enable/disable phase coupling
- Transient/steady-state solution (max. Courant number)
- Enable/disable correction of momentum transferred to the Eulerian phase
- Choose interpolation/integration schemes for the LPT and treatment of source terms

How to set up the particle cloud

basicThermoCloudProperties:

```
constantProperties
{
    rho0                7874;
    T0                  300;
    Cp0                 450;
    youngsModulus       1.3e5;
    poissonsRatio       0.35;

    subModels
    {
        particleForces
        {
            sphereDrag;
            gravity;
        }
    }
}
```

Define the physical particle properties:

- Density
- Young's module (elastic modulus)
- Poisson's ratio

Define the relevant particle forces:

- Drag force
- Gravity/Buoyancy force

How to set up the particle cloud basicThermoCloudProperties:

```
injectionModels
  {
        model1
        {
          type        patchInjection;
          patch       inlet;
          duration        100;
          parcelsPerSecond        100;
          massTotal       0;
          parcelBasisType       fixed;
          flowRateProfile       constant 1;
          nParticle       1;
          SOI       0;
          U0        (9.39 0 0);
          sizeDistribution
          {
              type        fixedValue;
              fixedValueDistribution
              {
                  value       0.00007;
              }
          }
        }
  }
```

Define the particle injection:

- Injection model + injection patch name
- Total duration of particle injection
- Injected parcels/particles per second
- Number of particles per parcel
- Start-of-injection time (SOI)
- Initial parcel/particle velocity (U_0)
- Size distribution model (normal size distribution, ...)

How to set up the particle cloud

basicThermoCloudProperties:

```
dispersionModel none;
patchInteractionModel standardWallInteraction;
standardWallInteractionCoeffs
{
    type rebound;
}
heatTransferModel none;
surfaceFilmModel none;
collisionModel none;
stochasticCollisionModel none;
radiation off;
signleMixtureFractionCoeffs
{
phases
(
    gas
    {
    }
        liquid
        {
        }
    solid
    {
        C 1;
    }
);
YGasTot0 0;
YLiquidTot0 0;
YSolidTot0 1;
}
```

more options...

Time settings

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

- time interval

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

- write interval

```
writeInterval    1;
```

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

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

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. The specification of the linear equation solvers and tolerances and other algorithm controls is made in the *fvSolution* dictionary. In file *fvSolution* you can see that we use only central difference scheme.

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

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

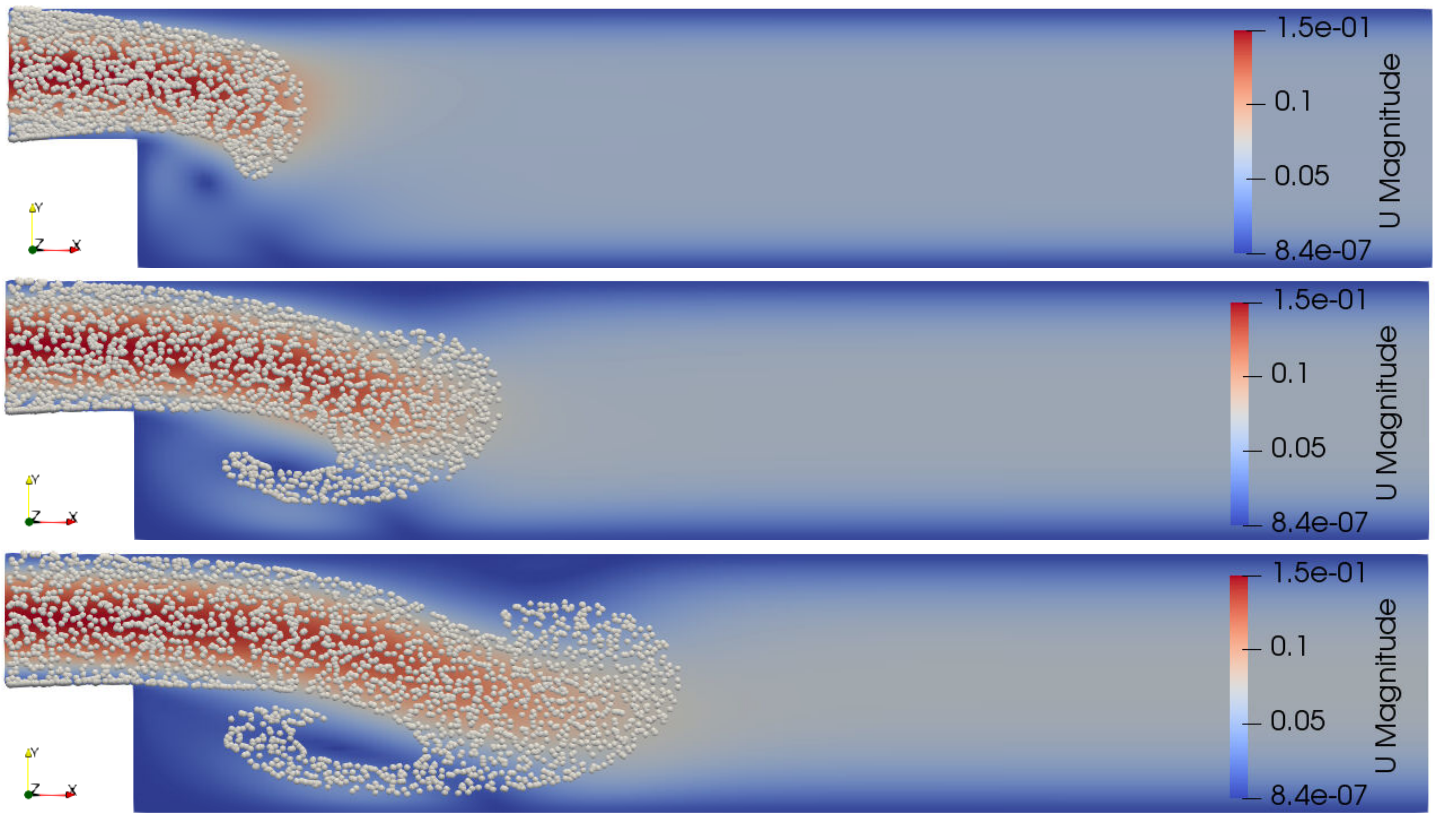
Results

After the entering of the command **particleQHDFoam**, 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
```

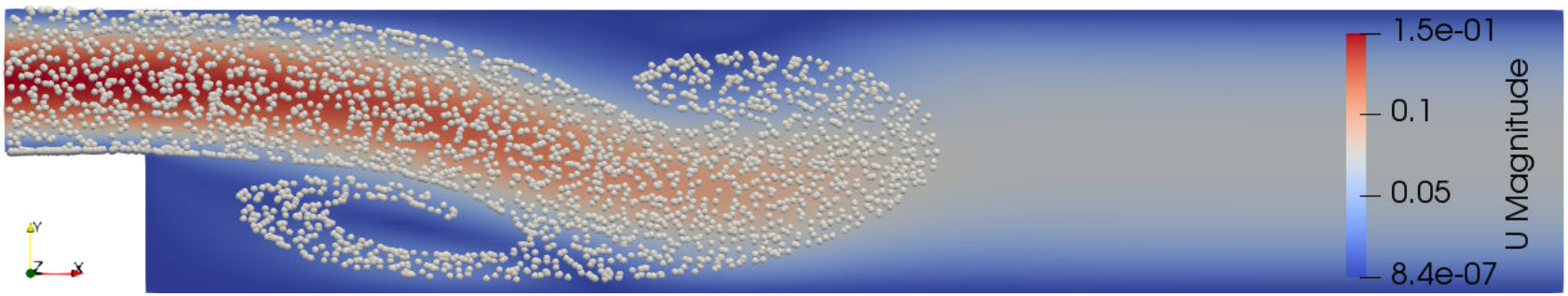
Results

$Re = 100$

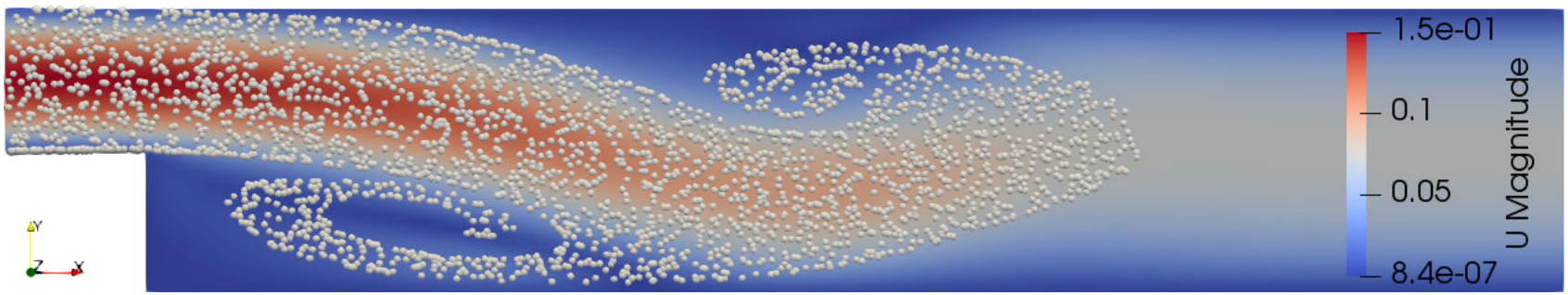


Results

$Re = 100$



$t = 80$



$t = 100$

Results

Let's do some changes

$Re = 100$

- inject particles at the 5^{th} second from patch outlet
- particles per second: 10;
- initial particle velocity $U_0 = (-700\ 0\ 0)$
- particle diameter $d = 0.01$;
- narrow down expansion channel length 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

$Re = 1000$

Figs/Re_1000/1.PNG

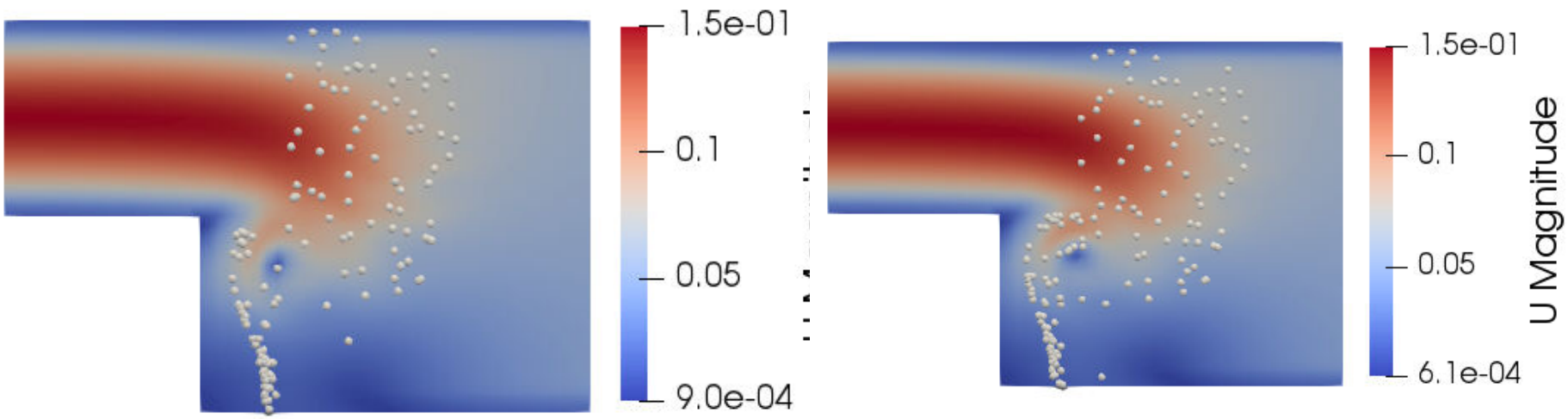
$t = 1$

Figs/Re_1000/5.PNG

$t = 5$

Figs/Re_1000/10.PNG

$t = 10$



$t = 18$

$t = 20$

Results

Summary

- We looked how particlesQHDFoam works;
- We learned some boundary conditions;
- We studied how to set up and solve cases step-by-step on the basic example in OpenFoam v1912.

Let's talk about training track.
Some questions?

particlesQHDFoam: backward step

Backward facing step considered in mulesQHDFoam case.

- particle type: copper $\rightarrow D_p = 70\mu m, \rho_p = 8800 kg/m^3$
- particles per second: 10^3, $U_0 = 9.39 m/s$
- liquid density $\rho_l = 1000$, liquid kinematic viscosity $1.5 \times 10^{-2} m^2/s$
- $\Delta t = 10^{-3} s$, $\tau = 0.1 s$
- Least Squares Method is used for τ-terms approximation