

# Incompressible flow simulation using regularized hydrodynamics equations in OpenFOAM v1912

**Instructors:** Tatiana V. Stenina, Aleksandr 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

- ① Training course materials
- ② Introduction to regularized/QHD equations
  - History
  - Application
  - Key points of training course
- ③ Theoretical part
  - Governing equations
  - Boundary conditions
  - Parameter  $\tau$
- ④ Practical part
  - How to install QHD solver
  - Stages of solution
  - Basic case
  - 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
<u><a href="#">cases</a></u>	Cases that will be used to demonstrate QHD solver's work during the track
<u><a href="#">materials</a></u>	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

## History

1982 – QGD system derived from Boltzmann equation



Prof. Boris N.  
Chetverushkin



Prof. Tatiana G.  
Elizarova

1997 – QGD system formulated as conservation laws



Prof. Yu. V.  
Sheretov

### From then to now

regularized or sometime Quasi Gas Dynamic (QGD) and Quasi Hydro Dynamic (QHD) equations are extensively used for various flows simulations – incompressible, compressible, multicomponent, magnetohydrodynamic, porous flows, two-phase flows – in Russia, Europe and in Keldysh Institute of Applied Mathematics of the RAS  
<https://keldysh.ru/>

# Pro's and Con's of QGD

## Advantages of QGD algorithms

- they can work without flux limiters
- they converge monotonically to real solution
- they do not involve Rieman-solvers
- the procedure of approximation is universal for all types of flows
- they can be integrated with other OpenFOAM models
- by contrast to PISO/SIMPLE they don't involve non-orthogonal or pressure-velocity correctors
- all abovementined features make QGD algorithms a useful tool for studying transient flows phenomena

## Drawbacks of QGD algorithms

- they are usually slower (3-4 times) than conventional PISO or Godunov-type methods
- additional conditions are imposed for stability criteria
- they require finer grids and smaller time steps in comparison with PISO algorithm for advection-dominated flows

## QGD Target audience

According to stated advantages and drawbacks of QGD algorithms, they could be useful to:

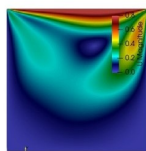
- scientists, who want to solve complex set of equations, but still haven't elaborated PISO/SIMPLE or Godunov-type procedure
- researches or engineers who want to validate other methods and programs and numerical models, but they don't have analytic solution
- engineers, who want to simulate complex transient flows which could not be reproduced by PISO/SIMPLE algorithms



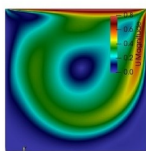
## Key points of training course

The following points will be considered:

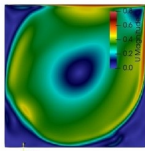
- a description of the basic principles of the solver (QHDFoam);
- setting the input parameters (initial and boundary conditions);
- running tutorials for OpenFOAM v1912.



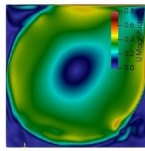
$Re = 10^2$



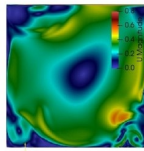
$Re = 10^3$



$Re = 10^4$



$Re = 2 \times 10^4$



$Re = 5 \times 10^4$

## Part II

### Theoretical part

---

QHDFoam: how it works

## Governing equations

- Continuity equation:

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

- Momentum equation:

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

- Scalar (temperature) transport equation:

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

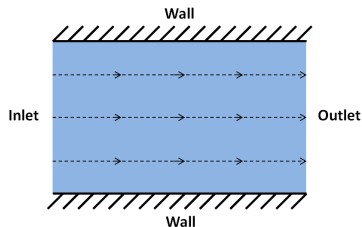
- Incompressible EoS and regularized stress tensor:

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

## Boundary conditions

### Types of boundary conditions:

- wall
- inlet
- outlet



The mathematical description of these BCs within the framework of regularized equations is set out on next slides.

## Wall

Velocity of a fluid at a wall equals  
(0, 0, 0)

Mass flux through a wall equals 0

$$\vec{U} = (0, 0, 0)$$

$$\vec{n} \cdot (\vec{U} - \vec{W}) = 0 \Rightarrow$$

$$\vec{n} \cdot \vec{U} - \tau (\vec{U} \cdot \nabla \vec{U} - \frac{1}{\rho_0} \nabla p) \cdot \vec{n} = 0 \Rightarrow$$

$$\frac{\partial p}{\partial n} = \rho_0 \vec{n} \cdot (-\vec{U} \cdot \nabla \vec{U})$$



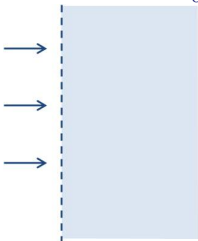
## Inlet

Velocity of a fluid at the inlet is fixed

Mass flow is constant

$$\vec{U} = \vec{U}_{in}$$

$$\begin{aligned}\vec{n} \cdot (\vec{U}_{in} - \vec{W}) &= \vec{n} \cdot \vec{U}_{in} \Rightarrow \\ \tau(\vec{U} \cdot \nabla \vec{U} - \frac{1}{\rho_0} \nabla p) \cdot \vec{n} &= 0 \Rightarrow \\ \frac{\partial p}{\partial n} &= -\rho_0 \vec{n} \cdot \vec{U} \cdot \nabla \vec{U}\end{aligned}$$



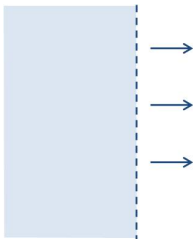
## Outlet

Velocity gradient equals 0

Set the total pressure to maintain  
the mass flow

$$\frac{\partial \vec{U}}{\partial \vec{n}} = 0$$

$$p = p_0 - \frac{\rho U^2}{2}, U^2 = \vec{U} \cdot \vec{U}$$



## Keywords in QHDFoam

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 $\nabla p$ : $\vec{n} \cdot \vec{U} - \tau(\vec{U} \cdot \nabla \vec{U} - \frac{1}{\rho_0} \nabla p) \cdot \vec{n} = 0$
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 $\tau$

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

- For simple cases,  $\tau$  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  $C_\tau$  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  $C_\tau$  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 $\tau$

Let us determine the value of the regularization coefficient  $\tau$  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  $\tau$ , and is often chosen as:

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

## 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
```

## QHDFoam case structure

It is similar to *rhoPimpleFoam* case structure

### Initial and boundary conditions

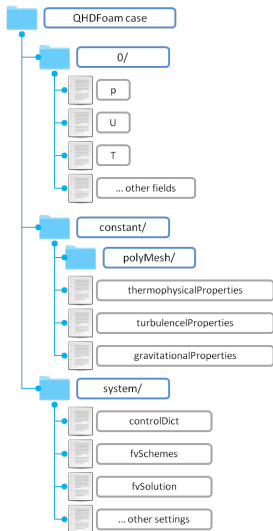
Initial conditions are set in the folder “0”. Three fields are mandatory to start a simulation: pressure “p”, velocity “U” and temperature “T”

### Fluid properties

Thermophysical fluid properties (density, heat capacity coefficients, viscosity and heat conductivity coefficients) are set in “thermophysicalProperties” dictionary. By default the turbulence modelling is turned off in the “turbulenceProperties” dictionary. Value and direction of gravity bulk field is set in “gravitationalProperties”

### Numerical schemes

Numerical schemes settings are stored in “fvSchemes” and “fvSolution”, time advancement control is in “controlDict”



## Stages of solution

See folder QHDFoam-OFv1912

- prepare new case folder:

```
cp cases/cavity cases/cavityRe1000 -r
```

- mesh generation
- boundary conditions setup
- physical properties setup
- $\tau$  selection
- advancement time settings
- numerical schemes settings

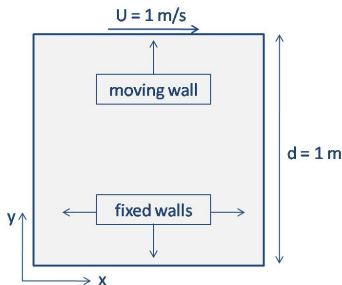


## Basic case

### Case set up

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

- 2D case (cavity): square with side  $d = 1$  m
- one moving top wall:  $\vec{U} = (1 \ 0 \ 0)$  m/s
- fluid:  $\rho = 1000$ ,  $\mu = 1$
- stable flow:  $Re = 1000$



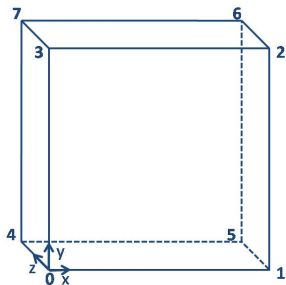
## Mesh generation

See file system/*blockMeshDict*  
Set scale:

```
convertToMeters 1;
```

Set vertices:

```
vertices
(
    (0 0 0)      // 0
    (1 0 0)      // 1
    (1 1 0)      // 2
    (0 1 0)      // 3
    (0 0 0.1)    // 4
    (1 0 0.1)    // 5
    (1 1 0.1)    // 6
    (0 1 0.1)    // 7
);
```



## Mesh generation

Create one box:

```
blocks
(
    hex (0 1 2 3 4 5 6 7) (50 50 1) simpleGrading (1 1 1)
);
```

Describe boundaries:

```
boundary
(
    movingWall
    {
        type wall;
        faces
        (
            (3 7 6 2)
        );
    }
);
```

## Mesh generation

```
fixedWalls
{
    type wall;
    faces ( (0 4 7 3) (2 6 5 1) (1 5 4 0) );
}
frontAndBack
{
    type empty;
    faces ( (0 3 2 1) (4 5 6 7) );
}
);
```

**Command:** blockMesh

## Boundary conditions

See folder 0/

Name	U, m/s	p, Pa	T, K
<b>movingWall</b>	fixedValue (1 0 0)	qhdFlux	zeroGradient
<b>fixedWalls</b>	noSlip	qhdFlux	zeroGradient
<b>frontAndBack</b>	empty	empty	empty

## Physical properties

See folder constant/

See file *thermophysicalProperties*

Set density:

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

Set dynamic viscosity:

```
transport
{
    mu      1;
    Pr      0.73;
    beta    0.0; // buoyancy factor
}
```

## $\tau$ calculation

See file *thermophysicalProperties*

```
QGD
{
    implicitDiffusion true; // approximation of viscous terms
    QGDCoeffs constTau; // choice of calculation option for  $\tau$ 
    constTauDict
    {
        Tau 1e-3; //  $\tau \sim \tau_0 = \frac{\nu}{U^2} = \frac{10^{-3}}{1^2} = 10^{-3}$ 
    }
    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.

## implicitDiffusion

**true**

Implicit approximation of viscous terms:  $\nabla \cdot \left( \frac{1}{\rho_0} \hat{\Pi} \right)$  and  $\nabla \cdot \left( \frac{\mu}{\rho_0 Pr} \nabla T \right)$

**false**

Explicit approximation of viscous terms:  $\nabla \cdot \left( \frac{1}{\rho_0} \hat{\Pi} \right)$  and  $\nabla \cdot \left( \frac{\mu}{\rho_0 Pr} \nabla T \right)$



## QGDCoeffs

We have different methods for  $\tau$  calculation:

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

## H2bynuQHD

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

where  $h$  is a grid step,  $\nu$  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

$$\tau = \frac{\tau_0}{Gr},$$

where  $\tau_0$  is a characteristic time,  $Gr$  is a Grashof number.

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

where  $g$  is acceleration of gravity,  $\beta$  is the coefficient of thermal expansion,  $T_s$  is the surface temperature,  $T_0$  is a bulk temperature,  $L$  is the characteristic length,  $\nu$  is a kinematic viscosity.

```
QGD
{
  implicitDiffusion true;
  QGDCoeffs T0byGr;
  T0byGrDict
  {
    Gr 100; T0 1e-2; //T0 =  $\tau_0$ 
  }
  pRefCell 0;
  pRefValue 0;
}
```

constTau

$\tau = \text{constant}$

```
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); // g=gy = -10
```

**See** file *turbulenceProperties*

Set flow type:

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

## 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  $C_\tau^{-1}$  (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 20;
```



## 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 ***QHDFoam*** command.

Sequence of all commands is placed in the script file: `./Allrun`.

Clean results: `./Allclean`.

# Results

$Re = 1000$

After the entering the QHDFoam command, you will see on the screen:

```

HPC CLUSTER h220 (hs 872d80377e3b)
File Edit View Terminal Tabs Help

Time = 1.6205
GAMGPCG: Solving for p, Initial residual = 0.000160216, Final residual = 8.36112e-11, No Iterations 9
GAMGPCG: Solving for Ux, Initial residual = 9.83893e-05, Final residual = 2.03472e-19, No Iterations 1
GAMGPCG: Solving for Uy, Initial residual = 0.000138275, Final residual = 2.11342e-19, No Iterations 1
GAMGPCG: Solving for T, Initial residual = 0.00102896, Final residual = 1.20196e-15, No Iterations 1
max/min of T: 309/300
ExecutionTime = 212.64 s ClockTime = 214 s

Time = 1.621
GAMGPCG: Solving for p, Initial residual = 0.000160193, Final residual = 8.33544e-11, No Iterations 9
GAMGPCG: Solving for Ux, Initial residual = 9.83804e-05, Final residual = 1.95165e-19, No Iterations 1
GAMGPCG: Solving for Uy, Initial residual = 0.000138239, Final residual = 1.93586e-19, No Iterations 1
GAMGPCG: Solving for T, Initial residual = 0.0010174, Final residual = 1.18932e-15, No Iterations 1
max/min of T: 309/300
ExecutionTime = 212.7 s ClockTime = 214 s

Time = 1.6215
GAMGPCG: Solving for p, Initial residual = 0.000160169, Final residual = 8.30978e-11, No Iterations 9
GAMGPCG: Solving for Ux, Initial residual = 9.83714e-05, Final residual = 2.03577e-19, No Iterations 1
GAMGPCG: Solving for Uy, Initial residual = 0.000138264, Final residual = 2.10733e-19, No Iterations 1
GAMGPCG: Solving for T, Initial residual = 0.00101204, Final residual = 1.16042e-15, No Iterations 1
max/min of T: 308/300
ExecutionTime = 212.75 s ClockTime = 214 s

Time = 1.622
GAMGPCG: Solving for p, Initial residual = 0.000160144, Final residual = 8.28416e-11, No Iterations 9
GAMGPCG: Solving for Ux, Initial residual = 9.83624e-05, Final residual = 1.9487e-19, No Iterations 1
GAMGPCG: Solving for Uy, Initial residual = 0.00013817, Final residual = 1.79713e-19, No Iterations 1
GAMGPCG: Solving for T, Initial residual = 0.00100914, Final residual = 1.17235e-15, No Iterations 1
max/min of T: 308/300
ExecutionTime = 212.81 s ClockTime = 214 s

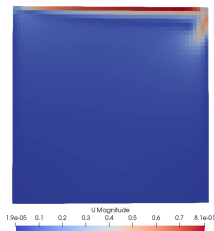
Time = 1.6225
GAMGPCG: Solving for p, Initial residual = 0.000160119, Final residual = 8.25858e-11, No Iterations 9
GAMGPCG: Solving for Ux, Initial residual = 9.83532e-05, Final residual = 2.03838e-19, No Iterations 1
GAMGPCG: Solving for Uy, Initial residual = 0.000138135, Final residual = 2.36318e-19, No Iterations 1
GAMGPCG: Solving for T, Initial residual = 0.00100385, Final residual = 1.16152e-15, No Iterations 1
max/min of T: 308/300
ExecutionTime = 212.87 s ClockTime = 214 s

Time = 1.623
GAMGPCG: Solving for p, Initial residual = 0.000160094, Final residual = 8.23308e-11, No Iterations 9

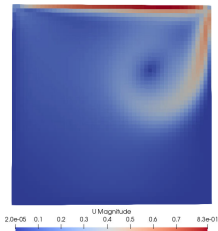
```

# Results

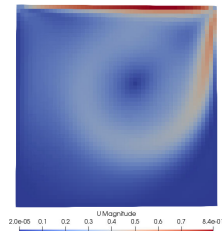
$Re = 1000$



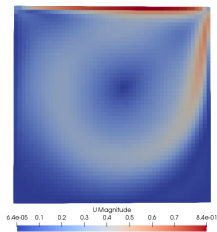
$t = 1$



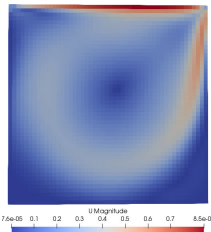
$t = 5$



$t = 10$



$t = 15$



$t = 20$

# Results

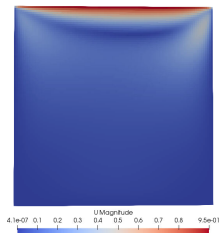
## Results

$$Re = 100$$

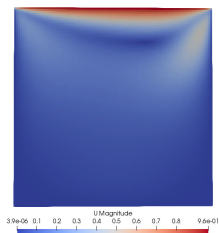
- mesh:  $100 \times 100$ ;
- $\mu = 10$ ;
- $\tau \sim \tau_0 = \frac{\nu}{U^2} = \frac{10^{-2}}{1^2} = 10^{-2}$ ;
- $\Delta t = \frac{\tau}{2} = 0.5 \cdot 10^{-2}$ ;
- startTime = 0;
- endTime = 5;
- writeInterval = 0.5;

# Results

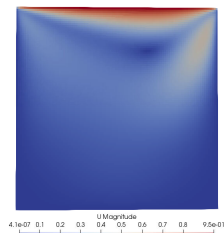
$Re = 100$



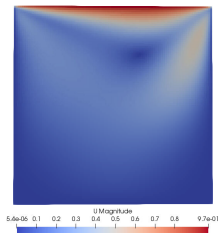
$t = 0.5$



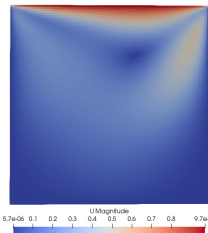
$t = 1$



$t = 2$



$t = 4$



$t = 5$

## Results

## Summary

- We look how QHDFoam for OpenFOAM v1912 works;
- We learned how to set boundary conditions for QHDFoam;
- We studied how to solve cases step-by-step on the basic example.

Let's talk about training track.  
Some questions?



## Contacts

**Telegram:** [https://t.me/qgd\\_qhd](https://t.me/qgd_qhd)

**GitHub:** <https://github.com/unicfdlab/libAcoustics/issues>