Incompressible flow simulation using regularized hydrodynamic equations in OpenFOAM v2012

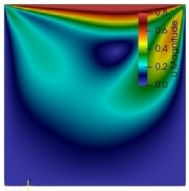
Instructors: Aleksadr V. Ivanov, Daniil A. Ryazanov
Authors: T.V. Stenina, A.V. Ivanov, M.V. Kraposhin, I.N. Sibgatullin, D.A. Ryazanov
Training level: Intermediate
Session type: Lecture with examples
Software stack: OpenFOAM v2012

`https://github.com/unicfdlab`

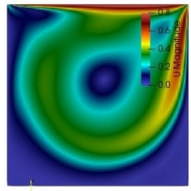
Plan of training course

About this training

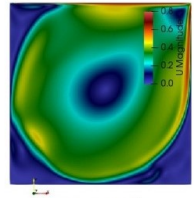
- The main purpose of this training is to introduce you to **QHDFoam** solver
- **QHDFoam** is part of QGDSolver framework
- Alternative way to model incompressible flows in OpenFOAM
- Application area – incompressible viscous fluid flows modeling with buoyancy force



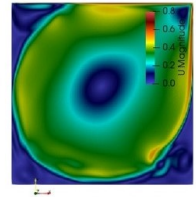
$Re = 10^2$



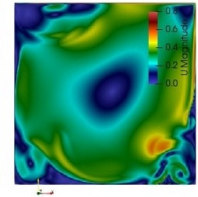
$Re = 10^3$



$Re = 10^4$



$Re = 2 \times 10^4$



$Re = 5 \times 10^4$

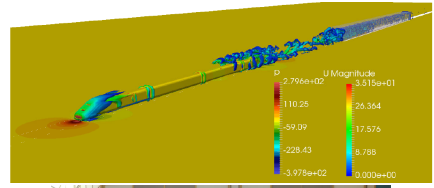
Key points of training course

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 v2012.

UniCFD Laboratory

- Open-source Software Laboratory for Digital Modeling of Technical Systems
- Founded: 15 March, 2011
- Purpose: Development of Russian tools solving continuum mechanics problems for the modeling technical systems, processes and industrial problems using open-source software (OSS)
- Technology stack: SALOME (preprocessor), OpenFOAM (solver), ParaView (postprocessor)
- Other packages: Dakota, AMReX, Code_Aster, Nek5000, Bem++, Nektar, LIGGGHTS, Peridigm
- Team: 20 specialists (1 Dr.Sci., 7 Ph.D.)



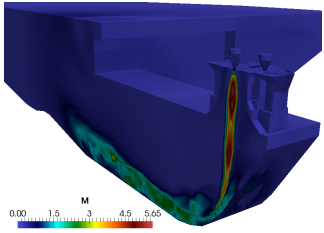
SALOME

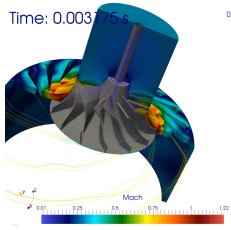
OpenFOAM

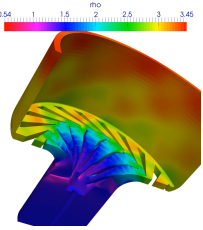
ParaView

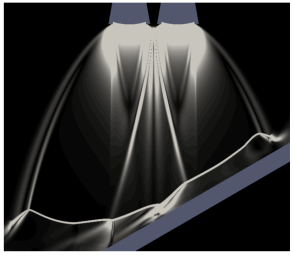
Our software

- **libAcoustics** – the toolbox for numerical analysis of acoustic noise `https://github.com/unicfdlab/libAcoustics`
- **hybridCentralSolvers** – a set of programs for the simulation of compressible flows in wide Mach number range. `https://github.com/unicfdlab/hybridCentralSolvers`
- **QGDsolver framework** `https://github.com/unicfdlab/QGDsolver`









QGDsolver framework

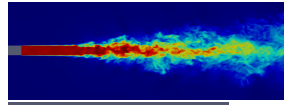
It contains:

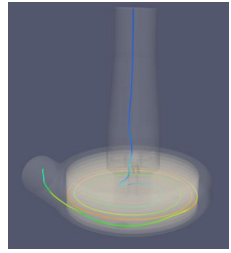
- solver for incompressible viscous fluid flows
  - in rotating frame of reference
  - in domains with deforming boundary
  - with 2-phase flows
  - with scalar transport modeling
  - with particles modeling
- solver for compressible viscous perfect gas flows in a wide Mach number range – from 0 to infinity
  - with particles modeling
  - reacting multicomponent compressible viscous perfect gas

inter
scalarTransport
Particles
SRF
QHD Foam
DyMFoam

particles
reactingLagrangian
QGD Foam

Quasi
Gas
Hydro
Dynamic





Other training tracks

https://unicfd.ru/en/training-tracks/
https://github.com/unicfdlab/TrainingTracks

- How to use free-surface flows in OpenFOAM v1812
- Implementation of the solver for coupled simulation for heat transfer in gas and solid
- Computational Aeroacoustics Methods with OpenFOAM v1812
- Implementation of simple FSI model with functionObject
- **Compressible flow simulation using regularized quasi-gas dynamic equations in OpenFOAM v2012 – current OpenFOAM Workshop 2021, 11^{th} June**

Before we start

If you are listener of this course, you should:

- have basic knowledge of OpenFOAM
- know basic commands for Linux terminal
- **have preinstalled OpenFOAM v2012 on your laptop**

Training course materials

- Course location: `https://github.com/unicfdlab/TrainingTracks`
- Folder QHDFoam-OFv2012

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
Beginning

What are regularized hydrodynamics equations?

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 Riemann-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 abovementioned 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

Part II
Theoretical part

QHDFoam: how it works

Governing equations

- 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:

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

- Scalar (temperature) transport equation:

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

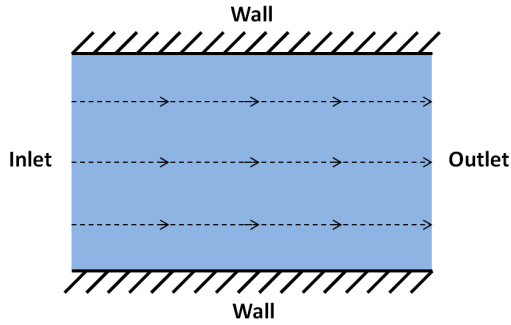
- Incompressible EoS and regularized stress tensor:

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

Boundary conditions

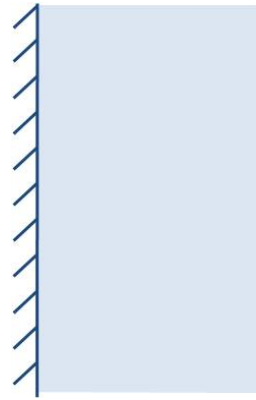
Types of boundary conditions:

- 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:

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

Mass flux through a wall:

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

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

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

Inlet

Velocity of a fluid at the inlet is fixed:

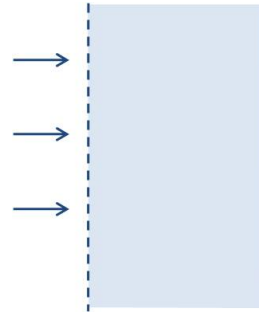
$$\vec{U} = \vec{U}_{in}$$

Mass flux through a wall:

$$\vec{n} \cdot (\vec{U}_{in} - \vec{W}) = \vec{n} \cdot \vec{U}_{in} \implies$$

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

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

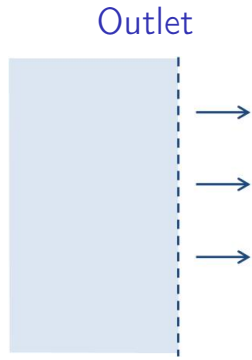


Velocity gradient:

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

Set the total pressure to maintain the mass flow

$$p = p_0 - \frac{\rho |\vec{U}|^2}{2}, \quad |\vec{U}|^2 = \left(\vec{U} \cdot \vec{U}\right).$$



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: $\vec{U}_n = 0, \frac{\partial \vec{U}_\tau}{\partial \vec{n}} = 0$
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 to balance τ-terms of mass flux: $\vec{n} \cdot \vec{U} - \tau(\vec{U} \cdot \nabla \vec{U} - \frac{1}{\rho_0} \nabla p) \cdot \vec{n} = 0$
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: $\tau \approx \dfrac{Co^{max} \Delta x}{|\vec{U}|^{max}} C_\tau$, where C_τ is a constant less than 1

Three stability criteria are used:

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

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+ v2012, for other OpenFOAM version, different branches should be used.

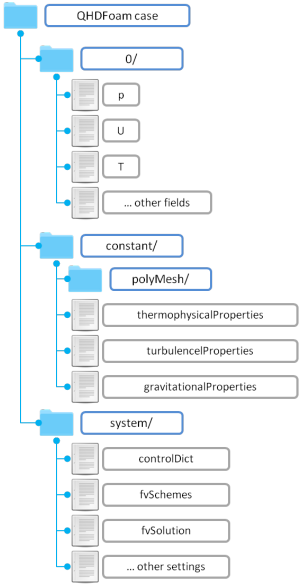
- Download QGDSolver directly from `https://github.com/unicfdlab/QGDsolver/tree/digitef-dev-2012` you can try short link: `https://clck.ru/QgNJy`
  or using git clone:

```
git clone https://github.com/unicfdlab/QGDsolver.git
git checkout digitef-dev-2012
```

- 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 "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"

Numerical schemes

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

Stages of solution

See folder QHDFoam-OFv2012

- prepare new case folder:

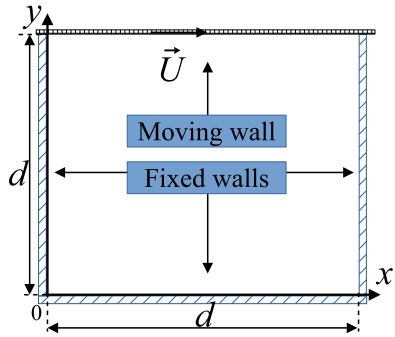
```
cp cases/cavity cases/cavityRe1000 -r
```

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

Case set up

See folder QHDFoam-OFv2012. 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: $Re = 1000$



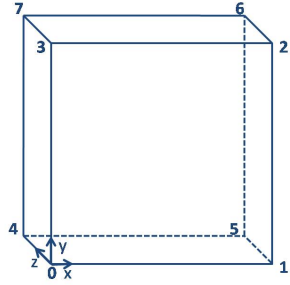
See file system/*blockMeshDict*

Set scale:

```
scale 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
);
```



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
movingWall	fixedValue (1 0 0)	qhdFlux	zeroGradient
fixedWalls	noSlip	qhdFlux	zeroGradient
frontAndBack	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
}
```

τ calculation

See file *thermophysicalProperties*

```
QGD
{
    implicitDiffusion true; // approximation of viscous terms
    QGDCoeffs constTau; // choice of calculation option for τ
    constTauDict
    {
        Tau 1e-3; // 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.

implicitDiffusion

true

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

false

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

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;
```

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

where τ_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, β is the coefficient of thermal expansion, $T_s$is the surface temperature, T_0 is a bulk temperature, L is the characteristic length, ν is a kinematic viscosity.

```
QGD
{
    implicitDiffusion true;
    QGDCoeffs T0byGr;
    T0byGrDict
    {
        Gr 100; T0 1e-2; //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 -10 0); // g = g_y = −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_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    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.

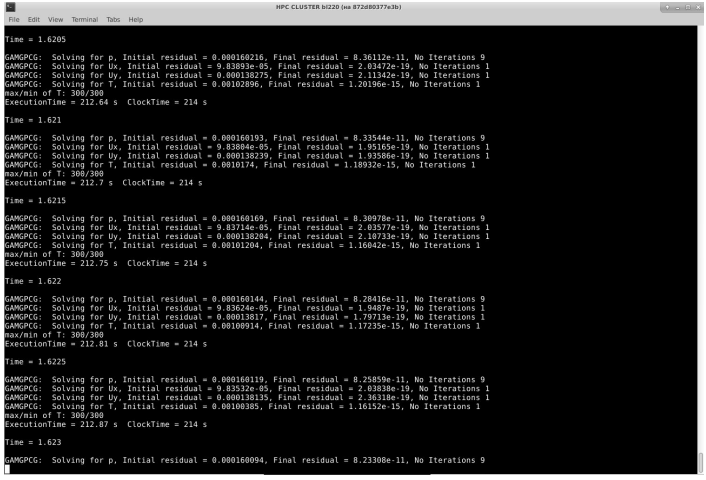
You can start application by ***QHDFoam*** command.

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

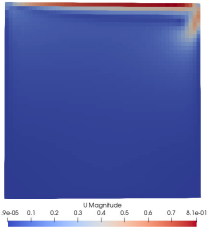
Results

$Re = 1000$

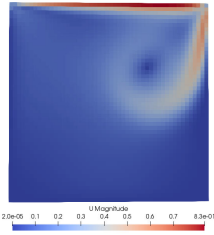
After the entering the QHDFoam command, you will see on the screen:



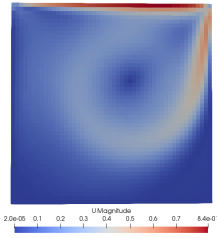
$Re = 1000$



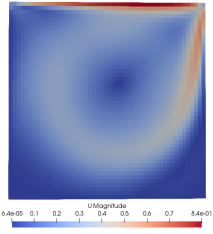
$t = 1$



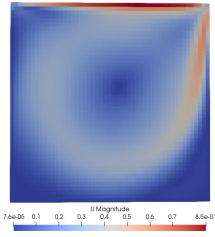
$t = 5$



$t = 10$



$t = 15$



$t = 20$

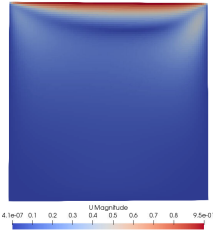
Results

Results

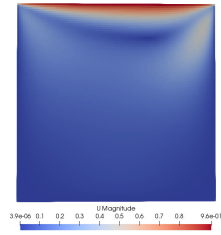
$Re = 100$

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

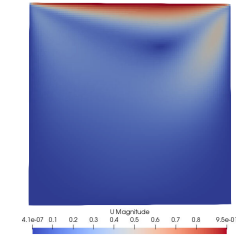
$Re = 100$



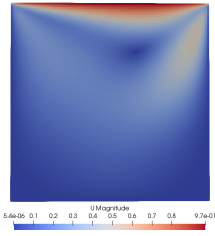
$t = 0.5$



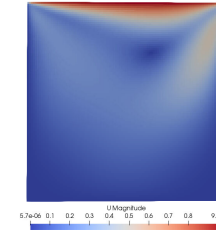
$t = 1$



$t = 2$



$t = 4$



$t = 5$

Results

Summary

- We look how QHDFoam for OpenFOAM v2012 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`

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