

Incompressible flow simulation using regularized hydrodynamic equations in OpenFOAM v2112

Instructors: Aleksandr 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 v2112

<https://github.com/unicfdlab>

Before we start

If you are listener of this course, you should:

- have basic knowledge of the OpenFOAM
- know basic commands for Linux terminal
- **have preinstalled OpenFOAM v2112 on your laptop**
- **preinstall the QGDSolver v2112 on your laptop**
 - clone git repository

```
> git clone https://github.com/unicfdlab/QGDSolver.git  
> cd QGDSolver  
> git checkout digitef-dev-2112
```

- Install QGDSolver:

```
> ./Allwmake
```

Plan of training course

① Before we start

② Introduction

About this training

Key points of training course

Our laboratory

③ Training course materials

④ What are regularized/QHD equations

History

Application

⑤ Theoretical part

Governing equations

Boundary conditions

Parameter τ

⑥ Practical part

How to install QHD solver

Stages of solution

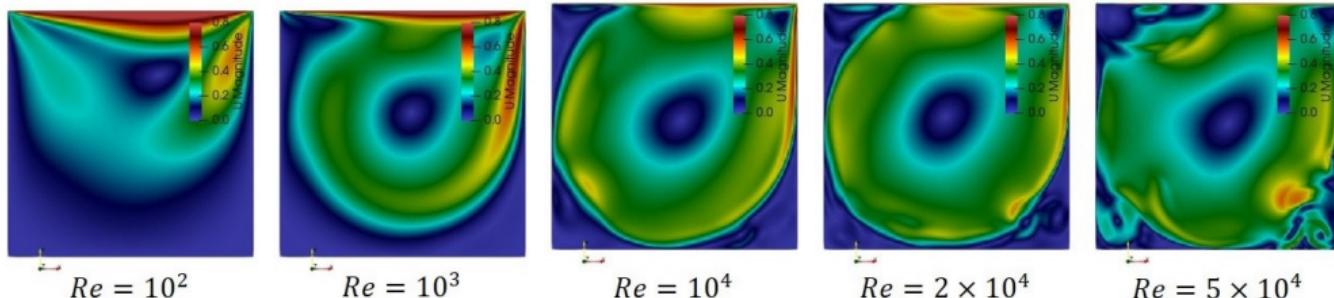
Basic case

Results

⑦ Summary

About this training

- This is a hands-on tutorial
- The main purpose of this training is to introduce you to the **QHDFoam** solver
- The **QHDFoam** is a part of the QGDSolver framework
- It is an alternative way to model incompressible flows in OpenFOAM
- Its application area – incompressible viscous fluid flows modeling with buoyancy force



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 v2112.

Designations:

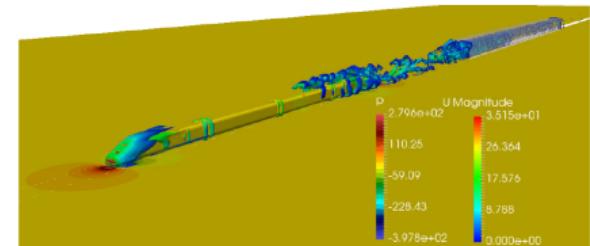
- Linux terminal command

> example command

- block of text from some file

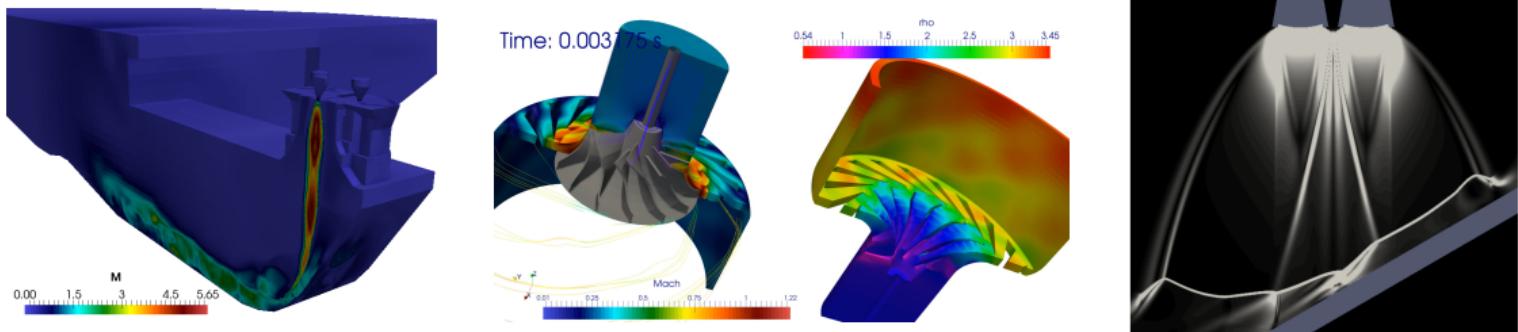
Some text.

UniCFD Laboratory

**SALOME****Open ∇ FOAM****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

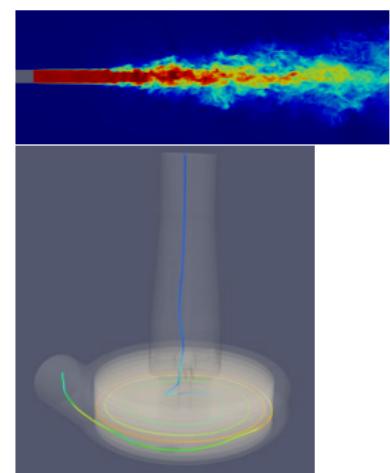
inter
scalarTransport
Particles
SRF

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 v2112 – current OpenFOAM Workshop 2022, 11th June**

Training course materials

- Course location: <https://github.com/unicfdlab/TrainingTracks>
- Folder QHDFoam-OFv2112

| Folder | Description |
|------------------|---|
| <u>cases</u> | Cases that will be used to demonstrate QHD solver's work during the track |
| <u>materials</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 Beginning

What are 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 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
- researchers 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:

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

- Momentum equation:

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

- Scalar (temperature) transport equation:

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

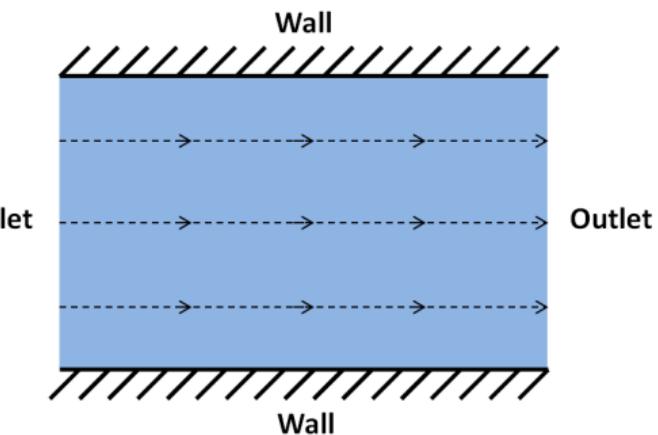
- Incompressible EoS and regularized stress tensor:

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

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:

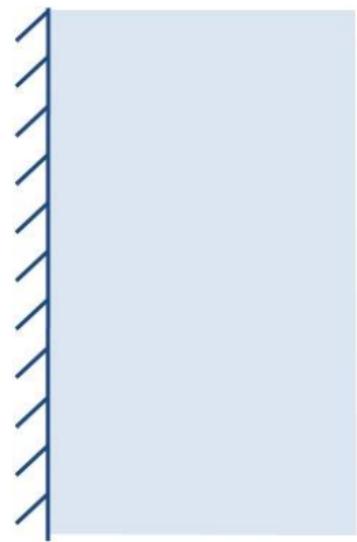
$$\vec{U} = (0, 0, 0)$$

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:

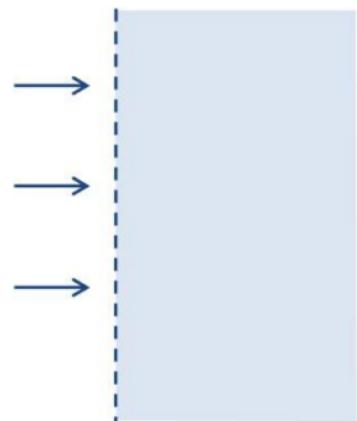
$$\vec{U} = \vec{U}_{in}$$

Mass flux through a wall:

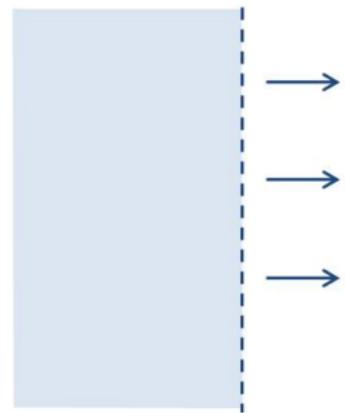
$$\vec{n} \cdot (\vec{U}_{in} - \vec{W}) = \vec{n} \cdot \vec{U}_{in} \implies$$

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

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



Outlet



Velocity gradient:

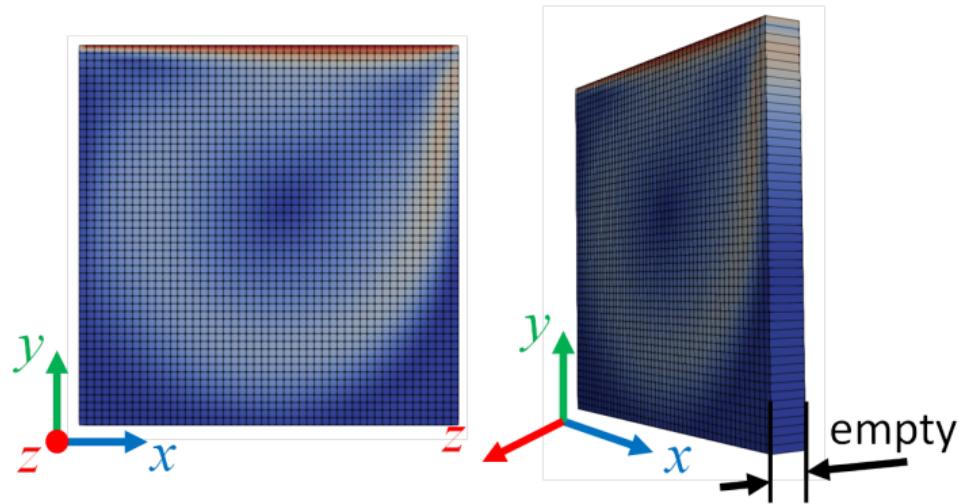
$$\frac{\partial \vec{U}}{\partial \vec{n}} = 0$$

Set the total pressure to maintain the mass flow

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

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.



Keywords in OpenFoam

There are prepared keywords for these boundary conditions dictionary 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);
}
```

The most common keywords

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

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:

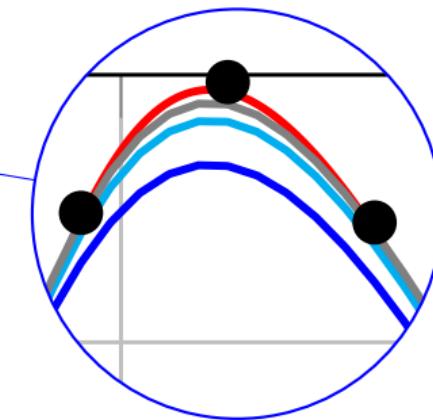
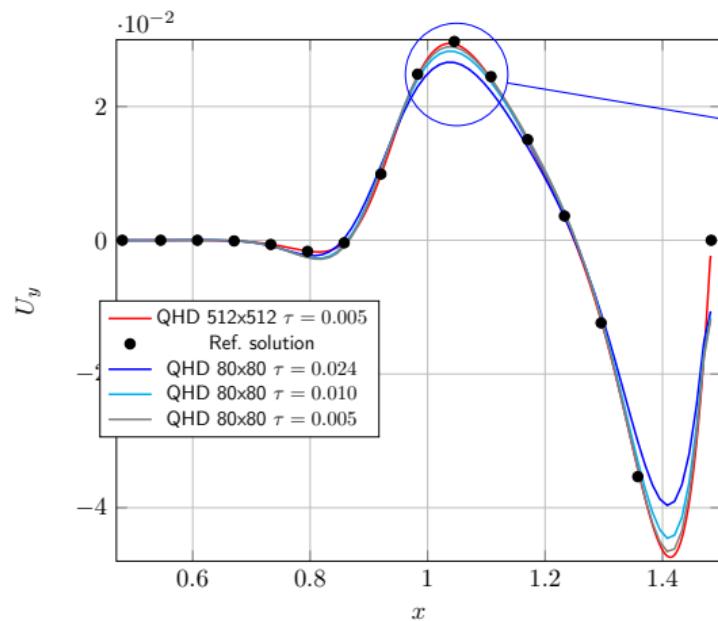
- For simple cases, τ could be estimated from dimensionless numbers (like Re, Gr or others):
 $\tau \approx \tau_0 Re^{-1}$, $\tau \approx \tau_0 Gr^{-1}$;
- Through the max CFL $Co^{max} = |\vec{U}|^{max} \Delta t / \Delta x$ number: $\tau \approx \frac{Co^{max} \Delta x}{|\vec{U}|^{max}} C_\tau$, where 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 $\tau |\vec{U}| \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}$

How τ works

- $\tau \rightarrow 0 \implies \text{solution} \rightarrow \text{exact solution}$
- limitation: $\Delta t \leq \tau$



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:

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

Part III Practical part

How to set up cases

How to install QGDSolver

This is for OpenFOAM+ v2112, for other OpenFOAM version, different branches should be used.

- Download QGDSolver

- directly from

<https://github.com/unicfdlab/QGDSolver/tree/digitef-dev-2112>

you can try short link:

<https://shorturl.at/dgC04>

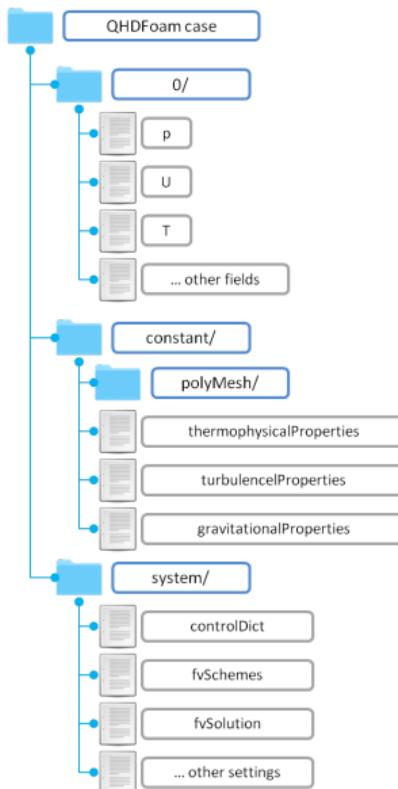
- or using git clone:

```
> git clone https://github.com/unicfdlab/QGDSolver.git  
> cd QGDSolver  
> git checkout digitef-dev-2112
```

- 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-OFv2112

- prepare new case folder:

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

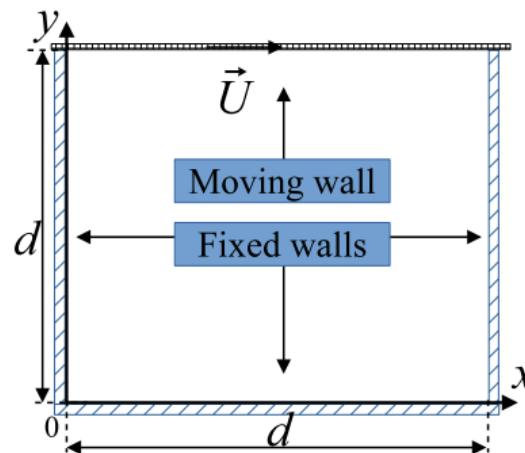
- mesh generation
- boundary conditions setup
- physical properties setup
- τ selection
- advancement time settings
- numerical schemes settings

Basic case

Case set up

See folder QHDFoam-OFv2112. The case is in the folder cases/

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



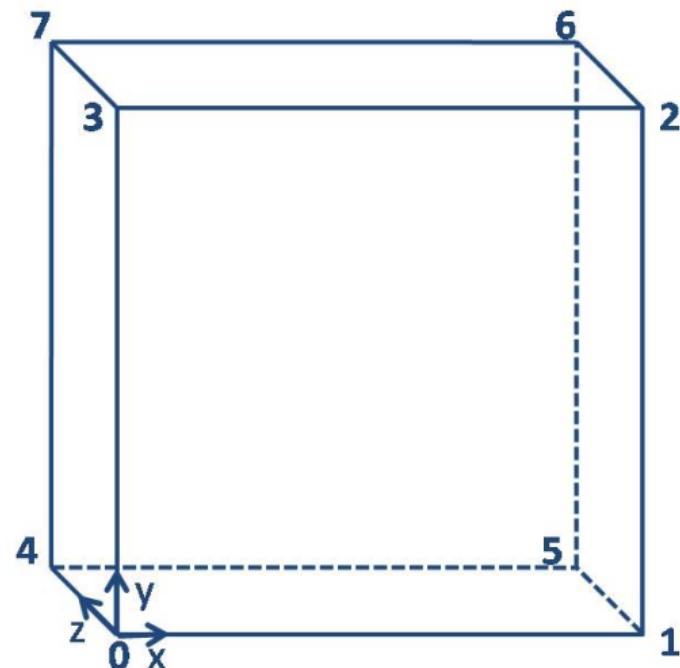
Mesh generation

See file system / *blockMeshDict*

```
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
);
```



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 |
|---------------------|-----------------------|---------|--------------|
| 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  $\tau$ 
    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: $\nabla \cdot (\frac{1}{\rho_0} \hat{\Pi})$ and $\nabla \cdot \left(\frac{\mu}{\rho_0 Pr} \nabla T \right)$

false

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

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;
}
```

$$\tau = \frac{\tau_0}{Gr}, \quad Gr = \frac{g\beta(T_s - T_0)L^3}{\nu^2}$$

T0byGr

where τ_0 is a characteristic time, Gr is a Grashof number; 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

$$\tau = \text{constant}$$

```
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 -9.81 0); //  $g = g_y = -9.81 \text{ m/s}^2$ 
```

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

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.

Run solver

You can start application by command:

```
> QHDFoam
```

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:

```
Time = 1.6205
GAMGCG: Solving for p. Initial residual = 0.000160216, Final residual = 8.26112e-11, No Iterations 9
GAMGCG: Solving for Ux, Initial residual = 9.83069e-05, Final residual = 2.03472e-19, No Iterations 1
GAMGCG: Solving for Uy, Initial residual = 0.000136275, Final residual = 2.11342e-19, No Iterations 1
GAMGCG: Solving for T, Initial residual = 0.00102896, Final residual = 1.20196e-15, No Iterations 1
max/min of T: 398/380
ExecutionTime = 212.64 s ClockTime = 214 s

Time = 1.621
GAMGCG: Solving for p. Initial residual = 0.000160193, Final residual = 8.33544e-11, No Iterations 9
GAMGCG: Solving for Ux, Initial residual = 9.83069e-05, Final residual = 1.95165e-19, No Iterations 1
GAMGCG: Solving for Uy, Initial residual = 0.000136239, Final residual = 1.93586e-19, No Iterations 1
GAMGCG: Solving for T, Initial residual = 0.0010174, Final residual = 1.18932e-15, No Iterations 1
max/min of T: 398/380
ExecutionTime = 212.7 s ClockTime = 214 s

Time = 1.6215
GAMGCG: Solving for p. Initial residual = 0.000160169, Final residual = 8.30978e-11, No Iterations 9
GAMGCG: Solving for Ux, Initial residual = 9.83714e-05, Final residual = 2.03577e-19, No Iterations 1
GAMGCG: Solving for Uy, Initial residual = 0.000136284, Final residual = 2.10733e-19, No Iterations 1
GAMGCG: Solving for T, Initial residual = 0.00101204, Final residual = 1.16642e-15, No Iterations 1
max/min of T: 398/380
ExecutionTime = 212.75 s ClockTime = 214 s

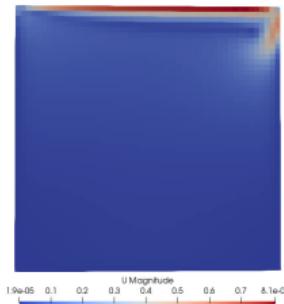
Time = 1.622
GAMGCG: Solving for p. Initial residual = 0.000160144, Final residual = 8.28416e-11, No Iterations 9
GAMGCG: Solving for Ux, Initial residual = 9.83624e-05, Final residual = 1.9487e-19, No Iterations 1
GAMGCG: Solving for Uy, Initial residual = 0.00013617, Final residual = 1.79713e-19, No Iterations 1
GAMGCG: Solving for T, Initial residual = 0.00100914, Final residual = 1.17235e-15, No Iterations 1
max/min of T: 398/380
ExecutionTime = 212.81 s ClockTime = 214 s

Time = 1.6225
GAMGCG: Solving for p. Initial residual = 0.000160119, Final residual = 8.25859e-11, No Iterations 9
GAMGCG: Solving for Ux, Initial residual = 9.83532e-05, Final residual = 2.03838e-19, No Iterations 1
GAMGCG: Solving for Uy, Initial residual = 0.000136135, Final residual = 2.36318e-19, No Iterations 1
GAMGCG: Solving for T, Initial residual = 0.00100385, Final residual = 1.16152e-15, No Iterations 1
max/min of T: 398/380
ExecutionTime = 212.87 s ClockTime = 214 s

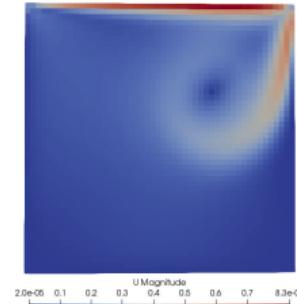
Time = 1.623
GAMGCG: Solving for p. Initial residual = 0.000160094, Final residual = 8.23308e-11, No Iterations 9
```

$Re = 1000$

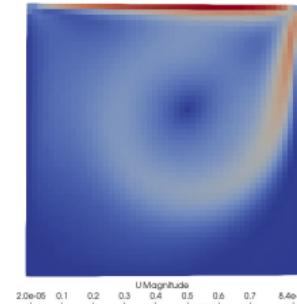
Results



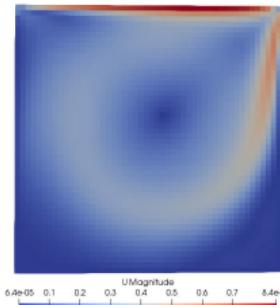
$t = 1$



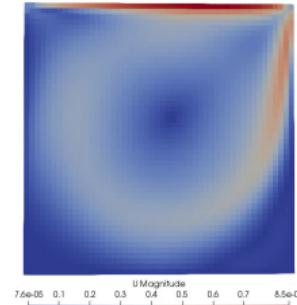
$t = 5$



$t = 10$



$t = 15$



$t = 20$

Results

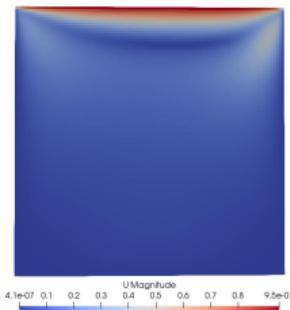
Results

$Re = 100$

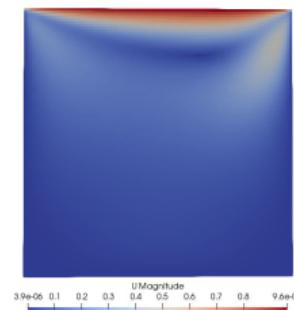
- mesh: 100×100 ;
- $\mu = 10$;
- $\tau \sim \tau_0 = \frac{\nu}{U^2} = \frac{10^{-2}}{1^2} = 10^{-2}$;
- $\Delta t = \frac{\tau}{2} = 0.5 \cdot 10^{-2}$;
- startTime = 0;
- endTime = 5;
- writeInterval = 0.5;

$Re = 100$

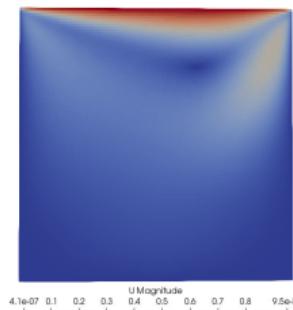
Results



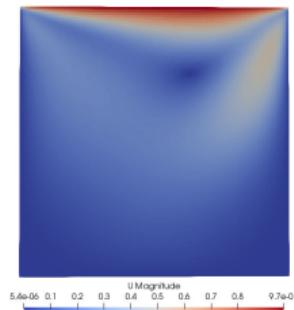
$t = 0.5$



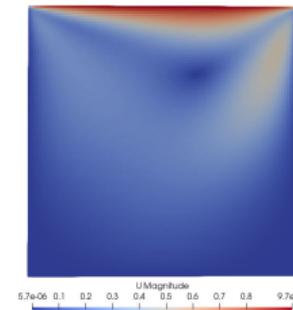
$t = 1$



$t = 2$



$t = 4$



$t = 5$

Results

Summary

- We have seen how QHDFoam for OpenFOAM v2112 works;
- We have learned how to set boundary conditions for QHDFoam;
- We have studied how to solve cases step-by-step on the basic example.

Let's talk about training track.
Some questions?

Contacts

Website: <https://unicfd.ru/en>
Telegram: https://t.me/qgd_qhd
GitHub: <https://github.com/unicfdlab/>
Instagram: @unicfd.lab
YouTube: UNICFD LAB
Twitter: @unicfd

