Compressible flow simulation using regularized quasi-gas dynamic equations in OpenFOAM v2012

Instructors: Maria A. Kiryushina, Andrey S. Epikhin
Authors: T.G. Elizarova, M.V. Kraposhin,
M.A. Kiryushina, A.S. Epikhin
Training level: Intermediate
Session type: Lecture with examples
Software stack: OpenFOAM v2012

`https://github.com/unicfdlab`

About ISP RAS

Open-source Software Laboratory for Digital Modeling of Technical Systems

Main areas:

- development of software for solving problems of mechanics of continuous media;
- numerical simulations including pre- and postprocessing of industrial problems;
- mesh generation;
- visualization;
- fundamental researches;
- education, consulting.
  `https://unicfd.ru/en/about`

Plan of training course

Before we start

For effective participation you should

- have basic knowledge of OpenFOAM
- know basic commands for Linux terminal
- **have preinstalled OpenFOAM v2012 on your laptop OR ability to boot from USB**
- have Internet connection

Training course materials

- **Course location**: `https://github.com/unicfdlab/TrainingTracks/tree/master/OpenFOAM/QGDFoam-OFv2012`
- **Full version of the solver**: `https://github.com/unicfdlab/QGDsolver`

Folder	**Description**
cases	To demonstrate QGD solver's work during the track
materials	This presentation and other materials used in the course

Key points of training course

The following points will be considered:

- a description of the basic principles of the QGDFoam solver;
- setting of the input parameters (initial and boundary conditions);
- running tutorials for OpenFOAM v2012.

Part I
Introduction

What is regularized gas dynamic equations

History

1982 – QGD system derived from Boltzmann equation

1997 – QGD system formulated as conservation laws

Prof. Boris N. Chetverushkin

Prof. Tatiana G. Elizarova

Prof. Yu. V. Sheretov

During last 20 years regularized or Quasi Gas Dynamic (QGD) equations are used for various flows simulations – compressible, multicomponent, magnetohydrodynamic, porous flows, two-phase flows and others especially in Institute of Applied Mathematics of the RAS.
`https://keldysh.ru/`

Pro's and Con's of QGD

Advantages of QGD algorithms

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

Drawbacks of QGD algorithms

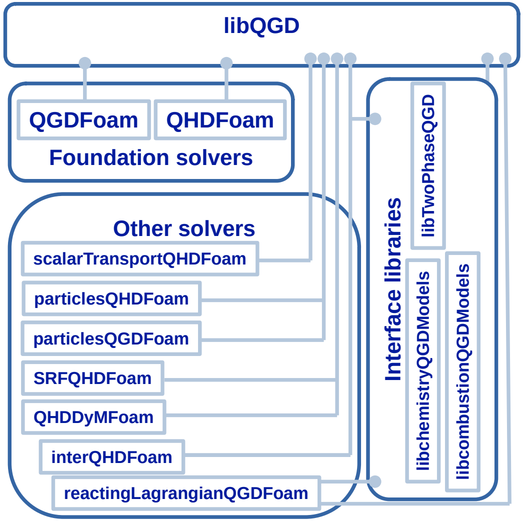
- they are usually slower (3-4 times) than conventional PISO or Godunov-type methods
- additional conditions are imposed for stability criteria
- they require finer grids and smaller time steps in comparison with PISO algorithm for advection-dominated flows

QGD target audience

QGD algorithms could be useful to:

- uniform flow simulations in the wide range of Mach numbers from subsonic to high velocity supersonic flows
- scientists to solve complex set of equations, but still haven't elaborated PISO/SIMPLE or Godunov-type procedure
- researches or engineers to validate other methods and programs and numerical models
- engineers to simulate complex transient flows which could not be reproduced by PISO/SIMPLE algorithms

QGDsolver framework structure



- Each solver implementing QGD algorithm must use libQGD library
- Two foundation solvers *QGDFoam* and *QHDFoam* show essential principles of QGD-algorithms
- Other solver could be regared as combination of foundation algorithms and OpenFOAM models
- Interface libraries are used to connect QGD solver to OpenFOAM models when interfaces have changed

Part II
Theoretical part

QGDFoam: how it works

QGDFoam governing equations

- Continuity equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \vec{j}_m = 0, \quad \vec{j}_m = \rho\left(\vec{U} - \vec{w}\right), \vec{w} = \frac{\tau}{\rho}\left(\nabla \cdot (\rho\vec{U} \otimes \vec{U}) + \nabla p\right)$$

- Momentum equation: $$\frac{\partial \rho\vec{U}}{\partial t} + \nabla \cdot \left(\vec{j}_m \otimes \vec{U}\right) + \nabla p = \nabla \cdot \hat{\Pi},$$

$$\hat{\Pi} = \hat{\Pi}_{NS} + \tau\vec{U} \otimes \left(\rho\left(\vec{U} \cdot \nabla\right)\vec{U} + \nabla p\right) + \tau\hat{I}\left(\left(\vec{U} \cdot \nabla\right)p + \gamma p \nabla\vec{U}\right)$$

$$\hat{\Pi}_{NS} = \mu\left((\nabla \otimes \vec{U}) + (\nabla \otimes \vec{U})^T - \frac{2}{3}\hat{I}\mathrm{div}\tilde{\mathrm{U}}\right)$$

- Energy equation: $$\frac{\partial \rho e}{\partial t} + \nabla \cdot \left(\vec{j}_m\left(e + \frac{p}{\rho}\right)\right) = \nabla \cdot \left(\hat{\Pi} \cdot \vec{U}\right) - \nabla\vec{q}$$

- Perfect gas:

$$p = \rho\tilde{R}T, \quad u = e - \frac{1}{2}\vec{U} \cdot \vec{U}, \quad \vec{q} = \vec{q}_{NS} - \tau\rho\vec{U}\left(\left(\vec{U} \cdot \nabla\right)\vec{U} + p\left(\vec{U} \cdot \nabla\right)\frac{1}{\rho}\right)$$

Regularization parameter τ

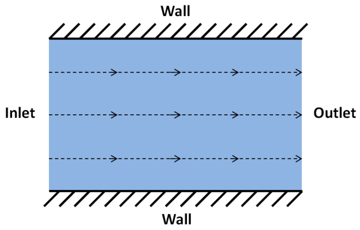
Value of τ coefficient is selected to be equal or less than some characteristic time using speed of sound and grid step Δx:

- $\tau = \dfrac{\mu}{pSc}$,
- $\tau = \alpha^{QGD}\dfrac{\Delta x}{c}$,
- $\tau = \alpha^{QGD}\dfrac{\Delta x}{c} + \dfrac{\mu}{pSc}$.
- $\mu = \mu_0 \left(\dfrac{T}{T_0}\right)^{\omega}$,
- $\mu^{QGD} = \alpha^{QGD}\dfrac{\Delta x}{c} pSc$,
- $\mu^{eff} = \mu + \mu^{QGD}$,
- $\kappa = \dfrac{\mu}{Pr \cdot (\gamma - 1)}$.

Initial and boundary conditions

Types of boundary conditions:

- wall
- inlet
- outlet

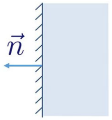


The mathematical description of these BCs within the framework of regularized equations is set out on next slides.

Wall

Velocity condition	slip	$\vec{n} \cdot \vec{U}_n = 0, \ \dfrac{\partial \vec{U}_\tau}{\partial \vec{n}} \cdot \vec{\tau} = 0$
	noSlip	$\vec{U} = (0, 0, 0)$
Temperature condition	zeroGradient	$\dfrac{\partial T}{\partial \vec{n}} = 0$
	fixedValue	$T = T_{inlet}$
Pressure condition	qgdFlux/zeroGradient	$\dfrac{\partial p}{\partial \vec{n}} = 0$

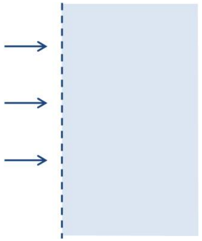
$$\vec{j}_m \cdot \vec{n} =$$
$$= \rho \vec{n} \cdot \vec{U} - \tau \left(div(\rho \vec{U} \otimes \vec{U}) + \nabla p \right) \cdot \vec{n} = 0$$



Inlet

Supersonic inlet – fixed p, T and $\vec{U}$

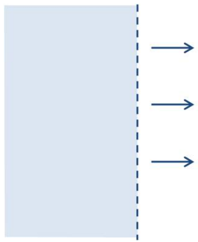
$$p = p_{inlet}$$
$$T = T_{inlet}$$
$$\vec{U} = \vec{U}_{inlet}$$



Outlet

"Soft" BC's for supersonic outlet – zero derivative in normal direction for pressure, temperature and velocity

$$\frac{\partial p}{\partial \vec{n}} = 0, \quad \frac{\partial T}{\partial \vec{n}} = 0, \frac{\partial \vec{U}}{\partial \vec{n}} = 0$$



Keywords in OpenFOAM

Boundary conditions

slip	Provides a slip constraint: $\vec{n}\vec{U}_n = 0,\ \frac{\partial U_\tau}{\partial n}\tau = 0$
noSlip	Fixes the velocity to zero at walls, similar to fixedValue $= 0$
fixedValue	Provides a fixed value constraint, and is the base class for a number of other boundary conditions
zeroGradient	Applies a zero-gradient in normal direction condition
qgdFlux	Specific condition for ∇p: $\rho\vec{n}\cdot\vec{U} - \tau\left(div(\rho\vec{U}\otimes\vec{U}) + \nabla p\right)\cdot\vec{n} = 0$
totalPressure	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.

Part III
Practical part

How to set up case

How to install QGDSolver

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

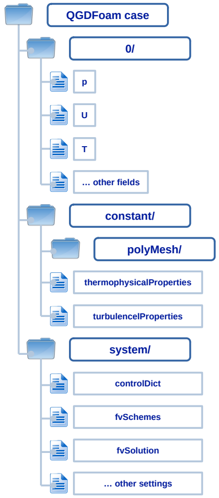
- Download QGDSolver directly from `https://github.com/unicfdlab/QGDsolver/tree/digitef-dev-v2012`
  or using git clone:

```
git clone https://github.com/unicfdlab/QGDsolver.git
cd QGDSolver
git checkout digitef-dev-v2012
```

- Install QGDSolver:

```
./Allwmake
```

QGDFoam case structure



Initial and boundary conditions

To set in the folder "0" pressure "p", velocity "U", temperature "T", "alphaQGD", "ScQGD", "constr.include"

Gas properties

Thermophysical gas properties (density from state equation, 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.

Numerical schemes

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

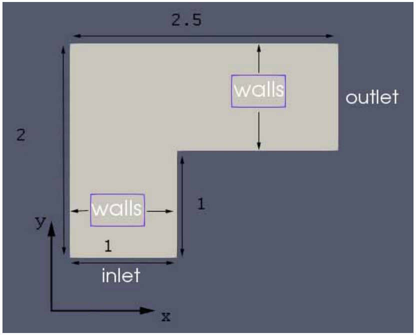
Stages of solution

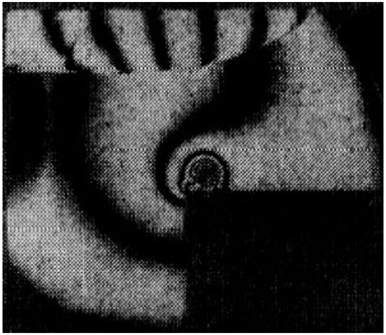
See folder QGDFoam-OFv2012.
Prepare new case folder:

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

Basic case

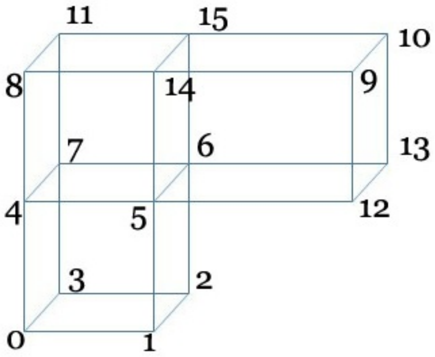
See folder QGDFoam-OFv2012. The case **L–channel** is in the folder **cases**/
Sides 1, 1, 1.5, 1, 2.5, 2.





Schliren image

Mesh generation



Command: blockMesh

Mesh generation

See file system/*blockMeshDict*

Set scale:

```
convertToMeters 1;
```

Set vertices:

```
vertices
(
    (0 0 0.04)     //0    (2.5 2 0.04)     //9
    (1 0 0.04)     //1    (2.5 2 -0.04)    //10
    (1 0 -0.04)    //2    (0 2 -0.04)      //11
    (0 0 -0.04)    //3    (2.5 1 0.04)     //12
    (0 1 0.04)     //4    (2.5 1 -0.04)    //13
    (1 1 0.04)     //5    (1 2 0.04)       //14
    (1 1 -0.04)    //6    (1 2 -0.04)      //15
    (0 1 -0.04)    //7
    (0 2 0.04)     //8
);
```

Mesh generation

Create boxes:

```
blocks
(
    hex (3 2 6 7 0 1 5 4) (50 50 1) simpleGrading (1 1 1)
    hex (7 6 15 11 4 5 14 8) (50 50 1) simpleGrading (1 1 1)
    hex (6 13 10 15 5 12 9 14) (76 50 1) simpleGrading (1 1 1)
);
```

Describe boundaries:

```
boundary
(
  inlet
  {
      type patch;
      faces
      ( (3 2 1 0) );
  }
```

Mesh generation

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

Command: blockMesh

Mesh generation

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

Command: blockMesh

Keywords in QGDFoam

Example of keywords for these boundary conditions in OpenFOAM.
For example, set pressure on the wall

```
wall
{
    type    zeroGradient;
}
```

or set a fixed velocity for the inlet:

```
inlet
{
    type     fixedValue;
    value    uniform (0 1.2 0);
}
```

Initial and boundary conditions

See folder 0/

Name	U, m/s	p, Pa	T, K	Sc
internalField	uniform (0 0 0)	uniform 0.7142	uniform 1.0	1.0
inlet	1. fixedValue (0 1.2 0) 2. fixedValue (0 1.7 0)	0.7142	1.0	1.0
outlet	zeroGradient	zeroGradient	zeroGradient	1.0
walls	slip	zeroGradient	zeroGradient	0.0

alphaQGD

See file alphaQGD in folder 0/

```
dimensions [0 0 0 0 0 0 0];
internalField uniform 0.5;
boundaryField
{
    inlet
    {
        type     calculated;
        value    $internalField;
    }
    outlet
    {
        type     calculated;
        value    $internalField;
    }
}
```

alphaQGD

See file alphaQGD in folder 0/

```
        walls
        {
            type     calculated;
            value    $internalField;
        }
        sides
        {
            type     empty;
        }
    }
```

ScQGD

See file ScQGD in folder 0/

```
dimensions [0 0 0 0 0 0 0];
internalField uniform 1.0;
boundaryField
{
    inlet
    {
        type     calculated;
        value    $internalField;
    }
    outlet
    {
        type     calculated;
        value    $internalField;
    }
```

ScQGD

See file ScQGD in folder 0/

```
        walls
        {
            type      fixedValue;
            value     uniform 0.0;
        }
        sides
        {
            type      empty;
        }
    }
```

U

See file U in folder 0/

```
dimensions [0 1 -1 0 0 0 0];
internalField uniform (0 0 0);
boundaryField
{
    inlet
    {
        type      fixedValue;
        value     uniform (0 1.2 0);
    }
    outlet
    {
        type      zeroGradient;
    }
```

U

See file U in folder 0/

```
        walls
        {
                type    slip;
        }
        sides
        {
                type    empty;
        }
    }
```

Physical properties

See file *thermophysicalProperties* in folder constant/

```
thermoType
{
        type hePsiQGDThermo;
        mixture pureMixture;
        transport const;
        thermo hConst;
        equationOfState perfectGas;
        specie specie;
        energy sensibleInternalEnergy;
}
```

Physical properties

See file *thermophysicalProperties* in folder constant/

```
mixture
{
    specie
    {
        nMoles 1;
        molWeight 11640.3;
    }
    thermodynamics
    {
        Hf 0;
        Sf 0;
        Cp 2.5;
        Tref 0;
    }
```

Physical properties

See file *thermophysicalProperties* in folder constant/

```
        transport
        {
            mu 0.0;
            Pr 0.667;
        }
    }
```

τ calculation

See file *thermophysicalProperties* in folder constant/

```
QGD
{
    implicitDiffusion true; // approximation of viscous terms
    QGDCoeffs constScPrModel1;
    constScPrModel1Dict
        {
            ScQGD 1;
            PrQGD 1;
        }
}
```

QGDCoeffs

File *thermophysicalProperties* in folder constant/

QGDCoeffs	Formulas
constScPrModel1	$\tau = \alpha^{QGD}\dfrac{\Delta x}{c}$ $\mu \longrightarrow \mu + \mu^{QGD}$ $\mu^{QGD} = \tau p Sc^{QGD}$
constScPrModel2	$\tau = \alpha^{QGD}\dfrac{\Delta x}{c} + \dfrac{\mu}{pSc}$ $\mu \longrightarrow \mu + \mu^{QGD}$ $\mu^{QGD} = \alpha^{QGD}\dfrac{\Delta x}{c} p Sc^{QGD}$

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 $\nabla \cdot \left(\frac{\mu}{\rho_0 Pr}\nabla T\right)$

Time settings

See system/*controlDict* to create time settings:

- start time of calculations

```
startTime    0;
```

- end time of calculations

```
endTime    4.5;
```

- start time step interval

deltaT $1e^{-8}$;

- write interval

```
writeInterval    0.1;
```

Time settings

See system/*controlDict* to create time settings:

- adjusting the time steps to coincide with the writeInterval

```
writeControl    adjustableRunTime;
```

- to adjust the time step during the simulation

```
adjustTimeStep    yes;
```

- maximum Courant number

```
maxCo    0.2;
```

Numerical schemes settings. Running

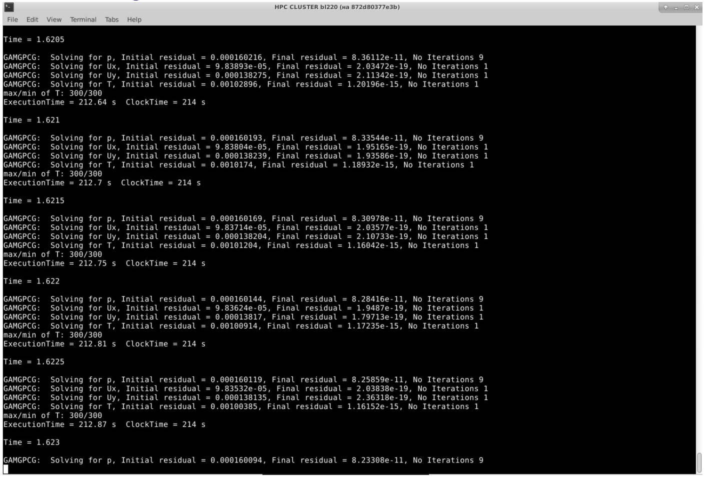
See file system/*fvSchemes* and system/*fvSolution*.
The user specifies the choice of finite volume schemes in the *fvSchemes* dictionary. In file *fvSchemes* you can see that we use only central difference scheme.
The specification of the linear equation solvers and tolerances and other algorithm controls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

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

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

Results

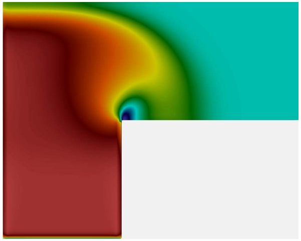
After the entering the QGDFoam command, you will see on the screen:

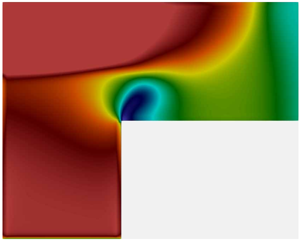

HPC CLUSTER bl220 (на 872d80377e3b)
File Edit View Terminal Tabs Help

```
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: 300/300
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: 300/300
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 = 0.000138204, 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: 300/300
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: 300/300
ExecutionTime = 212.81 s  ClockTime = 214 s

Time = 1.6225

GAMGPCG:  Solving for p, Initial residual = 0.000160119, Final residual = 8.25859e-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: 300/300
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

$\vec{U}_{inlet} = (0,\ 1.2,\ 0)$

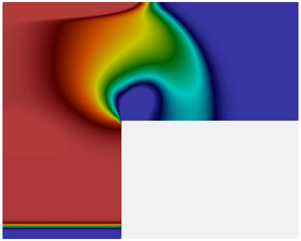

$t = 1.3$

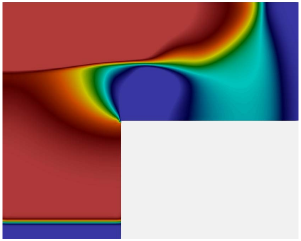

$t = 2.0$

$\alpha = 0.5$
$\tau \sim 0.008$
$\Delta t = 0.0018$

Results

$\vec{U}_{inlet} = (0,\ 1.7,\ 0)$


$t = 1.3$


$t = 1.8$

$\alpha = 0.5$
$\tau \sim 0.005$
$\Delta t = 0.0015$

Conclusions

- We look how QGDFoam for OpenFOAM v2012 works
- We learned how to set initial and boundary conditions for QGDFoam
- We studied how to solve cases step-by-step on the basic example

Some questions?

Contacts

Telegram:
`https://t.me/qgd_qhd`

GitHub:
`https://github.com/unicfdlab/QGDsolver`

Training Tracks:
`https://github.com/unicfdlab/TrainingTracks`

- libAcoustics
- QGDSolvers
- Simple FSI training track for OpenFOAM
- Implementation of the solver for coupled heat transfer in gas and solid