ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/282272911

OpenFOAM step by step tutorial ('beginners' version)

Technical Report · July 2021

CITATION
1

READS
90,029

1 author:

Victor Pozzobon
CentraleSupélec

38 PUBLICATIONS **327** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project

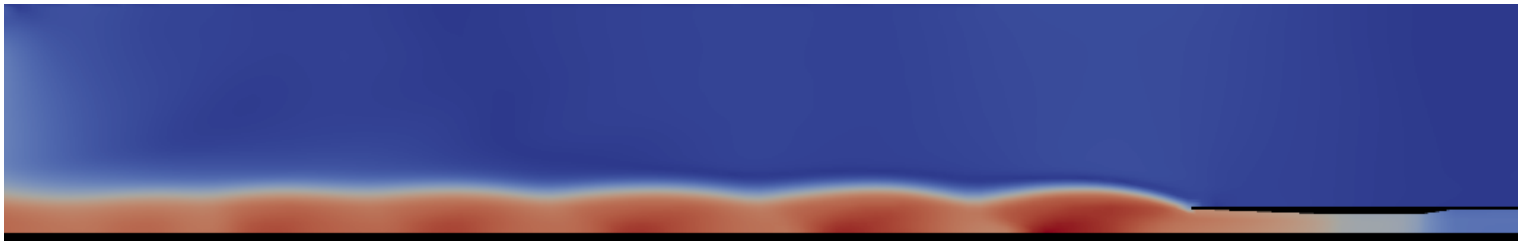
OpenFOAM Tutoring View project

Project

Development, modeling and optimization of an eco-efficient process for producing hydrogen from syngas, by gas-liquid absorption of CO2 by membrane contactor
View project

All content following this page was uploaded by Victor Pozzobon on 16 July 2021.

The user has requested enhancement of the downloaded file.



OpenFOAM tutorial

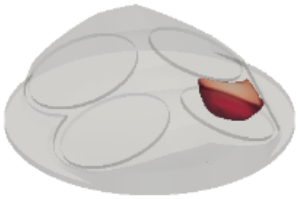
Discover it, tame it, use it

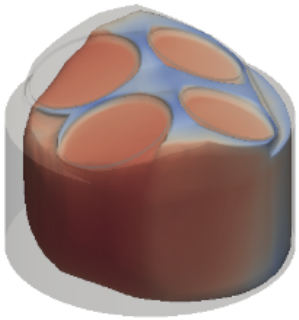
Greetings everyone

This tutorial is a brief introduction to OpenFOAM (Open Field Operation And Manipulation). This document is a step by step guide. It was designed to be used on its own, there should be no need for a presenter (myself). The following exercices were developed using OpenFOAM 2106 (ESI / openfoam.com) version (changes may appear in superior versions or with foundation version).

Before diving into this tutorial, one may ask: What are the main advantages of OpenFOAM? Here are a few reasons why:

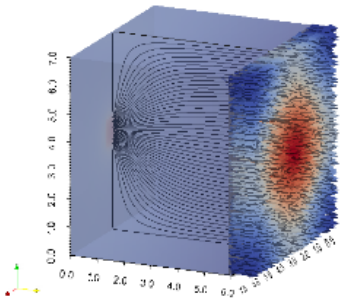
- free
- powerful
- opensource
- versatile
- has no GUI (in my opinion, it helps to focus on the physics, the equation and numerical methods)





$$\frac{\partial \rho \mathbf{U}}{\partial t} + \nabla \cdot \phi \mathbf{U} - \nabla \cdot \mu \nabla \mathbf{U} = -\nabla p$$

```
solve
(
      fvm::ddt(rho,U)
    + fvm::div(phi,U)
    - fvm::laplacian(mu,U)
      ==
    - fvc::grad(p)
);
```



Happy foaming!

Disclaimer

"This offering is not approved or endorsed by OpenCFD Limited, the producer of the OpenFOAM software and owner of the OPENFOAM® and OpenCFD® trade marks."

How to use this tutorial

Almost every command will passed through the terminal. As OpenFOAM has no GUI, we will modify files. For example, when you see:

```
$ gedit system/controlDict
```

You type it in the terminal. Then, you modify the file so that its content is the same as the one in the frame below. Then, you save it and quit.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1606+                                |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     compressibleInterDyMFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         5;

deltaT          1e-6;

writeControl    adjustableRunTime;

writeInterval   1e-1;

purgeWrite      3;

writeFormat     binary;

writePrecision  6;
```

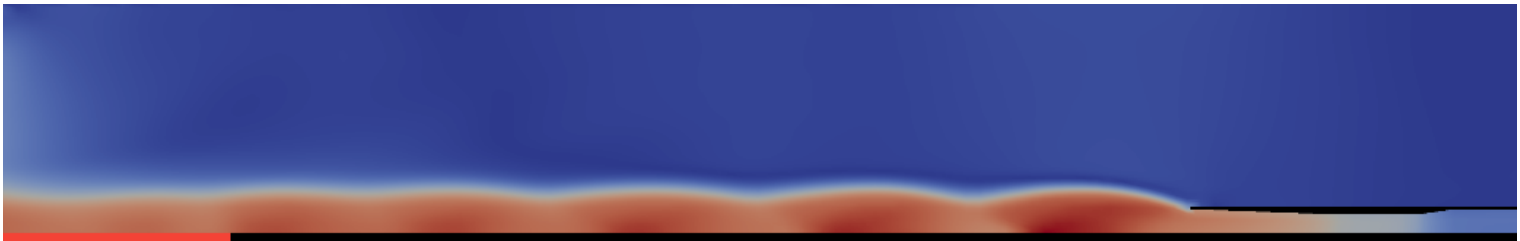
Battle plan

With the 10 coming exercises, we are going to progressively learn how to use OpenFOAM. Here is the roadmap:

- Ex. 1 - Cavity: getting started
- Ex. 2 - Square tube: mesh manipulation & steady state
- Ex. 3 - 1D beam: 1D problem & transient
- Ex. 4 - 2D flat plate: 2D problem
- Ex. 5 - 2D pipe: axisymmetric problem
- Ex. 6 - Tubular reactor: solver modifying (1/2)
- Ex. 7 - Tubular reactor: solver modifying (2/2)
- Ex. 8 - Heated pipe: custom boundary condition
- Ex. 9 - Cake backing: field affectation
- Ex. 10 - Tank filling: importing a mesh & running in parallel

From Ex. 2 to Ex. 5, we are going to learn how to use existing solvers in different geometrical configurations by setting 3D, 2D planar, 2D axisymmetrical and 1D cases. We are also going to explore postprocessing software basic functions.
Ex. 6 to 9 will deal with solver (equations we solve) specification.
Ex. 10 will cover some additional techniques, such as parallel runs.

I hope you will enjoy it!



Ex. 1 - Cavity

Objectives

They are quite simple:

- Accessing your work directory
- Accessing OpenFOAM tutorials
- Understanding case architecture
- Running your first OpenFOAM case
- Postprocessing with *ParaFoam* (*Paraview* with OpenFOAM reader module): data modifying, line date / point data extraction

Accessing your working directory and copying the case

To access your directory type in a terminal:

```
$ run
```

To get the full path type:

```
$ pwd
```

You will see:

```
$ /home/YourName/OpenFOAM/YourName-OpenFOAMVersion/run
```

To copy official tutorials to your run:

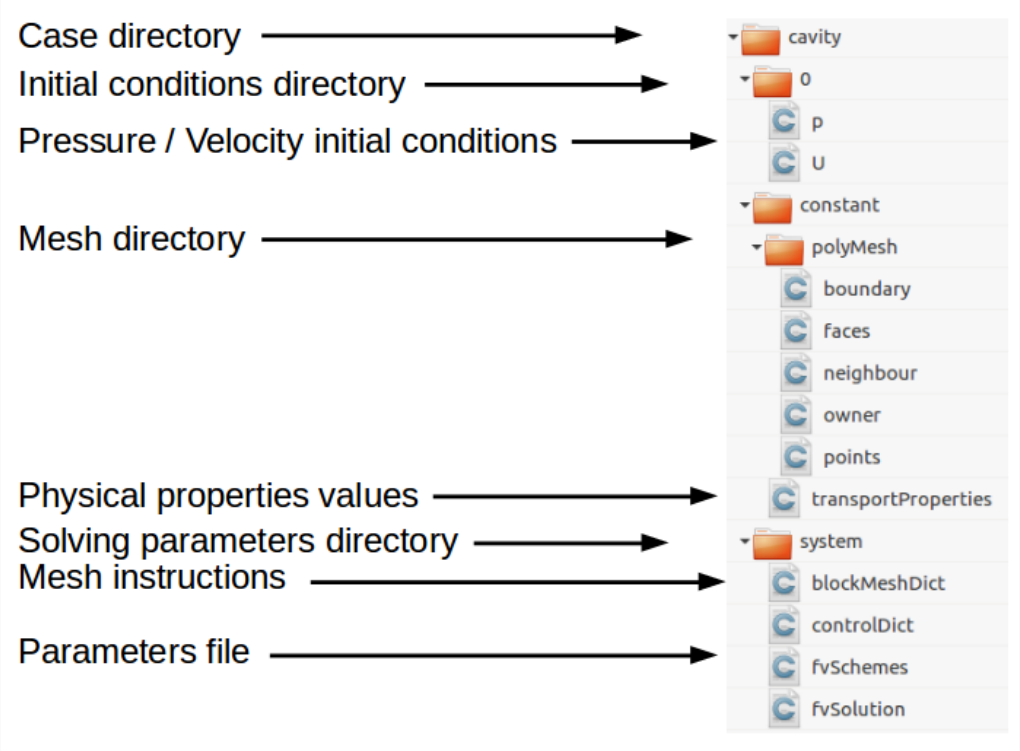
```
$ cp -r $FOAM_TUTORIALS Tutorials
```

Go to the case directory:

```
$ cd Tutorials/incompressible/icoFoam/cavity/cavity
```

Case architecture

A case is usually composed of three directories. Each dedicated to a specific part of the case.



Run the case

Once in *'cavity'* directory, create the mesh:

```
$ blockMesh
```

Call the solver:

```
$ icoFoam
```

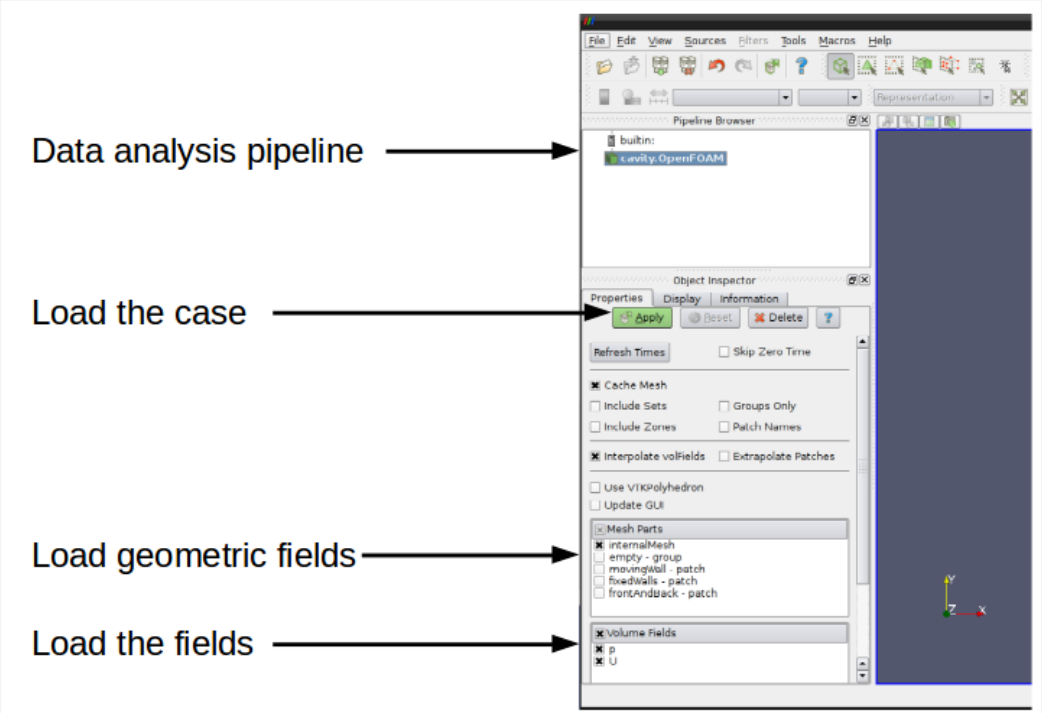
(you can note that the solver created several directories e.g. *'0.1'*. These directories contain data for a given time)

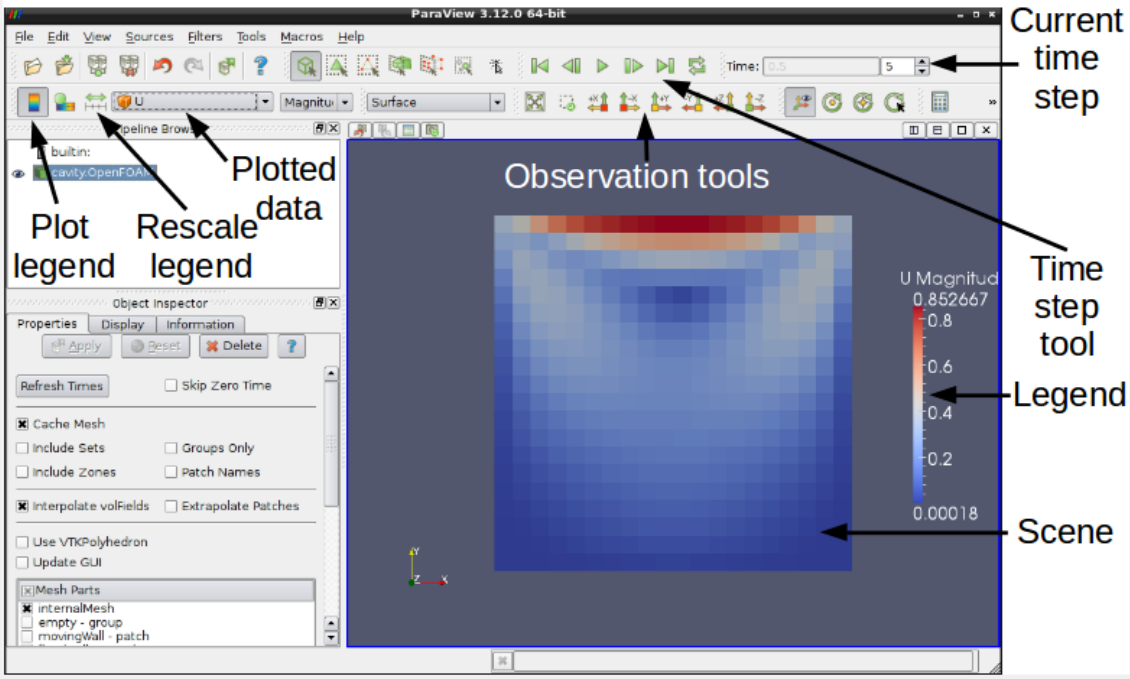
Process the case

Start the postprocessing software:

```
$ paraFoam
```

Here are its main features:





Make it look shinier:

```
$ Filters / Alphabetical / Cell data to point data
```

Calculate something:

```
$ Filters / Common / Calculator
```

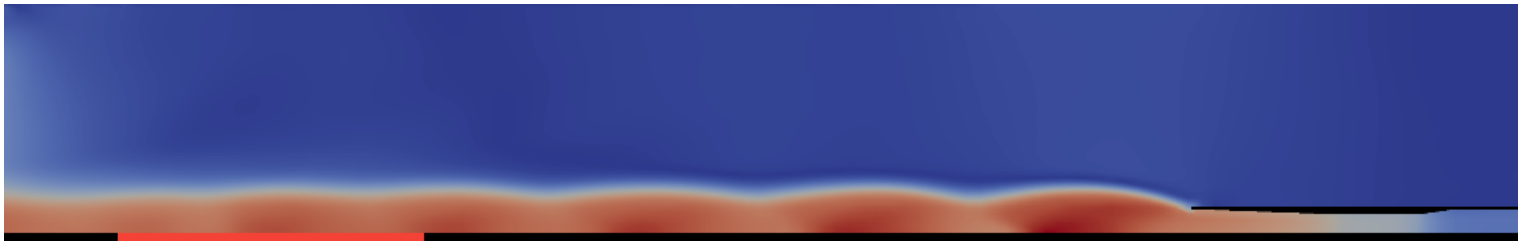
Extract profiles:

```
$ Filters / Data Analysis / Plot Over Line
```

Extract histories: `$ Filters / Data Analysis / Plot Selection Over Time`

It's over

That is all for the first exercice. The aim was to give you a quick overview of an OpenFOAM worflow. We are now going to learn how to manipulate the mesh.



Ex. 2 - Square tube

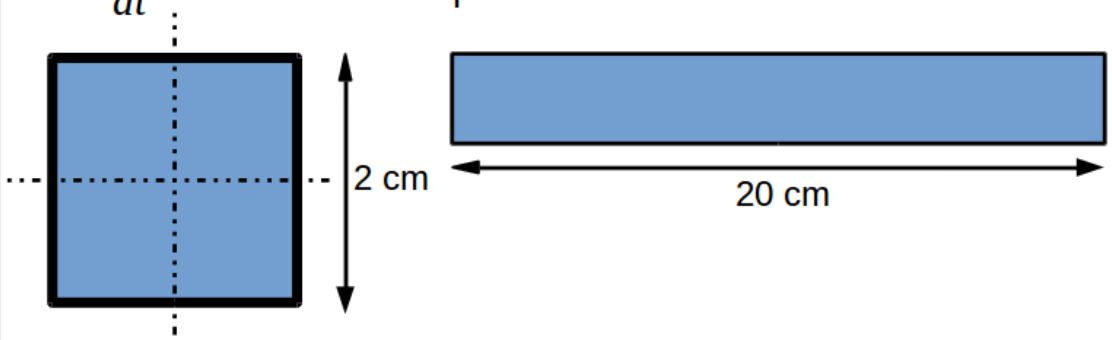
Objectives

- Creating your own case
- Creating a mesh
- Specifying initial / boundary conditions
- Solving a steady state problem
- Postprocessing with *ParaFoam*: integrating variable / checking mass conservation

Case setup

Solving steady state incompressible Navier-Stokes equation, in a fourth of a square pipe.

$$\frac{d\vec{U}}{dt} + \vec{U}\nabla.\vec{U} = -\nabla\frac{p}{\rho} + \eta\nabla^2\vec{U}$$



Setup a new case

Go to your *'run'* directory:

```
$ run
```

Copy an existing case:

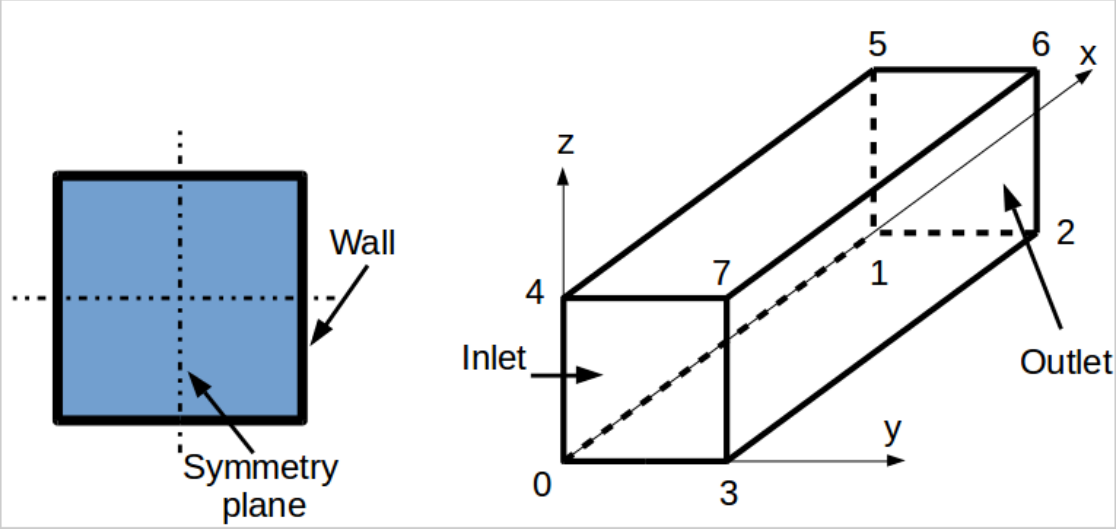
```
$ cp -r $FOAM_TUTORIALS/incompressible/simpleFoam/pitzDaily/ Ex2
```

Move to the case directory:

```
$ cd Ex2
```

Create a new mesh

The mesh is going to be a fourth of a square tube with symmetry plane. Points have to be prescribed in order to build a mesh (consider doing a sketch, like the one below).



Now, open the mesh file:

```
$ gedit system/blockMeshDict
```

Here is how it is constituted:

```
1 /*--------------------------------*- C++ -*----------------------------------*\
2 | =========                 |
3 | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4 |  \\    /   O peration     | Version:  plus                                  |
5 |   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
6 |    \\/     M anipulation  |
7 \*---------------------------------------------------------------------------*/
8 FoamFile
9 {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     object      blockMeshDict;
14 }
15 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
16
17 scale 0.01;
18
19 vertices
20 (
21         (0 0 0)
22         (20 0 0)
23         (20 1 0)
24         (0 1 0)
25         (0 0 1)
26         (20 0 1)
27         (20 1 1)
28         (0 1 1)
29 );
30
31 blocks
32 (
33     hex (0 1 2 3 4 5 6 7) (100 10 10) simpleGrading (1 3 3)
34 );
35
36 edges
37 (
38 );
39
```

Scaling factor

Block division along axis

Points locations

Division grading

Points needed to build a block

```
blockMeshDict ×   p ×   U ×
40 boundary
41 (
42     inlet
43     {
44         type patch;
45         faces
46         (
47             (0 3 7 4)
48         );
49     }
50     outlet
51     {
52         type patch;
53         faces
54         (
55             (1 2 6 5)
56         );
57     }
58     wall
59     {
60         type wall;
61         faces
62         (
63             (0 1 2 3)
64             (0 1 5 4)
65         );
66     }
67     symmetryWall
68     {
69         type symmetry;
70         faces
71         (
72             (3 2 6 7)
73             (4 5 6 7)
74         );
75     }
76
77 );
```

Face name

Face type

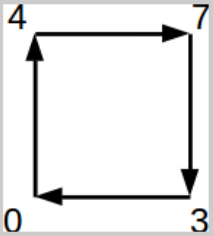
Point sequence delimiting the face

Finally, modify it according to the content below.

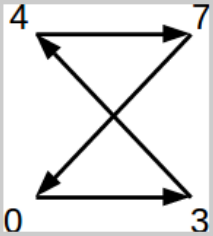
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

scale 0.01;

vertices
(
        (0 0 0)
        (20 0 0)
        (20 1 0)
        (0 1 0)
        (0 0 1)
        (20 0 1)
        (20 1 1)
        (0 1 1)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (100 10 10) simpleGrading (1 3 3)
);

edges
(
);

boundary
(
    inlet
    {
        type patch;
        faces
        (
                (0 3 7 4)
        );
    }
    outlet
    {
        type patch;
        faces
        (
                (1 2 6 5)
        );
    }
    wall
    {
        type wall;
        faces
        (
                (0 1 2 3)
                (0 1 5 4)
        );
    }
    symmetryWall
    {
        type symmetry;
        faces
        (
                (3 2 6 7)
                (4 5 6 7)
        );
    }
);

// ************************************************************************* //
```

Beware, the points are to be prescribed in order not to back cross a line. For example, for the *'inlet'* you should:

Do.



Don't.



Compile your mesh:

```
$ blockMesh
```

Check its quality:

```
$ checkMesh
```

Turbulence model and parameters

Turbulence models are specified in *'turbulenceProperties'* file. In this case, there is no turbulence, hence set the model to *'laminar'*:

```
$ gedit constant/turbulenceProperties
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType laminar;

RAS
{
    // Tested with kEpsilon, realizableKE, kOmega, kOmegaSST, v2f,
    // ShihQuadraticKE, LienCubicKE.
    RASModel        kEpsilon;

    turbulence      on;

    printCoeffs     on;
}

// ************************************************************************* //
```

Because we disabled the turbulence, there will be no need to specify: *epsilon, k, f, omega, v2, nut* nor *nuTilda* fields. Remove them:

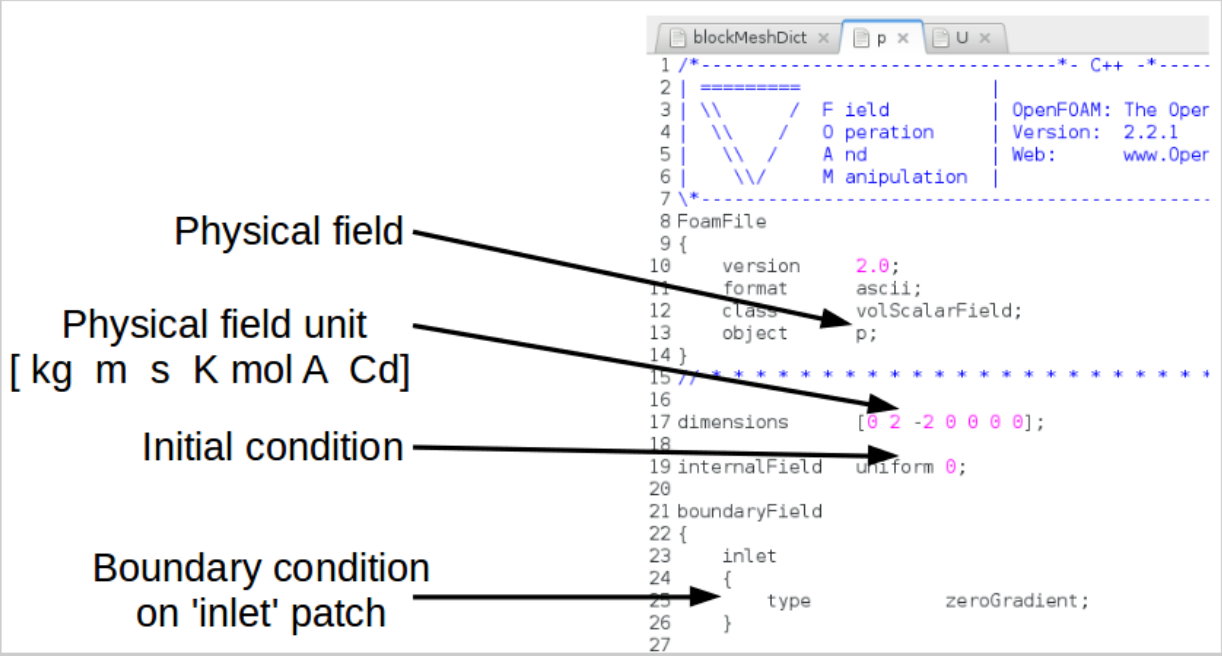
```
$ rm 0/epsilon 0/k 0/nut 0/nuTilda 0/omega
```

Initial and boundary conditions

Open initial / boundary conditions files in the *'0'* directory:

```
$ gedit 0/p 0/U
```

Here is how they are constructed:



Now modify them:

```
/*-------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    wall
    {
        type            zeroGradient;
    }

    symmetryWall
    {
        type            symmetry;
    }

}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (0.005 0 0);
    }

    outlet
    {
        type            zeroGradient;
    }

    wall
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    symmetryWall
    {
        type            symmetry;
    }

}

// ************************************************************************* //
```

Specifying iteration number

Open case control file:

```
$ gedit system/controlDict
```

Erase *'functions'* and below. We do not need that for our exercise.
In this case, *'endTime'* is the maximum number of iteration (convergence should be reached before). *'writeInterval'* is the frequency at which the code will write intermediate results on the disk.
You can also erase *'streamlines'* file in *'system'* folder:

```
$ rm system/streamlines
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     simpleFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         100;

deltaT          1;

writeControl    timeStep;
```

```
writeInterval   10;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

// ************************************************************************* //
```

Run the case

The mesh has already been built, so let's run the case:

```
$ simpleFoam
```

(it should converge in about 22 iterations)
Then, have a look at the results:

```
$ paraFoam
```

Postprocessing

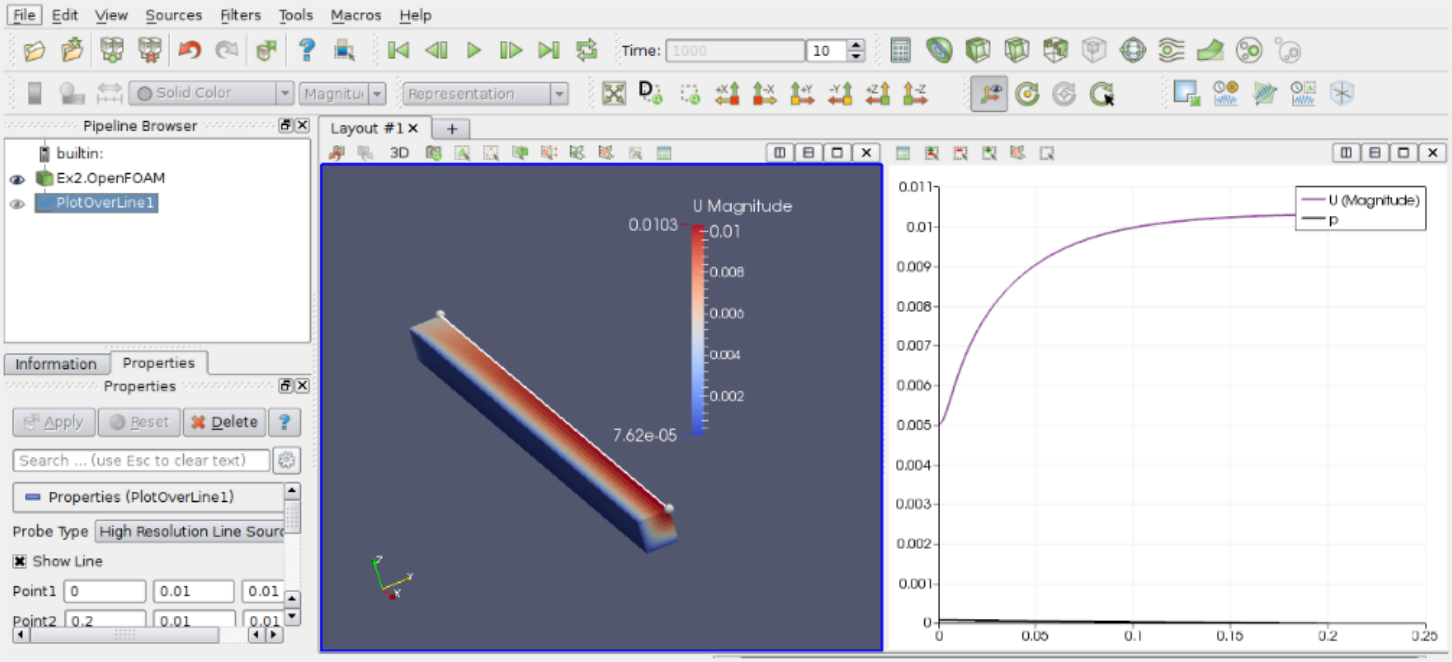
Check mass conservation:

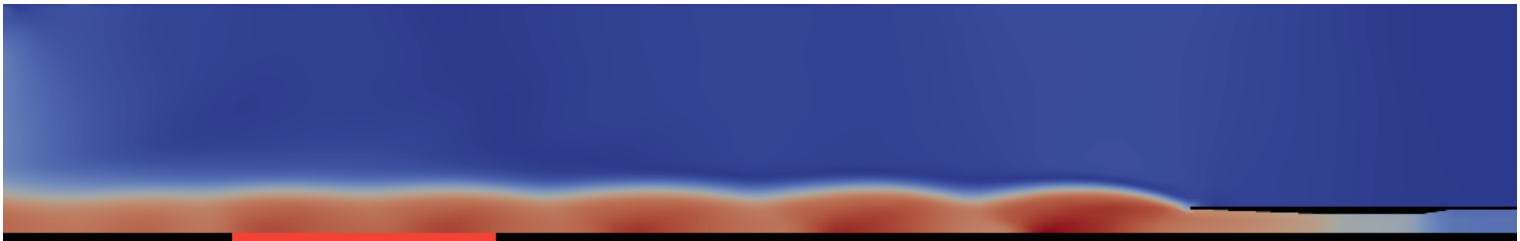
```
$ Filters / Common / Slice
```

```
$ Filters / Data Analysis / Integrate Variables
```

Check velocity profile:

```
$ Filters / Data Analysis / Plot Over Line
```





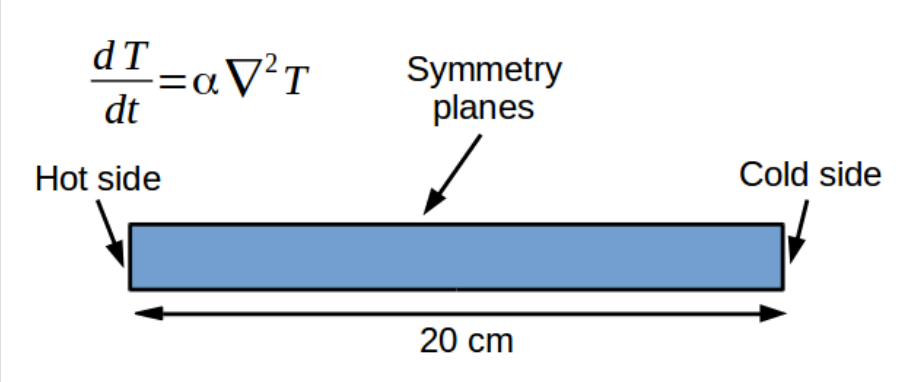
Ex. 3 - 1D beam

Objectives

- Setting a 1D case
- Solving a transient problem
- Modifying physical property value
- Postprocessing with *ParaFOAM*: importing variables (other than p and U)

Case setup

Solving transient heat conduction in a 1D beam.



Setup a new case

Reach *'run'* directory:

```
$ run
```

Copy an existing heat transfer case:

```
$ cp -r $FOAM_TUTORIALS/basic/laplacianFoam/flange Ex3
```

Move to the case directory:

```
$ cd Ex3
```

Clean the case directory:

```
$ rm Allclean Allrun Allrun-parallel
```

```
$ mv 0.orig 0
```

Copy Ex2 mesh:

```
$ cp ../Ex2/system/blockMeshDict system/.
```

Modify the mesh

Modify file:

```
$ gedit system/blockMeshDict
```

Build the mesh:

```
$ blockMesh
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

scale 0.01;

vertices
(
        (0 0 0)
        (20 0 0)
        (20 1 0)
        (0 1 0)
        (0 0 1)
        (20 0 1)
        (20 1 1)
        (0 1 1)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (400 1 1) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
    hotSide
    {
        type patch;
        faces
        (
            (0 3 7 4)
        );
    }
    coldSide
    {
        type patch;
        faces
        (
            (1 2 6 5)
        );
    }
    symmetryPlanes
    {
        type symmetry;
        faces
        (
            (3 2 6 7)
            (4 5 6 7)
            (0 1 2 3)
            (0 1 5 4)
        );
    }
);

// ************************************************************************* //
```

Initial and boundary conditions

Open and modify initial / boundary conditions file for the temperature field:

```
$ gedit 0/T
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 273;

boundaryField
{

    hotSide
    {
        type            fixedValue;
        value           uniform 373;
    }

    symmetryPlanes
    {
        type            symmetry;
    }

    coldSide
    {
        type            fixedValue;
        value           uniform 273;
    }
}

// ************************************************************************* //
```

Modifying physical property value

Modify the thermal diffusion coefficient in *'transportProperties'* file:

```
$ gedit constant/transportProperties
```

This file stores the values of physical parameters that are constant throughout the run. The units of the physical properties can be specified using the following syntax:
Property dimension: [kg m s K mol A Cd], so [0 2 -1 0 0 0 0] is m^2/s.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

DT              DT [0 2 -1 0 0 0 0] 1e-05;

// ************************************************************************* //
```

Running the case

Modify control dictionary:

```
$ gedit system/controlDict
```

Run the case:

```
$ laplacianFoam
```

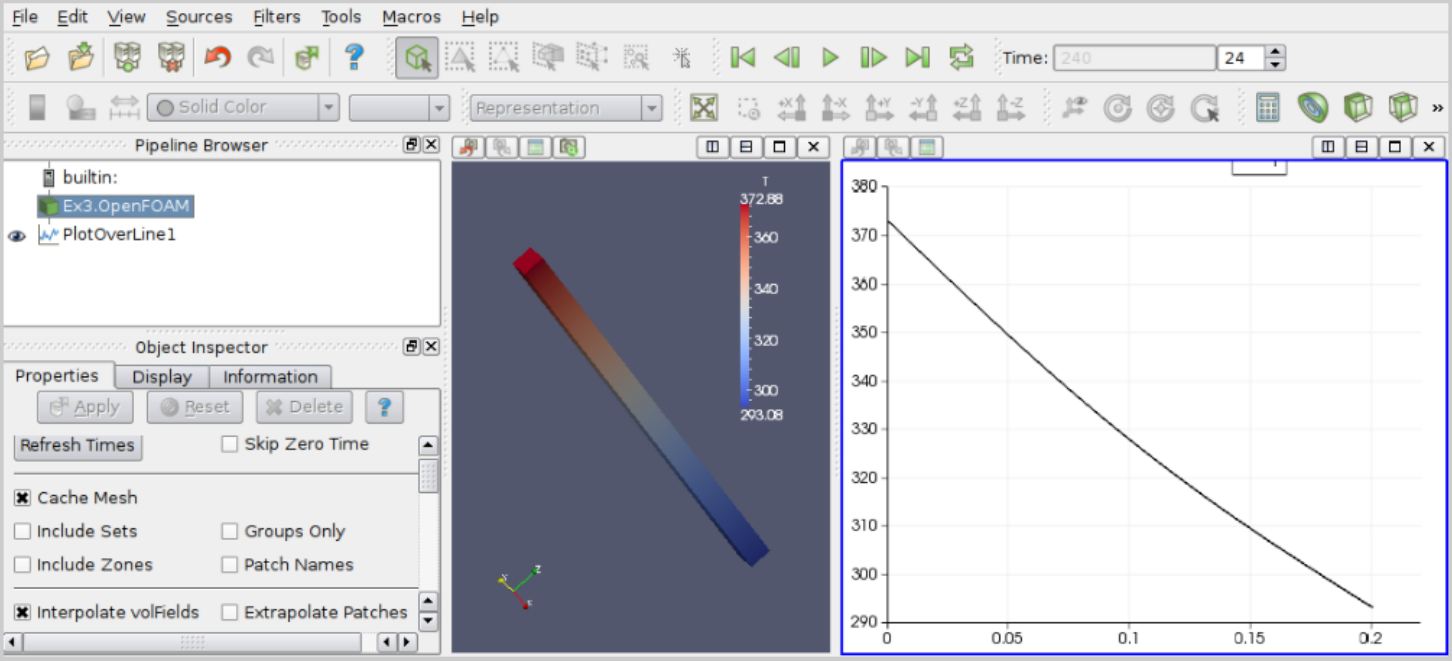
Postprocess the case:

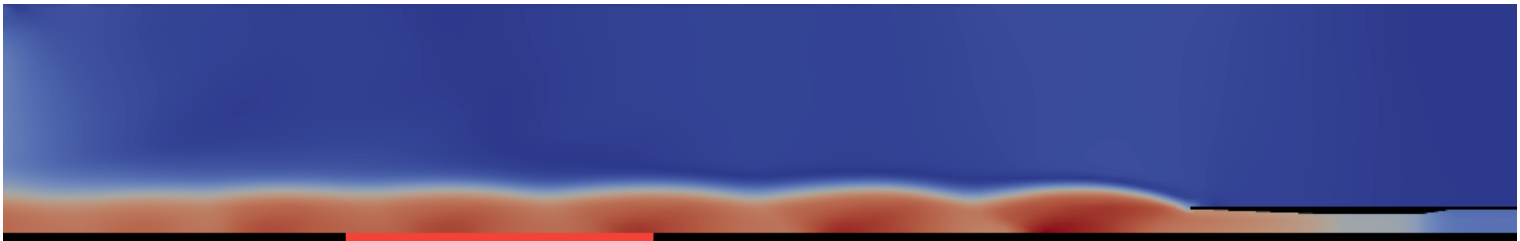
```
$ paraFoam
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     laplacianFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         300;

deltaT          0.1;

writeControl    runTime;

writeInterval   10;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;


// ************************************************************************* //
```

Postprocessing

Check T, as a *'Volume field'* when loading data and have a look at temperature evolution with time.





Ex. 4 - 2D flat plate

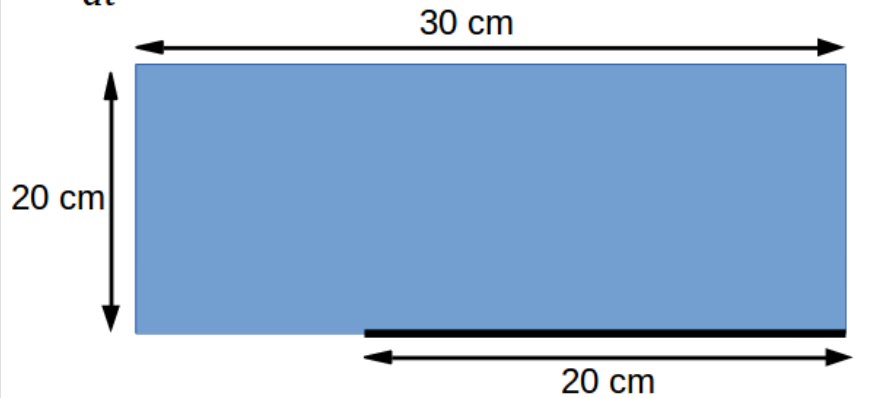
Objectives

- Setting a 2D planar case
- Building a 2 block mesh
- Changing convergence criterion
- Postprocessing with *ParaFoam*: plotting streamlines

Case setup

Solving incompressible laminar air flow above a flat plate, in steady state.

$$\frac{d\vec{U}}{dt}+\vec{U}\nabla.\vec{U}=-\nabla\frac{p}{\rho}+\eta\nabla^2\vec{U}$$



Setup a new case

Reach *'run'* directory:

```
$ run
```

Copy an existing case:

```
$ cp -r Ex2 Ex4
```

```
$ cd Ex4
```

Clean the case directory:

```
$ foamListTimes -rm
```

(this command removes time/iteration folders except the '0' folder)

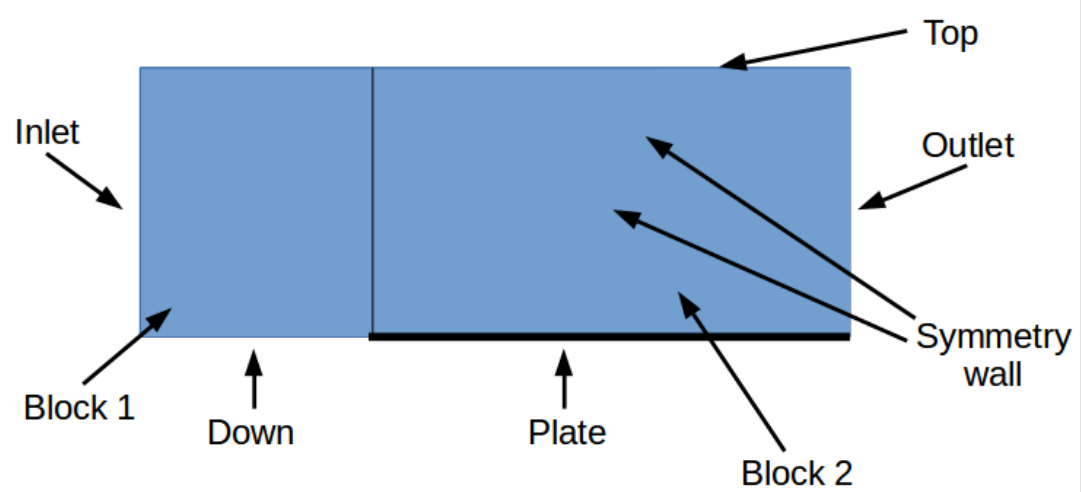
Modify the mesh

We will create a mesh made of two blocks. Modify the mesh file:

```
$ gedit system/blockMeshDict
```

blockMesh will automatically merge the two blocks. Once done modifying the file, build mesh:

```
$ blockMesh
```




```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

scale 0.1;

vertices
(
    // Block 1
    (0 0 0)
    (1 0 0)
    (1 2 0)
    (0 2 0)
    (0 0 0.1)
    (1 0 0.1)
    (1 2 0.1)
    (0 2 0.1)
    // Block 2
    (3 0 0)
    (3 0 0.1)
    (3 2 0)
    (3 2 0.1)
);

blocks
(
    // Block 1
    hex (0 1 2 3 4 5 6 7) (80 80 1) simpleGrading (0.5 2 1)
    // Block 2
    hex (1 8 10 2 5 9 11 6) (160 80 1) simpleGrading (2 2 1)
);

edges
(
);

boundary
(
    inlet
    {
        type patch;
        faces
        (
            (0 3 7 4)
        );
    }
    outlet
    {
        type patch;
```

```
        faces
        (
            (8 9 11 10)
        );
    }
    plate
    {
        type patch;
        faces
        (
            (1 8 9 5)
        );
    }
    symmetryWall
    {
        type empty;
        faces
        (
            (0 1 2 3)
            (4 5 6 7)
            (5 9 11 6)
            (1 8 10 2)
        );
    }
    down
    {
        type patch;
        faces
        (
            (0 1 5 4)
        );
    }
    top
    {
        type patch;
        faces
        (
            (6 11 10 2 )
            (7 6 2 3)
        );
    }
);

// ************************************************************************* //
```

Initial and boundary conditions

Open and modify initial / boundary conditions file for both velocity and pressure fields:

```
$ gedit 0/U 0/p
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [ 0 1 -1 0 0 0 0 ];

internalField   uniform ( 0 0 0 );

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform ( 0.1 0 0 );
    }
    outlet
    {
        type            zeroGradient;
    }
    plate
    {
        type            fixedValue;
        value           uniform ( 0 0 0 );
    }
    symmetryWall
    {
        type            empty;
    }
    "(top|down)"
    {
```

```
        type            zeroGradient;
    }
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [ 0 2 -2 0 0 0 0 ];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }
    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    top
    {
        type            fixedValue;
        value           uniform 0;
    }
    plate
    {
        type            zeroGradient;
    }
    symmetryWall
    {
        type            empty;
    }
    down
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //
```

Modifying kinematic viscosity value

Modify the kinematic viscosity (I like to specify the dimensions):

```
$ gedit constant/transportProperties
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              [0 2 -1 0 0 0 0] 1.5e-05;

// ************************************************************************* //
```

Specifying convergence criteria

Modify pressure / velocity convergence criteria:

```
$ gedit system/fvSolution
```

Each field has an associated solver with its own parameters (tolerance, relative tolerance, preconditionner, ...). They are found in the *'solvers'* subdictionnary. Final residuals can be specified in *'SIMPLE'* subdictionnary. Here, we set individual solver tolerance to 10^{-8} and convergence criteria to 10^{-6} for both U and p fields. Relaxation factors could also be specified, but we are not going to modify them in this case.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-08;
        relTol          0.1;
        smoother        GaussSeidel;
    }

    "(U|k|epsilon|omega|f|v2)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-08;
        relTol          0.1;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;
    consistent      yes;

    residualControl
    {
        p               1e-6;
        U               1e-6;
        "(k|epsilon|omega|f|v2)" 1e-6;
    }
}

relaxationFactors
{
    equations
    {
        U               0.9; // 0.9 is more stable but 0.95 more convergent
        ".*"            0.9; // 0.9 is more stable but 0.95 more convergent
    }
}

// ************************************************************************* //
```

Running the case

Modify control dictionary:

```
$ gedit system/controlDict
```

Run the case:

```
$ simpleFoam
```

Postprocess the case:

```
$ paraFoam
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     simpleFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         200;

deltaT          1;

writeControl    timeStep;

writeInterval   50;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

// ************************************************************************* //
```

Postprocessing

Lower the first scene opacity:

```
$ 'Click on the filter' / Display / Styling / Opacity
```

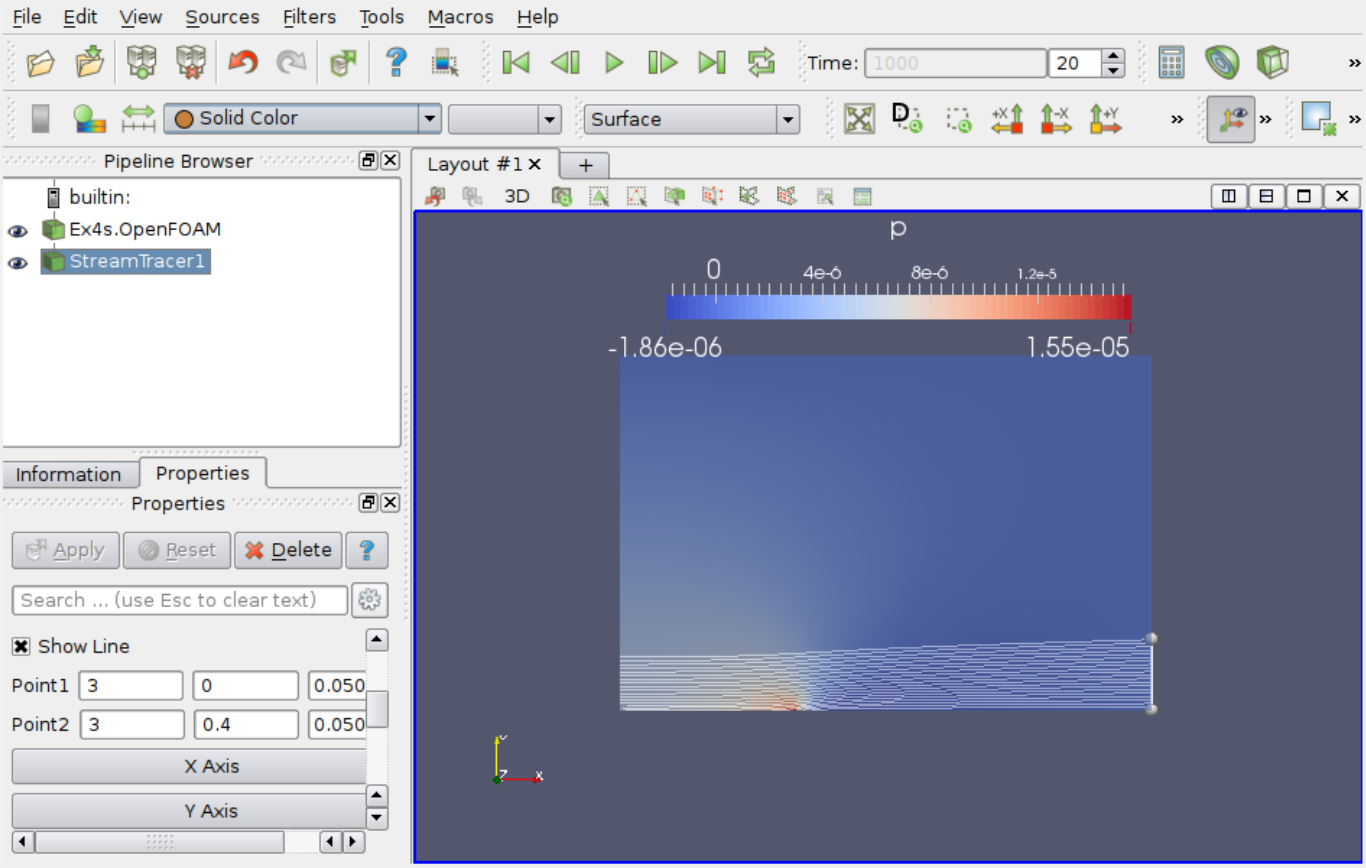
Plot the streamline from the inlet:

```
$ Filters / Common / Stream Tracer
```

Set streamlines' sources as a line, not a point:

```
$ 'Click on stream tracer' / Properties / Seeds / Seed type => High resolution line source
```

Set the line you want using the coordinates



Ex. 5 - 2D pipe

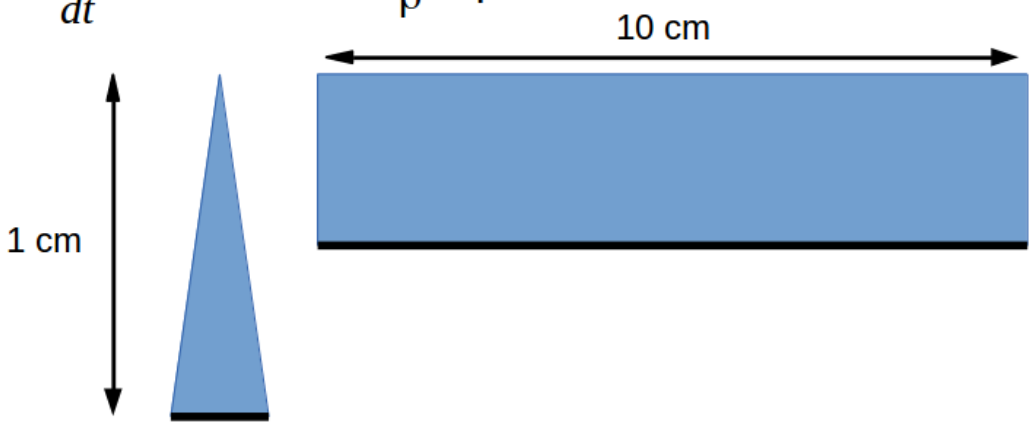
Objectives

- Setting a 2D axi-symmetrical case

Case setup

Solving incompressible laminar flow in a 2D axisymmetrical pipe, in steady state.

$$\frac{d\vec{U}}{dt}+\vec{U}\nabla.\vec{U}=-\nabla\frac{p}{\rho}+\eta\nabla^2\vec{U}$$



Setup a new case

Reach *'run'* directory:

```
$ run
```

Copy an existing case:

```
$ cp -r Ex2 Ex5
```

```
$ cd Ex5
```

Clean the case directory:

```
$ foamListTimes -rm
```

Modify the mesh

The mesh is going to be 5° wedge. The two lateral sides of the wedge are of a special type, i.e. *'wedge'*. Modify the mesh file:

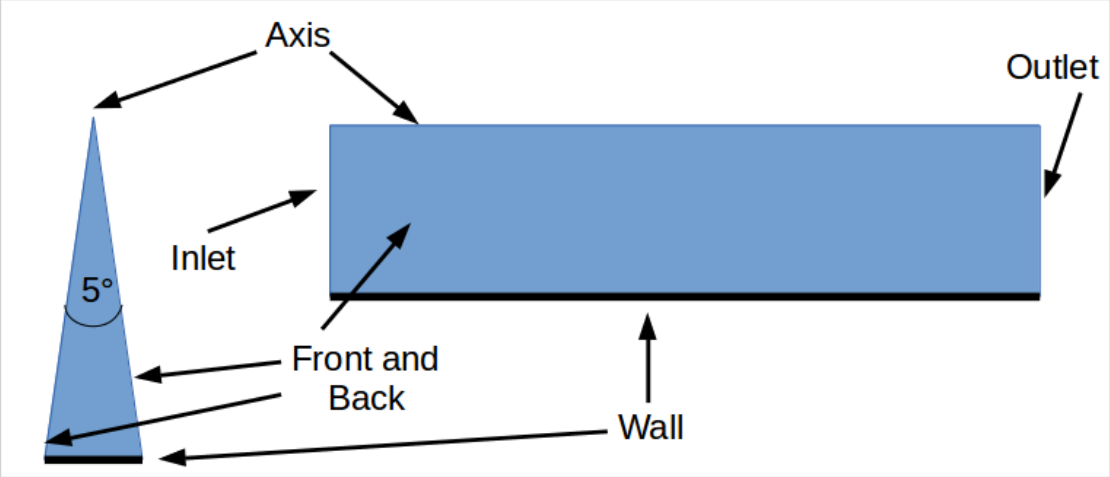
```
$ gedit system/blockMeshDict
```

Stange values (0.9990482216, actually cos(2.5°), and so on ...) are used to draw a wegde, with a 5° total angle (or twice 2.5° if you prefer)
Points 1, 2 and 3 are not used to build the geometry. OpenFOAM uses them to define x, y and z directions :

- point 0 to point 1 = x vector direction
- point 1 to point 2 = y vector direction
- point 0 to point 4 = z vector direction

Once done modifying the file, build mesh:

```
$ blockMesh
```




```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

scale 0.01;

vertices
(
    (0 0 0) //0
    (1 0 0)
    (1 1 0)
    (0 1 0)
    (0 0 10) //4
    (0.9990482216 -0.0436193874 0) //5
    (0.9990482216 0.0436193874 0) //6
    (0.9990482216 -0.0436193874 10) //7
    (0.9990482216 0.0436193874 10) //8
);

blocks
(
    hex (0 5 6 0 4 7 8 4) (30 1 300) simpleGrading (0.1 1 1)
);

edges
(
);

boundary
(
    inlet
    {
        type patch;
        faces
        (
            (0 5 6 0)
        );
    }
```

```
    outlet
    {
        type patch;
        faces
        (
            (4 7 8 4)
        );
    }
    wall
    {
        type wall;
        faces
        (
            (5 6 8 7)
        );
    }
    front
    {
        type wedge;
        faces
        (
            (0 4 7 5)
        );
    }
    back
    {
        type wedge;
        faces
        (
            (0 4 8 6)
        );
    }
    axis
    {
        type empty;
        faces
        (
            (0 4 4 0)
        );
    }
);

// ************************************************************************* //
```

Initial and boundary conditions

Open and modify initial / boundary conditions file for both velocity and pressure fields:

```
$ gedit 0/U 0/p
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (0 0 0.01);
    }

    outlet
    {
        type            zeroGradient;
    }

    wall
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    "(front|back)"
    {
        type            wedge;
    }
```

```
    axis
    {
        type            empty;
    }
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    wall
    {
        type            zeroGradient;
    }

    "(front|back)"
    {
        type            wedge;
    }

    axis
    {
        type            empty;
    }
}

// ************************************************************************* //
```

Running the case

The control dictionary does not need to be modified, you can directly run the case:

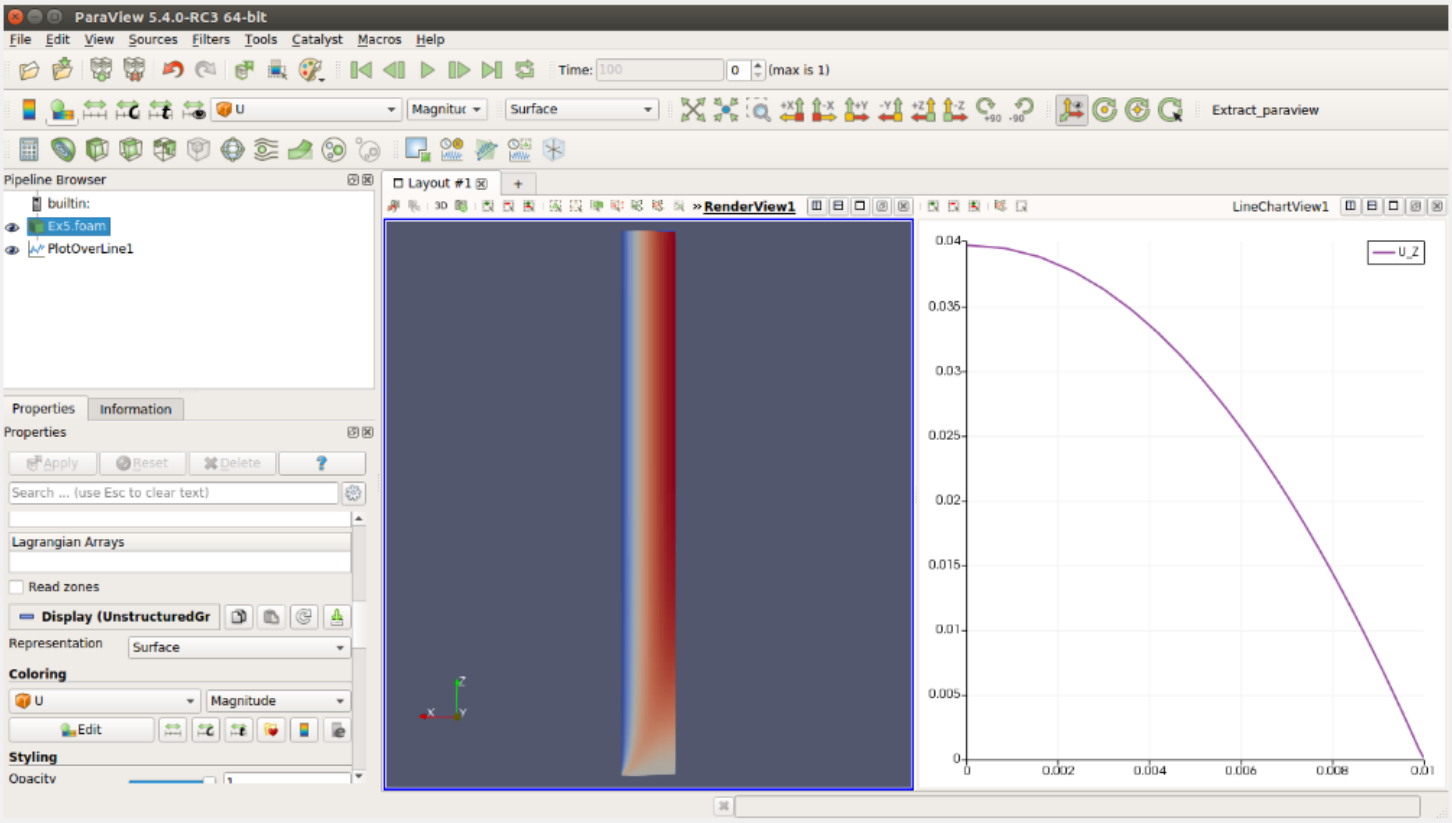
```
$ simpleFoam
```

Postprocess the case:

```
$ paraFoam
```

Postprocessing

E.g. check velocity profile.



Battle plan

With the five first exercises:

- We have used two different built-in solvers (*icoFoam, simpleFoam* and *laplacianFoam*)
- We have set cases in 3D, 2D planar, 2D axi-symmetrical and 1D configurations
- We have learnt how to process data with *Paraview* (*paraFoam*)
- We are going to learn how to solve our own equations and set fancy cases

Ex. 6 - Tubular reactor

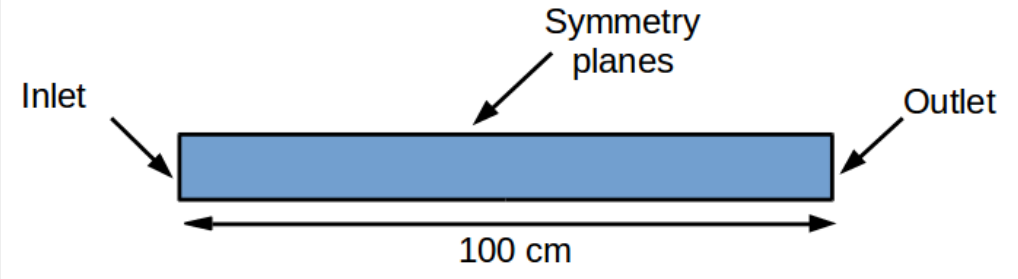
Objectives

- Create our own solver
- Use basic differential operators
- Create the associated case

Case setup

1D tubular reactor with heterogeneous catalysis on porous medium without chemical reaction in steady state / hydrodynamics only.

$$\nabla . \vec{U} = 0 \qquad \vec{U} = \frac{-\kappa}{\mu} \nabla p \qquad \nabla^2 p = 0$$



Create a solver folder

Reach *'run'* directory:

```
$ run
```

Create a solver directory:

```
$ mkdir solvers
```

Copy an existing solver:

```
$ cp -r $FOAM_APP/solvers/basic/laplacianFoam solvers/.
```

Rename the solver directory:

```
$ mv solvers/laplacianFoam solvers/tubeFoam
```

Move to the solver directory:

```
$ cd solvers/tubeFoam
```

Clean the directory:

```
$ wclean
```

```
$ rm -r overLaplacianDyMFoam write.H
```

Rename the file:

```
$ mv laplacianFoam.C tubeFoam.C
```

Modify the compilation file

Modify compilation file:

```
$ gedit Make/files
```

You can specify compilation options:

```
$ gedit Make/options
```

(None need to be applied here)

```
tubeFoam.C

EXE = $(FOAM_USER_APPBIN)/tubeFoam
```

Adding new field to the solver

Fields and physical properties are declared in the *'createFields.H'* file. We are going to modify *'createFields.H'* to create new fields. A pressure field (scalar field) and velocity field (vector field). We will also add physical properties (permeability and viscosity). Now edit the file:

```
$ gedit createFields.H
```

```
    Info<< "Reading field p\n" << endl;

    volScalarField p
    (
        IOobject
        (
            "p",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );

    Info<< "Reading field U\n" << endl;
    volVectorField U
    (
        IOobject
        (
            "U",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );

    Info<< "Reading transportProperties\n" << endl;

    IOdictionary transportProperties
    (
        IOobject
        (
            "transportProperties",
            runTime.constant(),
            mesh,
            IOobject::MUST_READ_IF_MODIFIED,
            IOobject::NO_WRITE
        )
    );

    Info<< "Reading dynamic viscosity mu\n" << endl;

    dimensionedScalar mu("mu", transportProperties);

    Info<< "Reading permeability kappa\n" << endl;

    dimensionedScalar kappa("kappa", transportProperties);
```

Solving a new equation

Modify *'tubeFoam.C'* to specify the equations that will be solved: pressure laplacian equal to 0 and Darcy's law:

```
$ gedit tubeFoam.C
```

(note that the syntax: *'solve (fvm::laplacian(p));'* implicitly set pressure laplacian to 0)

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Application
    tubeFoam

\*---------------------------------------------------------------------------*/

#include "fvCFD.H"
#include "fvOptions.H"
#include "simpleControl.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
    #include "setRootCase.H"

    #include "createTime.H"
    #include "createMesh.H"

    simpleControl simple(mesh);

    #include "createFields.H"
    #include "createFvOptions.H"

    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

    Info<< "\nCalculating pressure distribution\n" << endl;

    while (simple.loop())
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;

        while (simple.correctNonOrthogonal())
        {
            solve
            (
                fvm::laplacian(p) // implicitly equal to 0
            );
        }
        U = -kappa / mu * fvc::grad(p);

        runTime.write();

        Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
            << "  ClockTime = " << runTime.elapsedClockTime() << " s"
            << nl << endl;
    }

    Info<< "End\n" << endl;

    return 0;
}

// ************************************************************************* //
```

Compile the solver

Clean the directory and compile:

```
$ wclean; wmake
```

If you get an error, here is what to do:

- 1. Read the exception message
- 2. Get error file and line number
- 3. Take care of it

```
[vpozzobo@localhost tubeFoam]$ wclean;wmake
Making dependency list for source file tubeFoam.C
SOURCE=tubeFoam.C ;  g++ -m64 -Dlinux64 -DWM_DP -Wall -Wextra -Wno-unused-parameter
e/vpozzobo/OpenFOAM/OpenFOAM-2.2.1/src/finiteVolume/lnInclude -IlnInclude -I. -I/ho
/OpenFOAM-2.2.1/src/OSspecific/POSIX/lnInclude   -fPIC -c $SOURCE -o Make/linux64Gc
tubeFoam.C: In function 'int main(int, char**)':
tubeFoam.C:61:47: erreur: 'Intentional_Error' was not declared in this scope
             U = - Kappa / Mu * fvc::grad(p) * Intentional_Error;
                                               ^
make: *** [Make/linux64GccDPOpt/tubeFoam.o] Erreur 1
```

Set basic case folder

(This fastidious steps can be skipped using *'Ex6_Root'* folder. Just type: `$ cp -r Ex6_Root Ex6` from your *'run'* folder)

Now that the solver has been compiled, let's create the case we are going to solve. Since it is a 1D case, copy Ex3 (1D thermal beam) case directory:

```
$ run
```

```
$ cp -r Ex3 Ex6
```

Go to the case directory, clean it:

```
$ cd Ex6
```

```
$ foamListTimes -rm
```

Then, modifiy the existing mesh:

```
$ gedit system/blockMeshDict
```

Once done modifying the file, compile the mesh:

```
$ blockMesh
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

scale 0.01;

vertices
(
        (0 0 0)
        (20 0 0)
        (20 1 0)
        (0 1 0)
        (0 0 1)
        (20 0 1)
        (20 1 1)
        (0 1 1)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (1000 1 1) simpleGrading (1 1 1)
);

edges
(
);

boundary
```

```
(
    inlet
    {
        type patch;
        faces
        (
            (0 3 7 4)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (1 2 6 5)
        );
    }
    symmetryPlanes
    {
        type symmetry;
        faces
        (
            (3 2 6 7)
            (4 5 6 7)
            (0 1 2 3)
            (0 1 5 4)
        );
    }
);

// ************************************************************************* //
```

Create new files for pressure and velocity fields initial / boundary conditions. A gradient has to be specified because velocity and pressure gradient are linked by Darcy's law:

```
$ mv 0/T 0/p

$ cp 0/p 0/U

$ gedit 0/p 0/U
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{

    inlet
    {
        type            fixedGradient;
        gradient        uniform 5e4;
    }

    symmetryPlanes
    {
        type            symmetry;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
```

```
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{

    inlet
    {
        type            fixedValue;
        value           uniform (0.05 0 0);
    }

    symmetryPlanes
    {
        type            symmetry;
    }

    outlet
    {
        type            zeroGradient;
    }
}


// ************************************************************************* //
```

Specifying the value of the new physical properties

As we declared new physical properties (permeability and dynamic viscosity), we have to specify their value and dimension in *'transportProperties'* dictionary:

```
$ gedit constant/transportProperties
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

mu                      mu [1 -1 -1 0 0 0 0] 1e-3;
kappa                   kappa [0 2 0 0 0 0 0] 1e-9;

// ************************************************************************* //
```

Modifying numerical schemes and solver parameters

Ex3 only used a temperature fileld called *'T'*. Here, we solved a pressure field named *'p'*. Thus, we have to define the numerical details associated with solving the new governing equation. Numerical schemes are specified in *'fvSchemes'* file, while solver technical details can be modified in *'fvSolution'* file. Edit both of them:

```
$ gedit system/fvSchemes
```

```
$ gedit system/fvSolution
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
```

```
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         steadyState;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;

    residualControl
    {
        p               1e-6;
    }
}

// ************************************************************************* //
```

Run the case

Modify the simulation control parameters and run the case:

```
$ gedit system/controlDict
```

```
$ tubeFoam
```

In this case, the solver is only calculating a Laplacian operator. Thus, a 10^{-6} residue is reached quite fast (2 iterations).

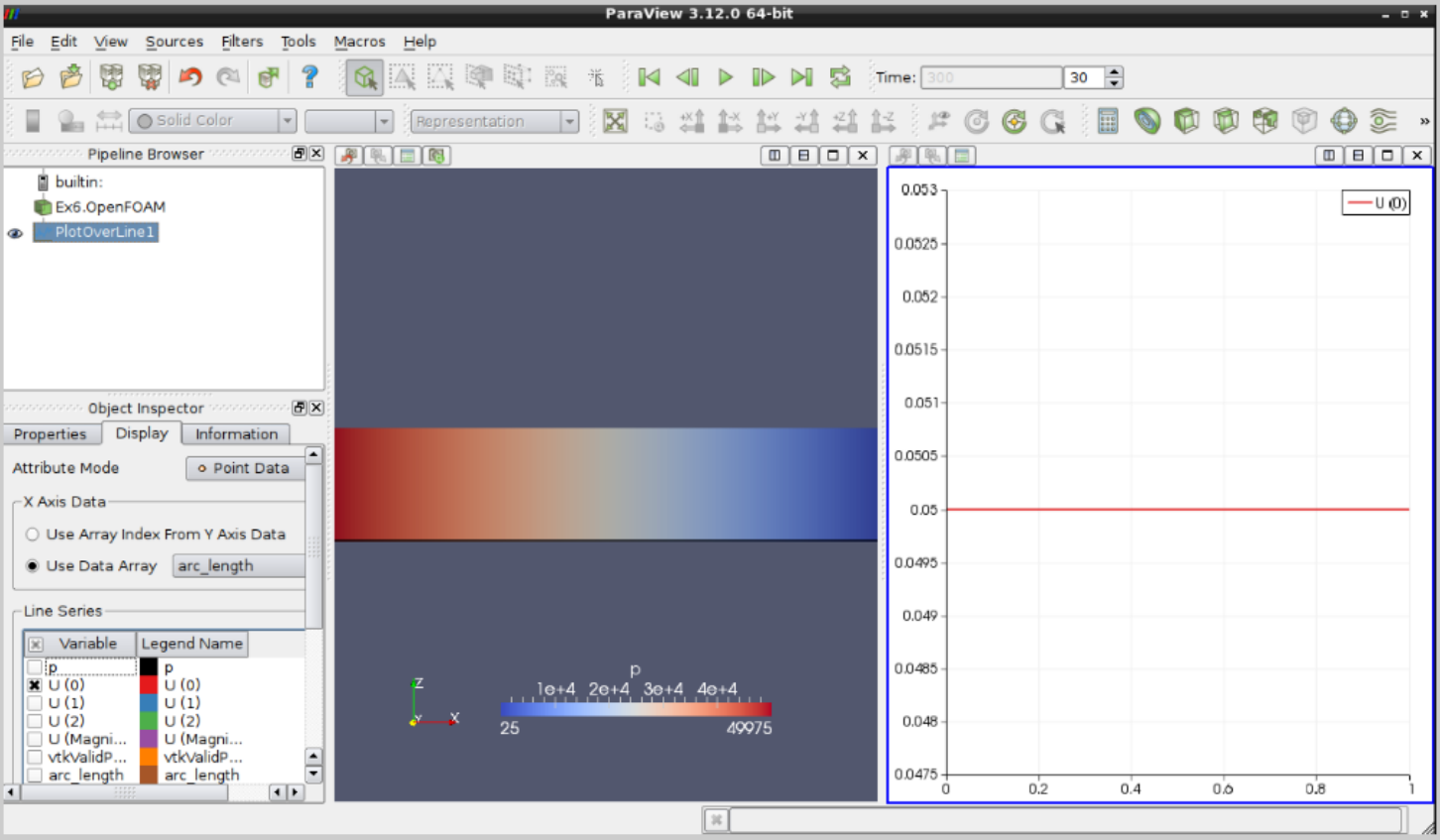
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
```

```
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     tubeFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         1000;

deltaT          1;

writeControl    runTime;

writeInterval   100;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

// ************************************************************************* //
```

Postprocess the case

Use *paraview* to postprocess the case (there is nothing great to see):

```
$ paraFoam
```



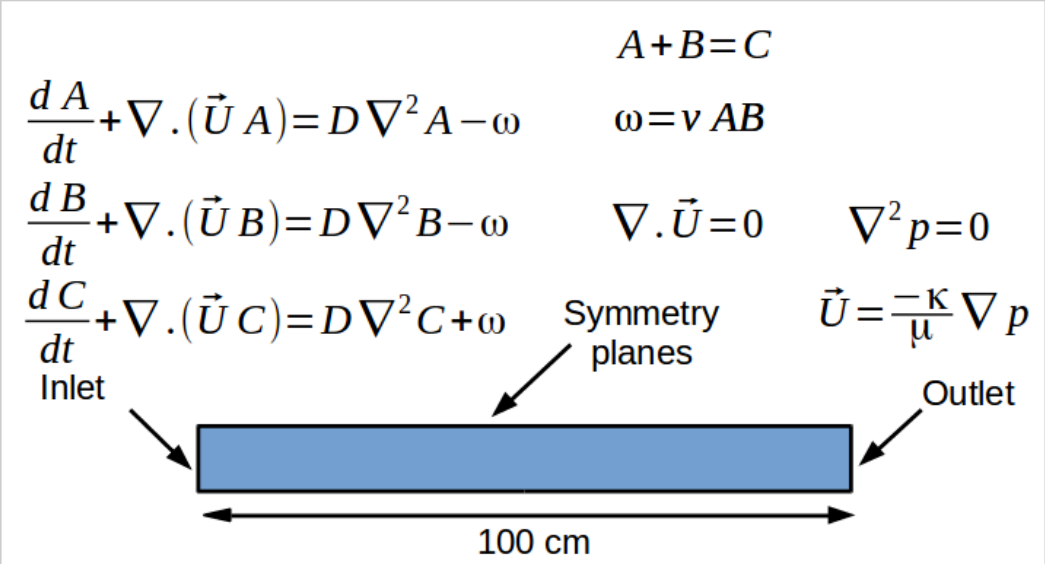
Ex. 7 - Tubular reactor

Objectives

- Create our own solver with more complex equations
- Use common differential operators
- Evolving a case

Case setup

1D tubular reactor with chemical reaction, in transient state.



Evolving an existing solver

Reach *'run'* directory:

```
$ run
```

Move to solver directory:

```
$ cd solvers
```

Copy the existing solver:

```
$ cp -r tubeFoam tubeChemFoam
```

Move to and clean the directory:

```
$ cd tubeChemFoam
```

```
$ wclean
```

Rename the file:

```
$ mv tubeFoam.C tubeChemFoam.C
```

Modify compilation file:

```
$ gedit Make/files
```

```
tubeChemFoam.C

EXE = $(FOAM_USER_APPBIN)/tubeChemFoam
```

Adding new chemical species field to the solver

Modify *'createFields.H'* to create the species fields and add mass diffusivity coefficient and chemical reaction rate:

```
$ gedit createFields.H
```

```
    Info<< "Reading field A, B and C\n" << endl;

    volScalarField A
    (
        IOobject
        (
            "A",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );
    volScalarField B
    (
        IOobject
        (
            "B",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );
    volScalarField C
    (
        IOobject
        (
            "C",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );

    Info<< "Reading field p\n" << endl;

    volScalarField p
    (
        IOobject
        (
            "p",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );

    Info<< "Reading field U\n" << endl;
    volVectorField U
    (
        IOobject
        (
            "U",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
```

```
        mesh
    );

    #include "createPhi.H"

    Info<< "Reading transportProperties\n" << endl;

    IOdictionary transportProperties
    (
        IOobject
        (
            "transportProperties",
            runTime.constant(),
            mesh,
            IOobject::MUST_READ_IF_MODIFIED,
            IOobject::NO_WRITE
        )
    );

    Info<< "Reading dynamic viscosity mu\n" << endl;

    dimensionedScalar mu("mu", transportProperties);

    Info<< "Reading permeability kappa\n" << endl;

    dimensionedScalar kappa("kappa", transportProperties);

    Info<< "Reading diffusivity D\n" << endl;

    dimensionedScalar D("D", transportProperties);

    Info<< "Reading chemical reaction rate v\n" << endl;

    dimensionedScalar v("v", transportProperties);
```

Adding new equations

Modify *'tubeChemFoam.C'* to add the equations governing the chemical species balance:

```
$ gedit tubeChemFoam.C
```

Two points are of note here:

- A and B equations are coupled
- The field named *'phi'* is the interpolation of the velocity field on the mesh. It is computed in order to later calculate the divergence operator. It is declared using *' #include "createPhi.H" '* in *'createFields.H'*

Once you are done, clean the directory and compile:

```
$ wclean; wmake
```

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Application
    tubeChemFoam

\*---------------------------------------------------------------------------*/

#include "fvCFD.H"
#include "fvOptions.H"
#include "simpleControl.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
    #include "setRootCase.H"

    #include "createTime.H"
    #include "createMesh.H"
```

```
    simpleControl simple(mesh);

    #include "createFields.H"
    #include "createFvOptions.H"

    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

    Info<< "\nCalculating temperature distribution\n" << endl;

    while (simple.loop())
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;

        while (simple.correctNonOrthogonal())
        {
            solve
            (
                fvm::laplacian(p) // implicitly equal to 0
            );

            U = -kappa / mu * fvc::grad(p);
            phi = linearInterpolate(U) & mesh.Sf();

            solve
            (
                fvm::ddt(A) + fvm::div(phi,A) == fvm::laplacian(D, A) - v*A*B
            );
            solve
            (
                fvm::ddt(B) + fvm::div(phi,B) == fvm::laplacian(D, B) - v*A*B
            );
            solve
            (
                fvm::ddt(C) + fvm::div(phi,C) == fvm::laplacian(D, C) + v*A*B
            );
        }

        runTime.write();

        Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
            << "  ClockTime = " << runTime.elapsedClockTime() << " s"
            << nl << endl;
    }

    Info<< "End\n" << endl;

    return 0;
}

// ************************************************************************* //
```

Settings case folder

(This fastidious steps can be skipped using '*Ex7_Root*' folder. Just type: `$ cp -r Ex7_Root Ex7` from your *'run'* folder)

Now that the solver has been compiled, we are going to evolve Ex6 case folder:

```
$ run
```

```
$ cp -r Ex6 Ex7
```

Go to the case directory, clean it:

```
$ cd Ex7
```

```
$ foamListTimes -rm
```

The mesh does not change, hence you can directly build it:

```
$ blockMesh
```

Create new files for the three chemical species:

```
$ cp 0/p 0/A
```

```
$ cp 0/p 0/B
```

```
$ cp 0/p 0/C
```

```
$ gedit 0/A 0/B 0/C
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
```

```
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      A;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 -3 0 0 1 0 0];

internalField   uniform 0;

boundaryField
{

    inlet
    {
        type            fixedValue;
        value           uniform 2;
    }

    symmetryPlanes
    {
        type            symmetry;
    }

    outlet
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      B;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 -3 0 0 1 0 0];

internalField   uniform 0;

boundaryField
{

    inlet
    {
        type            fixedValue;
        value           uniform 1;
    }

    symmetryPlanes
    {
        type            symmetry;
    }

    outlet
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      C;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 -3 0 0 1 0 0];

internalField   uniform 0;
```

```
boundaryField
{

    inlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    symmetryPlanes
    {
        type            symmetry;
    }

    outlet
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //
```

Add the newly declared physical properties to *'transportProperties'* dictionnary:

```
$ gedit constant/transportProperties
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

mu                      mu [1 -1 -1 0 0 0 0] 1e-3;
kappa                   kappa [0 2 0 0 0 0 0] 1e-9;
v                       v [0 3 -1 0 -1 0 0] 0.2;
D                       D [0 2 -1 0 0 0 0] 1e-6;

// ************************************************************************* //
```

Set numerical schemes for the divergence operator:

```
$ gedit system/fvSchemes
```

Add the solvers parameters for the chemical species fields:

```
$ gedit system/fvSolution
```

Due to the use of the divergence, we need to use asymmetric solvers for the species fields.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}
```

```
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
    }

    "(A|B|C)"
    {
        solver          PBiCGStab;
        preconditioner  DILU;
        tolerance       1e-06;
        relTol          0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;
}

// ************************************************************************* //
```

Modify control parameters and run the case:

```
$ gedit system/controlDict
```

```
$ tubeChemFoam
```

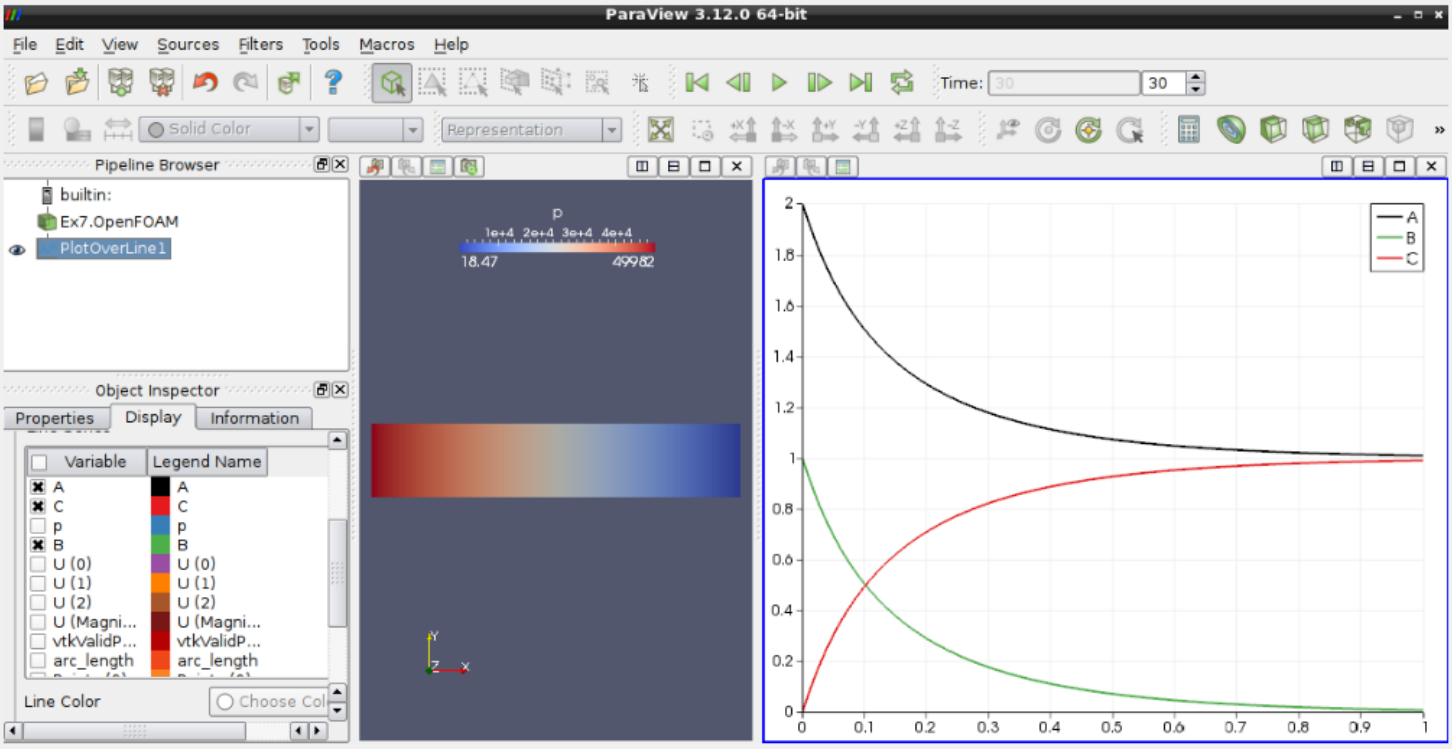
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     tubeChemFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         30;

deltaT          0.1;

writeControl    runTime;

writeInterval   1;

purgeWrite      0;
```

```
writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

// ************************************************************************* //
```

Postprocess the case

You can a look at chemical species variations in time and space:

```
$ paraFoam
```



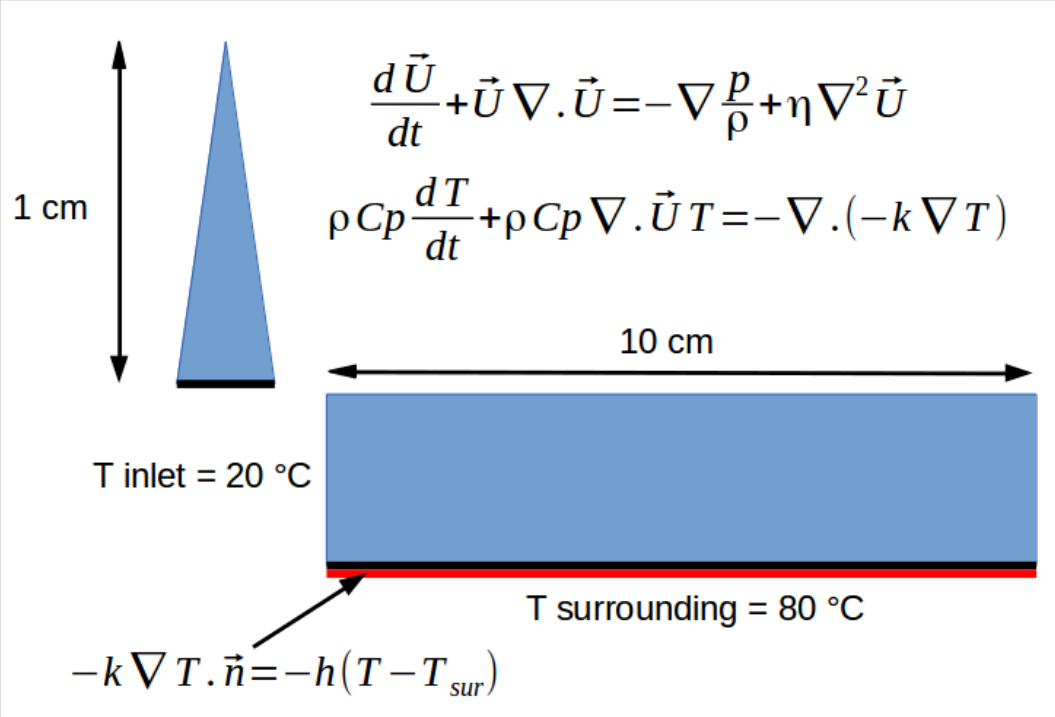
Ex. 8 - Heated pipe

Objectives

- Create a varying boundary condition

Case setup

Laminar incompressible flow in a 2D axisymmetrical pipe with heat transfer, in steady state.



Evolving an existing solver

This fastidious steps can be skipped using *'simpleThermFoam_Root'* folder. Just type:

```
$ run
$ cd solvers
$ cp -r simpleThermFoam_Root simpleThermFoam
$ cd simpleThermFoam
$ wclean; wmake
```

Otherwise ...
Reach *'run'* directory:

```
$ run
```

Move to solver directory:

```
$ cd solvers
```

Copy and rename an existing solver:

```
$ cp -r $FOAM_APP/solvers/incompressible/simpleFoam .
$ mv simpleFoam simpleThermFoam
```

Move to and clean the directory:

```
$ cd simpleThermFoam
$ wclean
$ rm -r porousSimpleFoam/ SRFSimpleFoam/ overSimpleFoam/
```

Rename the file:

```
$ mv simpleFoam.C simpleThermFoam.C
```

Modify compilation file:

```
$ gedit Make/files
```

```
simpleThermFoam.C

EXE = $(FOAM_APPBIN)/simpleThermFoam
```

Add temperature field and related physical properties to *'createFields.H'*:

```
$ gedit createFields.H
```

```
Info<< "Reading field T\n" << endl;
volScalarField T
(
    IOobject
    (
        "T",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

Info<< "Reading field p\n" << endl;
volScalarField p
(
    IOobject
    (
        "p",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
```

```
    mesh
);

Info<< "Reading field U\n" << endl;
volVectorField U
(
    IOobject
    (
        "U",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

#include "createPhi.H"


label pRefCell = 0;
scalar pRefValue = 0.0;
setRefCell(p, simple.dict(), pRefCell, pRefValue);
mesh.setFluxRequired(p.name());


singlePhaseTransportModel laminarTransport(U, phi);

autoPtr<incompressible::turbulenceModel> turbulence
(
    incompressible::turbulenceModel::New(U, phi, laminarTransport)
);

#include "createMRF.H"

IOdictionary transportProperties
(
        IOobject
        (
            "transportProperties",
            runTime.constant(),
            mesh,
            IOobject::MUST_READ_IF_MODIFIED,
            IOobject::NO_WRITE
        )
);


Info<< "Reading density rho\n" << endl;

dimensionedScalar rho("rho", transportProperties);

Info<< "Reading specific heat Cp\n" << endl;

dimensionedScalar Cp("Cp", transportProperties);

Info<< "Reading thermal conductivity k\n" << endl;

dimensionedScalar k("k", transportProperties);
```

Then, add temperature equation to *'simpleThermFoam.C'*:

```
$ gedit simpleThermFoam.C
```

```
/*-------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Application
    simpleThermFoam

\*-------------------------------------------------------------------------*/

#include "fvCFD.H"
#include "singlePhaseTransportModel.H"
#include "turbulentTransportModel.H"
#include "simpleControl.H"
```

```
#include "fvOptions.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
    #include "postProcess.H"

    #include "setRootCase.H"
    #include "createTime.H"
    #include "createMesh.H"
    #include "createControl.H"
    #include "createFields.H"
    #include "createFvOptions.H"
    #include "initContinuityErrs.H"

    turbulence->validate();

    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

    Info<< "\nStarting time loop\n" << endl;

    while (simple.loop())
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;

        // --- Pressure-velocity SIMPLE corrector
        {
            #include "UEqn.H"
            #include "pEqn.H"
        }

        laminarTransport.correct();
        turbulence->correct();

        // --- Temperature field
        solve
        (
                fvm::ddt(rho * Cp, T)
              + rho * Cp * fvm::div(phi, T)
              ==
              fvm::laplacian(k, T)
        );

        runTime.write();

        Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
            << "  ClockTime = " << runTime.elapsedClockTime() << " s"
            << nl << endl;
    }

    Info<< "End\n" << endl;

    return 0;
}


// ************************************************************************* //
```

Finnaly, compile the solver:

```
$ wclean; wmake
```

Creating a new boundary condition

As for solvers, we are going to create our own folder to store boundary conditions. Then, we will modify an existing boundary conditions to create our own.
Reach *'run'* directory:

```
$ run
```

Create a new directory and enter it:

```
$ mkdir boundaryConditions
$ cd boundaryConditions
```

Copy an existing boundary condition folder and enter it:

```
$ cp -r $FOAM_SRC/finiteVolume/fields/fvPatchFields/derived/totalTemperature
convectiveHeatFlux
$ cd convectiveHeatFlux
```

Renane the boundary condition files:

```
$ mv totalTemperatureFvPatchScalarField.C convectiveHeatFluxFvPatchScalarField.C
$ mv totalTemperatureFvPatchScalarField.H convectiveHeatFluxFvPatchScalarField.H
```

As there is no compilation files, nor folder, we have to create them:

```
$ mkdir Make
```

```
$ gedit Make/files Make/options
```

Fill the files and save them.

```
convectiveHeatFluxFvPatchScalarField.C

LIB = $(FOAM_USER_LIBBIN)/lconvectiveHeatFlux
```

```
EXE_INC = -I$(LIB_SRC)/finiteVolume/lnInclude

LIB_LIBS = -lfiniteVolume
```

Modify the boundary condition header file

This file contains the variables and functions declarations. You can find a lot of occurence of *'totalTemperature'* that have to be changed in *'convectiveHeatFlux'*:

```
$ gedit convectiveHeatFluxFvPatchScalarField.H
```

A nice way to replace everything at once is using the *'find & replace'* function of the editor. You can access *gedit* one by pressing *'ctrl + h'*
In addition, *'scalar affection'* has to be turned into *'gradient affectation'* and some variable (such as *'Tname_'*) have to be declared.

```
/*-------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Class
    Foam::convectiveHeatFluxFvPatchScalarField

\*-------------------------------------------------------------------------*/

#ifndef convectiveHeatFluxFvPatchScalarField_H
#define convectiveHeatFluxFvPatchScalarField_H

#include "fixedGradientFvPatchFields.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{

/*-------------------------------------------------------------------------*\
             Class convectiveHeatFluxFvPatchScalarField Declaration
\*-------------------------------------------------------------------------*/

class convectiveHeatFluxFvPatchScalarField
:
```

```
    public fixedGradientFvPatchScalarField
{
    // Private data

        //- Name of the velocity field
        word TName_;


public:

    //- Runtime type information
    TypeName("convectiveHeatFlux");


    // Constructors

        //- Construct from patch and internal field
        convectiveHeatFluxFvPatchScalarField
        (
            const fvPatch&,
            const DimensionedField<scalar, volMesh>&
        );

        //- Construct from patch, internal field and dictionary
        convectiveHeatFluxFvPatchScalarField
        (
            const fvPatch&,
            const DimensionedField<scalar, volMesh>&,
            const dictionary&
        );

        //- Construct by mapping given convectiveHeatFluxFvPatchScalarField
        //  onto a new patch
        convectiveHeatFluxFvPatchScalarField
        (
            const convectiveHeatFluxFvPatchScalarField&,
            const fvPatch&,
            const DimensionedField<scalar, volMesh>&,
            const fvPatchFieldMapper&
        );

        //- Construct as copy
        convectiveHeatFluxFvPatchScalarField
        (
            const convectiveHeatFluxFvPatchScalarField&
        );

        //- Construct and return a clone
        virtual tmp<fvPatchScalarField> clone() const
        {
            return tmp<fvPatchScalarField>
            (
                new convectiveHeatFluxFvPatchScalarField(*this)
            );
        }

        //- Construct as copy setting internal field reference
        convectiveHeatFluxFvPatchScalarField
        (
            const convectiveHeatFluxFvPatchScalarField&,
            const DimensionedField<scalar, volMesh>&
        );

        //- Construct and return a clone setting internal field reference
        virtual tmp<fvPatchScalarField> clone
        (
            const DimensionedField<scalar, volMesh>& iF
        ) const
        {
            return tmp<fvPatchScalarField>
            (
                new convectiveHeatFluxFvPatchScalarField(*this, iF)
            );
        }

        // Evaluation functions

            //- Update the coefficients associated with the patch field
            virtual void updateCoeffs();


        //- Write
        virtual void write(Ostream&) const;
};


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

} // End namespace Foam

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#endif

// ************************************************************************* //
```

Modify the boundary condition code file

Just like for the header file, there are a lot of occurence of *'totalTemperature'* that have to be changed in *'convectiveHeatFlux'*:

```
$ gedit convectiveHeatFluxFvPatchScalarField.C
```

Again use the *'find & replace'* function of the editor.
Several extra actions need to be carried out on this file. Constructor functions have to be modified to change the *'scalar affection'* to a *'gradient affectation'*. The heat flux equation has also to be implemented. Those steps are especially long and require a lot of rigor. This is the *'no fun'* part of this tutorial.

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

\*---------------------------------------------------------------------------*/

#include "convectiveHeatFluxFvPatchScalarField.H"
#include "addToRunTimeSelectionTable.H"
#include "fvPatchFieldMapper.H"
#include "volFields.H"
#include "surfaceFields.H"

// * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * * //

Foam::convectiveHeatFluxFvPatchScalarField::convectiveHeatFluxFvPatchScalarField
(
    const fvPatch& p,
    const DimensionedField<scalar, volMesh>& iF
)
:
    fixedGradientFvPatchScalarField(p, iF),
    TName_("T")
{}


Foam::convectiveHeatFluxFvPatchScalarField::convectiveHeatFluxFvPatchScalarField
(
    const convectiveHeatFluxFvPatchScalarField& ptf,
    const fvPatch& p,
    const DimensionedField<scalar, volMesh>& iF,
    const fvPatchFieldMapper& mapper
)
:
    fixedGradientFvPatchScalarField(ptf, p, iF, mapper),
    TName_(ptf.TName_)
{}


Foam::convectiveHeatFluxFvPatchScalarField::convectiveHeatFluxFvPatchScalarField
(
    const fvPatch& p,
    const DimensionedField<scalar, volMesh>& iF,
    const dictionary& dict
)
:
    fixedGradientFvPatchScalarField(p, iF),
    TName_(dict.lookupOrDefault<word>("T", "T"))
{
        fvPatchField<scalar>::operator= (patchInternalField());
        gradient() = 0.0;
}


Foam::convectiveHeatFluxFvPatchScalarField::convectiveHeatFluxFvPatchScalarField
(
    const convectiveHeatFluxFvPatchScalarField& tppsf
)
:
    fixedGradientFvPatchScalarField(tppsf),
    TName_(tppsf.TName_)
{}


Foam::convectiveHeatFluxFvPatchScalarField::convectiveHeatFluxFvPatchScalarField
(
```

```
    const convectiveHeatFluxFvPatchScalarField& tppsf,
    const DimensionedField<scalar, volMesh>& iF
)
:
    fixedGradientFvPatchScalarField(tppsf, iF),
    TName_(tppsf.TName_)
{}


// * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * //


void Foam::convectiveHeatFluxFvPatchScalarField::updateCoeffs()
{
    if (updated())
    {
        return;
    }

    const fvPatchField<scalar>& T =
        patch().lookupPatchField<volScalarField, scalar>(TName_);

    const dictionary& TP = db().lookupObject<IOdictionary>("transportProperties");

        dimensionedScalar h (TP.lookup("h"));
        dimensionedScalar k (TP.lookup("k"));
        dimensionedScalar Tsur (TP.lookup("Tsur"));

    gradient() = h.value() / k.value() * (Tsur.value() - T);

    fixedGradientFvPatchScalarField::updateCoeffs();
}


void Foam::convectiveHeatFluxFvPatchScalarField::write(Ostream& os) const
{
    fvPatchScalarField::write(os);
    writeEntryIfDifferent<word>(os, "T", "T", TName_);
    writeEntry("value", os);
}


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{
    makePatchTypeField
    (
        fvPatchScalarField,
        convectiveHeatFluxFvPatchScalarField
    );
}

// ************************************************************************* //
```

Once done, compile the boundary condition:

```
$ wclean; wmake
```

Set the thermal heat transfer case

(You are not going to learn anything new here, so if you would like to skip this step, just type: `$ cp -r Ex8_Root Ex8` from your *'run'* folder)

Now that the solver has been compiled, we are going to evolve Ex6 case folder:

```
$ run
```

```
$ cp -r Ex5 Ex8
```

Go to the case directory, clean it:

```
$ cd Ex8
```

```
$ foamListTimes -rm
```

The mesh does not change, hence you can directly build it:

```
$ blockMesh
```

Create new file for the temperature field:

```
$ cp 0/p 0/T
```

```
$ gedit 0/T
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
```

```
| \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 293;

boundaryField
{
    outlet
    {
        type            zeroGradient;
    }

    inlet
    {
        type            fixedValue;
        value           uniform 293;
    }

    wall
    {
        type            convectiveHeatFlux;
    }

    "(front|back)"
    {
        type            wedge;
    }
    axis
    {
        type            empty;
    }
}

// ************************************************************************* //
```

Add the new physical properties to *'transportProperties'* dictionnary:

```
$ gedit constant/transportProperties
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
| \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              [0 2 -1 0 0 0 0] 1e-5;

rho             rho [1 -3 0 0 0 0 0] 1000;
Cp              Cp [0 2 -2 -1  0 0 0] 4183;
k               k [1 1 -3 -1 0 0 0] 0.6;
h               h [1 0 -3 -1 0 0 0] 100;
Tsur            Tsur [0 0 0 1 0 0 0] 353;

// ************************************************************************* //
```

Set numerical schemes for temperature field:

```
$ gedit system/fvSchemes
```

Add the solvers parameters for the temperature field:

```
$ gedit system/fvSolution
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
| \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
```

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         steadyState;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default                             none;
    div(phi,U)                          bounded Gauss linearUpwind grad(U);
    div(phi,k)                          bounded Gauss limitedLinear 1;
    div(phi,epsilon)                    bounded Gauss limitedLinear 1;
    div(phi,omega)                      bounded Gauss limitedLinear 1;
    div(phi,v2)                         bounded Gauss limitedLinear 1;
    div((nuEff*dev2(T(grad(U)))))       Gauss linear;
    div(nonlinearStress)                Gauss linear;
    div(phi,T)                          bounded Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

wallDist
{
    method meshWave;
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0.1;
        smoother        GaussSeidel;
    }

    T
    {
        solver          PBiCG;
        tolerance       1e-06;
        preconditioner  DILU;
        relTol          0.1;
    }

    "(U|k|epsilon|omega|f|v2)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
```

```
        tolerance       1e-05;
        relTol          0.1;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;
    consistent      yes;

    residualControl
    {
        p               1e-2;
        U               1e-3;
        T               1e-3;
        "(k|epsilon|omega|f|v2)" 1e-3;
    }
}

relaxationFactors
{
    equations
    {
        U               0.9; // 0.9 is more stable but 0.95 more convergent
        ".*"            0.9; // 0.9 is more stable but 0.95 more convergent
    }
}

// ************************************************************************* //
```

Modify control parameters to specify that we are using an homemade boundary condition and run the case:

```
$ gedit system/controlDict
```

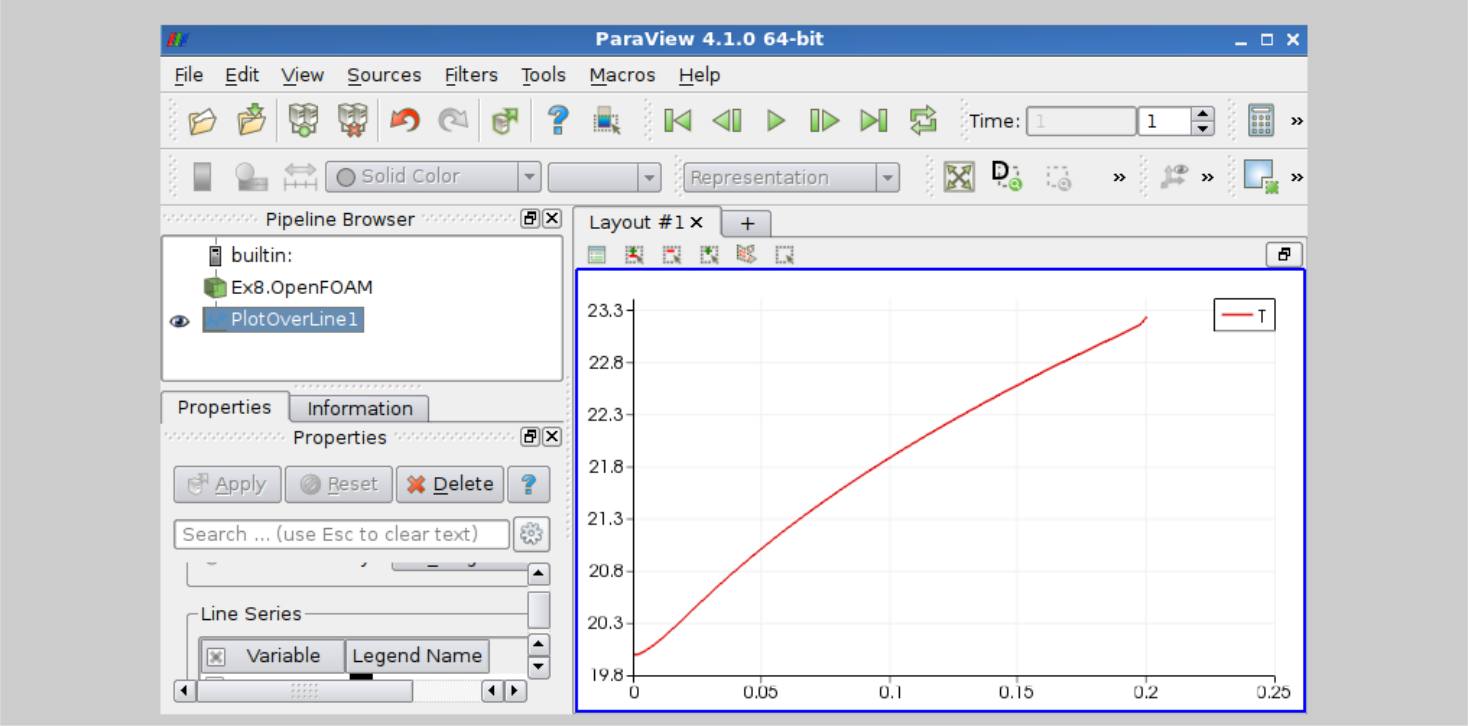
```
$ simpleThermFoam
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     simpleThermFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         100;

deltaT          1;

writeControl    timeStep;

writeInterval   10;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

libs ( "lconvectiveHeatFlux.so" );

// ************************************************************************* //
```

Postprocess the case

Have a look at both thermal and momentum boundary layers:

```
$ paraFoam
```


ParaView 4.1.0 64-bit
File
Edit
View
Sources
Filters
Tools
Macros
Help
Time:
Solid Color
Representation
Pipeline Browser
builtin:
Ex8.OpenFOAM
PlotOverLine1
Properties
Information
Properties
Apply
Reset
Delete
Search ... (use Esc to clear text)
Line Series
Variable
Legend Name
Layout #1
T
23.3
22.8
22.3
21.8
21.3
20.8
20.3
19.8
0
0.05
0.1
0.15
0.2
0.25

Ex. 9 - Cake backing

Objectives

- Set a field that depend on another field
- Set spatial non uniform physical field
- Use an existing case as initial condition

T: temperature, C: solid chocolate, LC: liquid chocolate, v: fusion/solidication rate

Case setup

We are going to simulate the backing of a cake:

- Heating up with chocolate melting (20 min, 180°C)
- Cooling down with chocolate solidification (1 hour)

$$\rho Cp \frac{dT}{dt} = -\nabla.(-k\nabla T) + Q \qquad Q = \left(\frac{dC}{dt} - \frac{dLC}{dt}\right)\Delta H$$

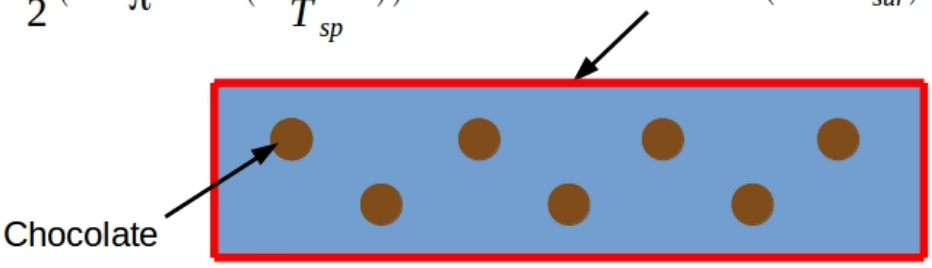
$$\frac{dC}{dt} = -v f(T) C + v(1 - f(T)) LC$$

$$\frac{dLC}{dt} = v f(T) C - v(1 - f(T)) LC$$

$$f(T) = \frac{1}{2}\left(1 + \frac{2}{\pi} atan\left(\frac{T - T_0}{T_{sp}}\right)\right)$$

T surrounding = 180 °C

$$-k\nabla T.\vec{n} = -h(T - T_{sur})$$



Evolving an existing solver

These long and uninstructive steps can be skipped using *'cakeFoam_Root'* folder. Just type:

```
$ run
$ cd solvers
$ cp -r cakeFoam_Root cakeFoam
$ cd cakeFoam
$ wclean; wmake
```

Otherwise ...
Reach *'run'* directory:

```
$ run
```

Move to solver directory:

```
$ cd solvers
```

Copy and rename an existing *'tubeFoam'* solver:

```
$ cp -r tubeFoam cakeFoam
```

Move to and clean the directory:

```
$ cd cakeFoam
```

```
$ wclean
```

Rename the solver file:

```
$ mv tubeFoam.C cakeFoam.C
```

Modify compilation file:

```
$ gedit Make/files
```

```
cakeFoam.C

EXE = $(FOAM_USER_APPBIN)/cakeFoam
```

Add temperature and chocolate fields and related physical properties to *'createFields.H'*:

```
$ gedit createFields.H
```

```
    Info<< "Reading field T\n" << endl;

    volScalarField T
    (
        IOobject
        (
            "T",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );

    volScalarField C
    (
        IOobject
        (
            "C",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );

    volScalarField LC
    (
        IOobject
        (
            "LC",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );

    Info<< "Reading transportProperties\n" << endl;

    IOdictionary transportProperties
    (
        IOobject
        (
            "transportProperties",
            runTime.constant(),
            mesh,
```

```
            IOobject::MUST_READ_IF_MODIFIED,
            IOobject::NO_WRITE
        )
    );

    Info<< "Reading density rho\n" << endl;

    dimensionedScalar rho("rho", transportProperties);

    Info<< "Reading thermal capacity Cp\n" << endl;

    dimensionedScalar Cp("Cp", transportProperties);

    Info<< "Reading thermal conductivity k\n" << endl;

    dimensionedScalar k("k", transportProperties);

    Info<< "Reading frequency factor v\n" << endl;

    dimensionedScalar v("v", transportProperties);

    Info<< "Chocolate reaction enthalpy DH\n" << endl;

    dimensionedScalar DH("DH", transportProperties);

    Info<< "Chocolate melting temperature T0\n" << endl;

    dimensionedScalar T0("T0", transportProperties);

    Info<< "Tweaking parameter Tsp\n" << endl;

    dimensionedScalar Tsp("Tsp", transportProperties);
```

Then, add temperature and chocolate balances to *'cakeFoam.C'*:

```
$ gedit cakeFoam.C
```

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Application
    cakeFoam

\*---------------------------------------------------------------------------*/

#include "fvCFD.H"
#include "fvOptions.H"
#include "simpleControl.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
    #include "setRootCase.H"

    #include "createTime.H"
    #include "createMesh.H"

    simpleControl simple(mesh);

    #include "createFields.H"
    #include "createFvOptions.H"

    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

    Info<< "\nCalculating temperature distribution\n" << endl;

    const double pi = 3.14159265358979323846 3;

    while (simple.loop())
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;

            solve
            (
                    fvm::ddt(rho * Cp, T)
```

```
                    ==
                    fvm::laplacian(k, T)
                    - v * (1 + 2/pi * Foam::atan((T-T0)/Tsp)) / 2 * C * DH
                    + v * (1 - 2/pi * Foam::atan((T-T0)/Tsp)) / 2 * LC * DH
            );

            solve
            (
                    fvm::ddt(C)
                    ==
                    - v * (1 + 2/pi * Foam::atan((T-T0)/Tsp)) / 2 * C
                    + v * (1 - 2/pi * Foam::atan((T-T0)/Tsp)) / 2 * LC
            );

            solve
            (
                    fvm::ddt(LC)
                    ==
                    v * (1 + 2/pi * Foam::atan((T-T0)/Tsp)) / 2 * C
                    - v * (1 - 2/pi * Foam::atan((T-T0)/Tsp)) / 2 * LC
            );

        //- Writing time step on the disk
        runTime.write();

        Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
            << "  ClockTime = " << runTime.elapsedClockTime() << " s"
            << nl << endl;
    }

    Info<< "End\n" << endl;

    return 0;
}

// ************************************************************************* //
```

Finnaly, compile the solver:

```
$ wclean; wmake
```

Seting up the case

(This part covers the first half of the *'setting the case'* procedure. There will be nothing new, so if you would like to skip this step, just type: `$ cp -r Ex9_Root Ex9` from your *'run'* folder)

Now that the solver has been compiled, we are going to evolve Ex6 case folder:

```
$ run
$ cp -r Ex4 Ex9
```

Go to the case directory, clean it:

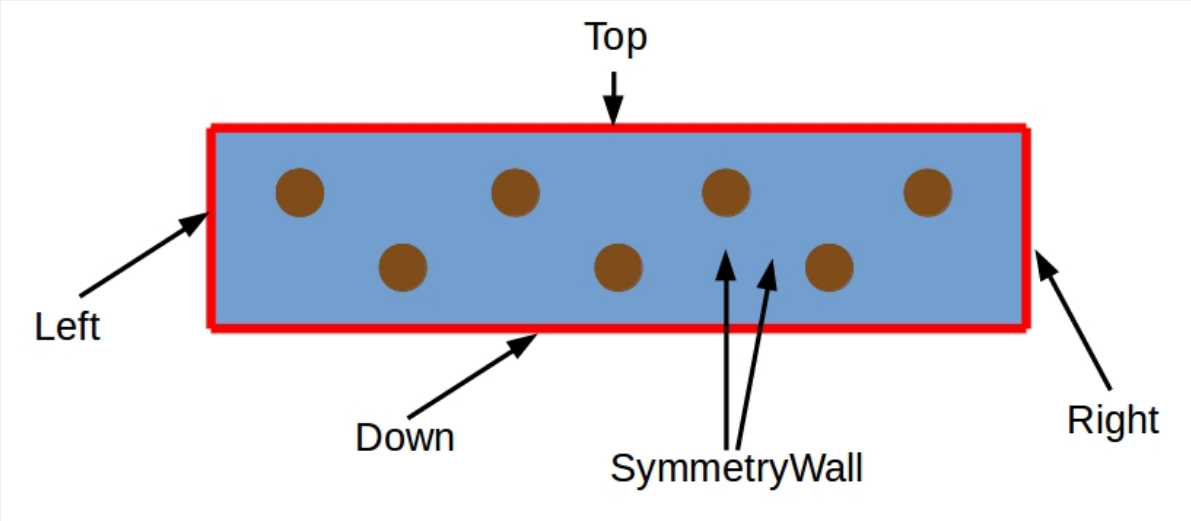
```
$ cd Ex9
$ foamListTimes -rm
$ rm 0/U
$ rm -r constant/turbulenceProperties
```

Modify the mesh as follows and compile it:

```
$ gedit system/blockMeshDict
```

Then:

```
$ blockMesh
```




```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

scale 0.01;

vertices
(
    (0 0 0)
    (15 0 0)
    (15 1 0)
    (0 1 0)
    (0 0 7)
    (15 0 7)
    (15 1 7)
    (0 1 7)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (50 1 25) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
    left
    {
        type patch;
        faces
        (
            (0 3 7 4)
        );
    }
    right
    {
        type patch;
        faces
        (
            (1 2 6 5)
        );
    }
    symmetryWall
    {
        type empty;
        faces
        (
            (0 1 5 4)
            (3 2 6 7)
        );
    }
    down
    {
        type patch;
        faces
        (
            (0 1 2 3)
        );
```

```
    }
    top
    {
        type patch;
        faces
        (
            (4 5 6 7)
        );
    }

);

// ************************************************************************* //
```

Set the initial / boundary condition for the three fields:

```
$ cp 0/p 0/C

$ cp 0/p 0/LC

$ mv 0/p 0/T

$ gedit 0/C 0/LC 0/T
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      C;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -3 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    left
    {
        type            zeroGradient;
    }
    right
    {
        type            zeroGradient;
    }
    symmetryWall
    {
        type            empty;
    }
    down
    {
        type            zeroGradient;
    }
    top
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      LC;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -3 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    left
```

```
    {
        type            zeroGradient;
    }
    right
    {
        type            zeroGradient;
    }
    symmetryWall
    {
        type            empty;
    }
    down
    {
        type            zeroGradient;
    }
    top
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 293;

boundaryField
{
    left
    {
        type            convectiveHeatFlux;
    }
    right
    {
        type            convectiveHeatFlux;
    }
    down
    {
        type            convectiveHeatFlux;
    }
    top
    {
        type            convectiveHeatFlux;
    }
    symmetryWall
    {
        type            empty;
    }
}

// ************************************************************************* //
```

Add the new physical properties to *'transportProperties'* dictionnary:

```
$ gedit constant/transportProperties
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

DH              DH      [ 0 2 -2 0 0 0 0 ]  200e3;
v               v       [ 0 0 -1 0 0 0 0 ]  0.1;
```

```
rho             rho     [ 1 -3 0 0 0 0 0 ]  250;
Cp              Cp      [ 0 2 -2 -1 0 0 0 ] 2500;
k               k       [ 1 1 -3 -1 0 0 0 ] 0.32;

h               h       [ 1 0 -3 -1 0 0 0 ] 40;
Tsur            Tsur    [ 0 0 0 1 0 0 0 ]   453; // 293 or 453
T0              T0      [ 0 0 0 1 0 0 0 ]   373;
Tsp             Tsp     [ 0 0 0 1 0 0 0 ]   5;

// ************************************************************************* //
```

Set numerical schemes for the three fields:

```
$ gedit system/fvSchemes
```

Add the solvers parameters for the three fields:

```
$ gedit system/fvSolution
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
}

laplacianSchemes
{
    default         none;
    laplacian(k,T)  Gauss linear orthogonal;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         orthogonal;
}

fluxRequired
{
    default         no;
}



// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "(T|C|LC)"
```

```
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-09;
        relTol          0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;
}

// ************************************************************************* //
```

Modify control parameters to specify that the heating phase lasts 20 min (1800 s) and the use of our personal thermal boundary condition:

```
$ gedit system/controlDict
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     cakeFoam;

startFrom       startTime;

startTime       0; // 0 or 1800

stopAt          endTime;

endTime         1800; // 1800 or 5400

deltaT          0.1;

writeControl    timeStep;

writeInterval   300;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

libs ( "lconvectiveHeatFlux.so" );

// ************************************************************************* //
```

Specifying a non uniform initial condition for a field and running the case

We have to specify the position of every chocolate chip. In this case, only three pieces will be set. First, copy an existing dictionary:

```
$ cp $FOAM_TUTORIALS/multiphase/interFoam/RAS/damBreak/damBreak/system/setFieldsDict
system/.
```

Then modify it to set the chocolate chips locations and shapes:

```
$ gedit system/setFieldsDict
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
```

```
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

defaultFieldValues
(
    volScalarFieldValue C 0
);

regions
(
    boxToCell
    {
        box (0.03 0 0.02) (0.04 0.01 0.03);
        fieldValues
        (
            volScalarFieldValue C 800
        );
    }
    boxToCell
    {
        box (0.07 0 0.04) (0.08 0.01 0.05);
        fieldValues
        (
            volScalarFieldValue C 800
        );
    }
    boxToCell
    {
        box (0.11 0 0.02) (0.12 0.01 0.03);
        fieldValues
        (
            volScalarFieldValue C 800
        );
    }
);

// ************************************************************************* //
```

Setting a non uniform initial condition will modify your files in the *'0'* folder. I would advice to save them before applying the modification:

```
$ cp -r 0 _0org
```

Then place the chocolate chips:

```
$ setFields
```

Finnaly, run the case for the heating phase:

```
$ cakeFoam
```

Continuing the run

Now that the heating phase has been computed, we are going to set the external temperature to 293 K and simulate the cooling down phase. To do that modify the eternal temperature in *'transportProperties'* and the control parameters in *'controlDict'*:

```
$ gedit constant/transportProperties
```

```
$ gedit system/controlDict
```

Then, continue the simulation:

```
$ cakeFoam
```

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

DH              DH      [ 0 2 -2 0 0 0 0 ]  200e3;
v               v       [ 0 0 -1 0 0 0 0 ]  0.1;

rho             rho     [ 1 -3 0 0 0 0 0 ]  250;
```

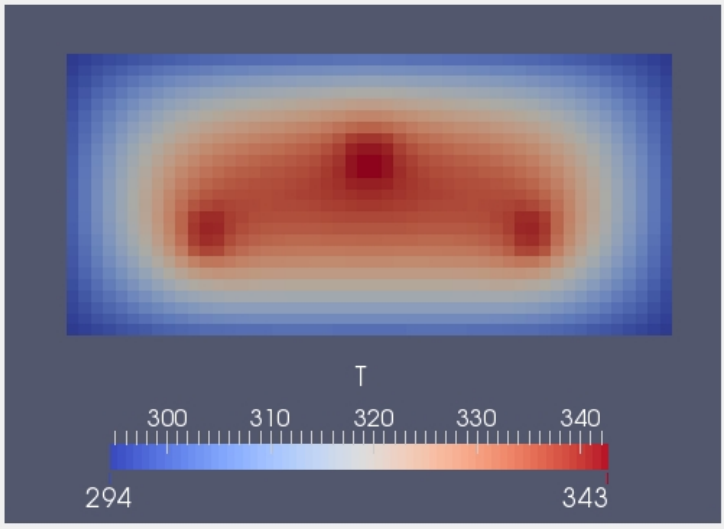
```
Cp              Cp      [ 0 2 -2 -1 0 0 0 ] 2500;
k               k       [ 1 1 -3 -1 0 0 0 ] 0.32;

h               h       [ 1 0 -3 -1 0 0 0 ] 40;
Tsur            Tsur    [ 0 0 0 1 0 0 0 ]   293; // 293 or 453
T0              T0      [ 0 0 0 1 0 0 0 ]   373;
Tsp             Tsp     [ 0 0 0 1 0 0 0 ]   5;

// ************************************************************************* //
```

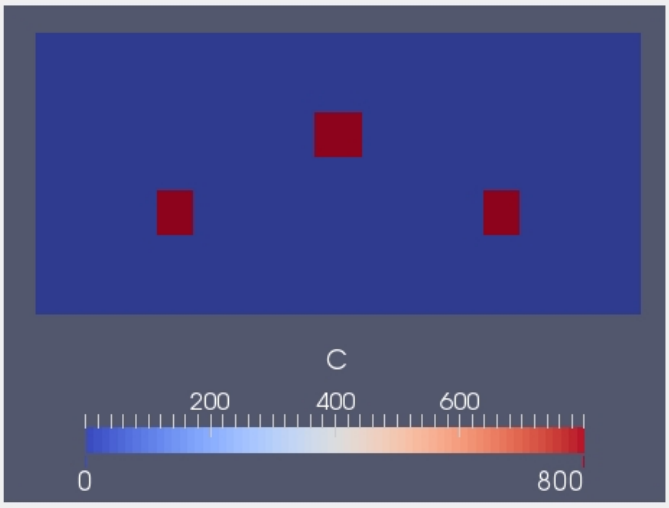
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     cakeFoam;

startFrom       startTime;

startTime       1800; // 0 or 1800

stopAt          endTime;

endTime         5400; // 1800 or 5400

deltaT          0.1;

writeControl    timeStep;

writeInterval   300;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

libs ( "lconvectiveHeatFlux.so" );

// ************************************************************************* //
```

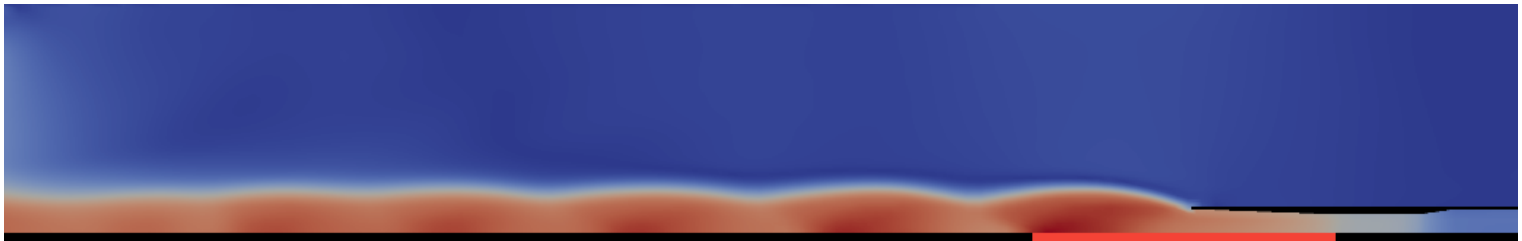
Postprocess the case

Look at the melting chips and the corresponding temperature field:

```
$ paraFoam
```







Ex. 10 - Tank filling

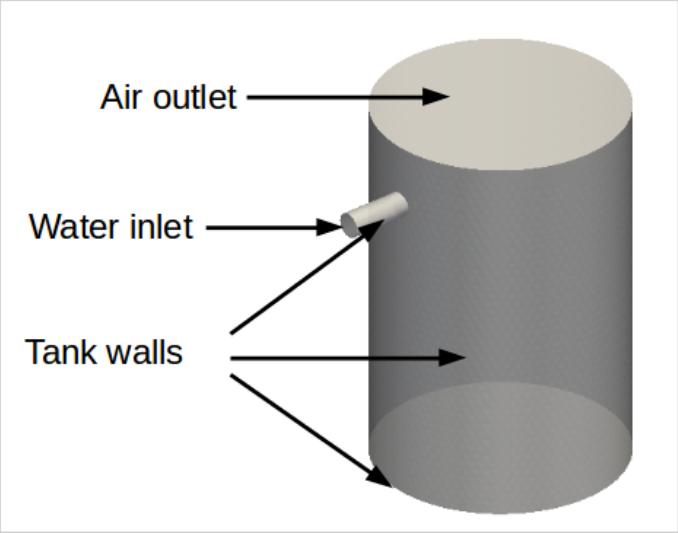
Objectives

This case is really basic. Its purpose is only to discuss topics that were left uncovered.

- Import a mesh created with an external tool
- Run a case in parallel
- Exporting an animation

Case setup

Transient filling of a tank. Initially, it is full of air, then, we are going to fill it using the two phases flow solver *interFoam*.



Copying an existing case

Reach *'run'* directory:

```
$ run
```

Copy *damBreak* case:

```
$ cp -r $FOAM_TUTORIALS/multiphase/interFoam/laminar/damBreak/damBreak/ Ex10
```

Go to the case directory, clean it:

```
$ cd Ex10
$ rm Allrun Allclean
$ mv 0.orig/ 0
$ mv 0/alpha.water 0/alpha.water.orig
```

Importing a mesh

The mesh is called *'Tank.unv'*. It was drawn using *Salome*. It should have already been provided to you, if this is not the case, you can find it here:
http://perso.mines-albi.fr/~vpozzobo/dl/Tank.unv
Move the mesh file to the case directory:

```
$ mv 'somewhere'/Tank.unv Ex10/.
```

or

```
$ wget http://perso.mines-albi.fr/~vpozzobo/dl/Tank.unv
```

Import the mesh:

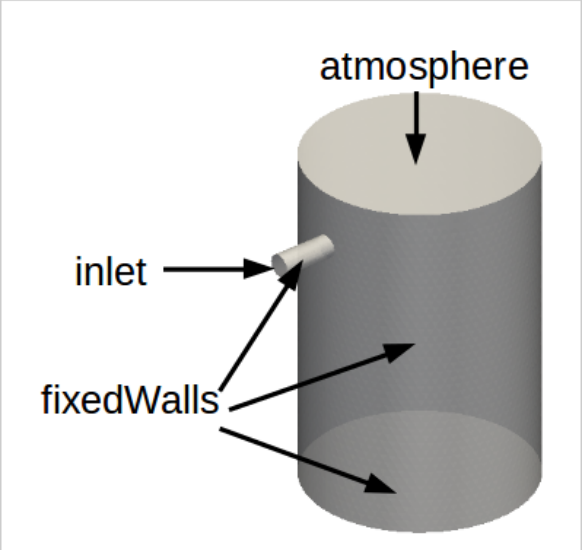
```
$ ideasUnvToFoam Tank.unv
```

There is no need to build the mesh using *blockMesh*, is has already been done when importing it. Yet, it has to be resized:

```
$ transformPoints -scale '(0.001 0.001 0.001)'
```

You can check the mesh using *paraFoam*:

```
$ paraFoam
```



Modifying initial / boundary condition and gravity

Modify velocity, pressure and phase indicator field (alpha.water) intial / boundary conditions according to the case setup:

```
$ gedit 0/U 0/p_rgh 0/alpha.water.orig
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (-0.75 0 0);
    }
    fixedWalls
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
```

```
    atmosphere
    {
        type            pressureInletOutletVelocity;
        value           uniform (0 0 0);
    }
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p_rgh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }
    fixedWalls
    {
        type            fixedFluxPressure;
        value           uniform 0;
    }

    atmosphere
    {
        type            totalPressure;
        p0              uniform 0;
        U               U;
        phi             phi;
        rho             rho;
        psi             none;
        gamma           1;
        value           uniform 0;
    }
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      alpha.water;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    atmosphere
    {
        type            inletOutlet;
        inletValue      uniform 0;
        value           uniform 0;
    }
    fixedWalls
    {
        type            zeroGradient;
    }
    inlet
```

```
    {
        type            inletOutlet;
        inletValue      uniform 1;
        value           uniform 1;
    }
}

// ************************************************************************* //
```

Copy the water phase file:

```
$ cp 0/alpha.water.orig 0/alpha.water
```

Keep the *'.orig'* file. It is used as a backup in case you would have modified the initial condition using *setFields*.
Modify gravity orientation, so that the tank will be vertical:

```
$ gedit constant/g
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -2 0 0 0 0];
value           (0 0 -9.81);


// ************************************************************************* //
```

Setting up parallel computing

OpenFOAM runs in parallel using a decomposition of mesh and fields. You can specify the cut yourself or ask *scotch* to do it for you (our call here). As a good rule of thumb, each processor should at least have 30000 cells to compute, if you want to see an increase in computational speed. Copy the decomposition dictionary from an existing case:

```
$ cp $FOAM_TUTORIALS/multiphase/interFoam/RAS/damBreak/damBreak/system/decomposeParDict
system/.
```

Modify the decomposition dictionary:

```
$ gedit system/decomposeParDict
```

Once it is done, decompose the case:

```
$ decomposePar
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      decomposeParDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 2;

method          scotch;

// ************************************************************************* //
```

Running the case

The tank takes about 40 seconds to fill up, which represented a cpu time of 7 hours for me. So set *'endTime'* according to your

liking. I would advice 2 second. It still requires 20 minutes to run. Open case control file:

```
$ gedit system/controlDict
```

The mesh has already been built. The case has been split between processors, so let's run the case:

```
$ mpirun -np 2 interFoam -parallel
```

OR

```
$ foamJob -s -p interFoam
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     interFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         2;

deltaT          0.001;

writeControl    adjustableRunTime;

writeInterval   0.05;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo           1;
maxAlphaCo      1;

maxDeltaT       1;

// ************************************************************************* //
```

Postprocessing the case

The case can then be processed. First, create a new directory to store the pictures that we will use to build the animation:

```
$ mkdir Extraction
```

Optionally, you can reconstruct the data (they are divided between the *'processorX'* folders):

```
$ reconstruct
```

Then launch the post processor:

```
$ paraFoam
```

Load all the fields, then set opacity of the base filter to 0.1:

```
$ Ex10.OpenFOAM / Style / Opacity
```

Add a contour filter, on alpha.water field, with a threshold of 0.1:

```
$ Filters / Common / Contour
```

Change the background color to white:

```
$ Edit / View Settings ... / Choose color
```

Set the camera angle to your liking by using the mouse or the 'Adjust Camera' tool.
Save the individual picture:

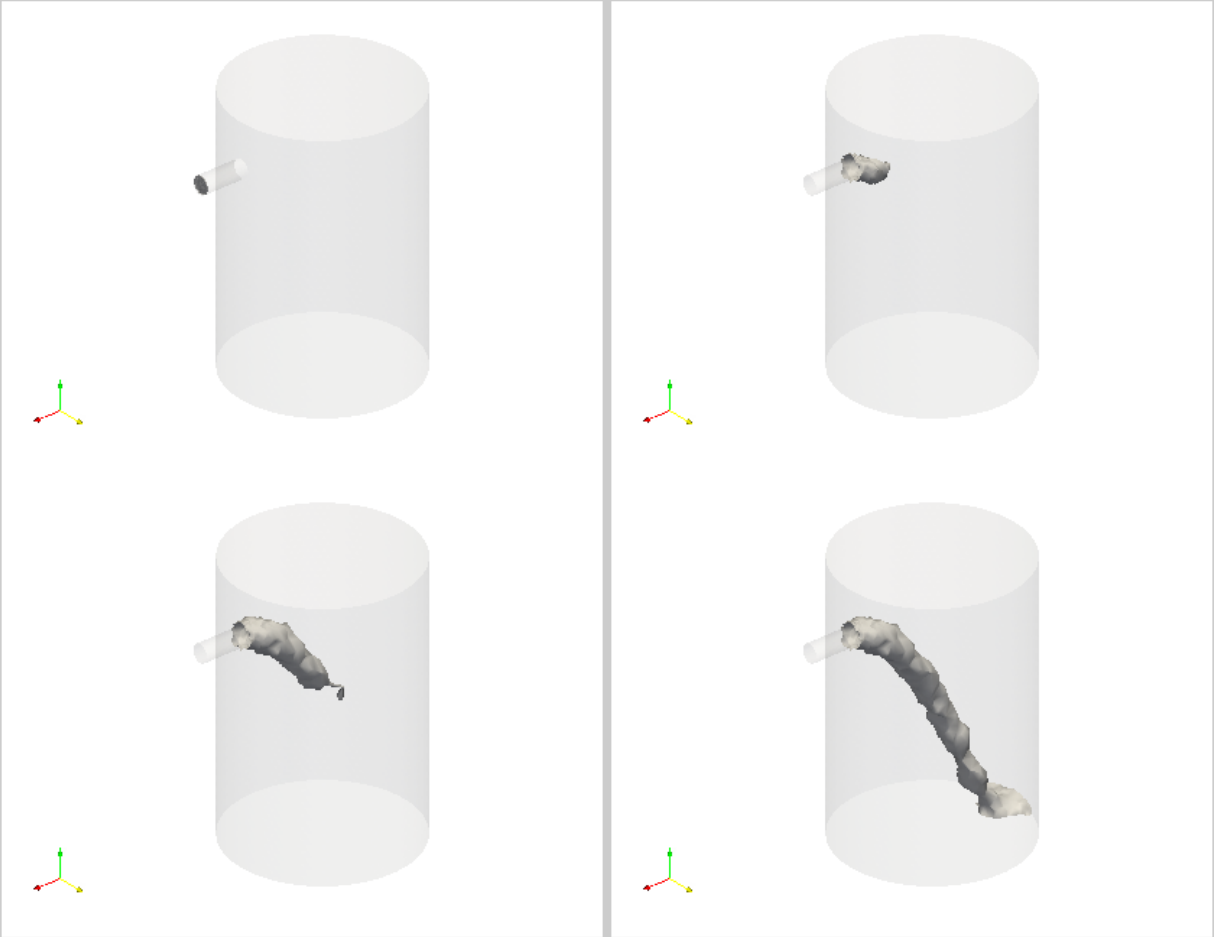
```
$ File / Save Animation ...
```

Click *'Save Animation'* on the prompted form (the settings do not matter here)
Pick *'Extraction'* folder and choose a name for the pictures (I chose 'Water')
Now paraFoam is filling *'Extraction'* directory with frames of the animation
Close paraFoam and go '*Extraction'* folder:

```
$ cd Extraction
```

Create your .gif file:

```
$ convert -delay 10 -loop 0 Water.*.png Your.gif
```



It's over

This tutorial is over, thank you for your attention.I hope you enjoyed it. It was just a brief and partial overview of OpenFOAM possibilities. Please feel free to contact me: victor.pozzobon@centralesupelec.fr

The extra mile
The open source software, I use to draw and mesh complex geometries: *SALOME* - www.salome-platform.org
Another open source software which can be used to process high volume results: *VisIt* - https://visit.llnl.gov
Where I ask for help: *CFD Online*: www.cfd-online.com/Forums/openfoam

View publication stats