4. Laminar flow through a circular pipe

4.0.6 Description of the case

As done in Chapter 2, this tutorial studies confined flow. Nevertheless, in the current chapter one of the most representative cases of internal fluid is presented: flow through a circular pipe. It is a very typical problem in the fluid mechanics field because of its wide presence in a great number of experiments, analysis and our daily life. As it is the basic fluid conveyance, the circular pipe and its influence on the circulating flow behaviour has been widely studied in the literature.

4.0.7 Hypotheses

- Incompressible flow
- Laminar flow
- Newtonian flow
- Cylindrical simmetry ($\frac{\partial}{\partial \theta} = 0$)
- Negligible gravitatory effects
- Steady flow ($\frac{\partial}{\partial t} = 0$)
- Very large pipe so that when flow becomes fully developed, $v_z \neq 0$ but $v_\theta = 0$, $v_r = 0$
- Smooth pipe so roughness does not influence

4.0.8 Physics of the problem

The problem encompasses a $L = 0.2$ m length circular pipe with a diameter of $D = 8$ mm. An inlet volumetric flow rate of $Q = 25.6\ cm^3/\text{s}$ and an outlet pressure

of $p_{outlet} = 9000$ Pa are imposed. For the mathematical introductory analysis and due to the fact that the case is axi-symmetric, cylindrical coordinates (r, θ, z) are used. z is parallel to the axis of the pipe, r points to the wall of the pipe and is perpendicular to the line tangent to the contour in the intersection point, and θ completes the coordinate system. The problem statement is shown at Figure 4.1.

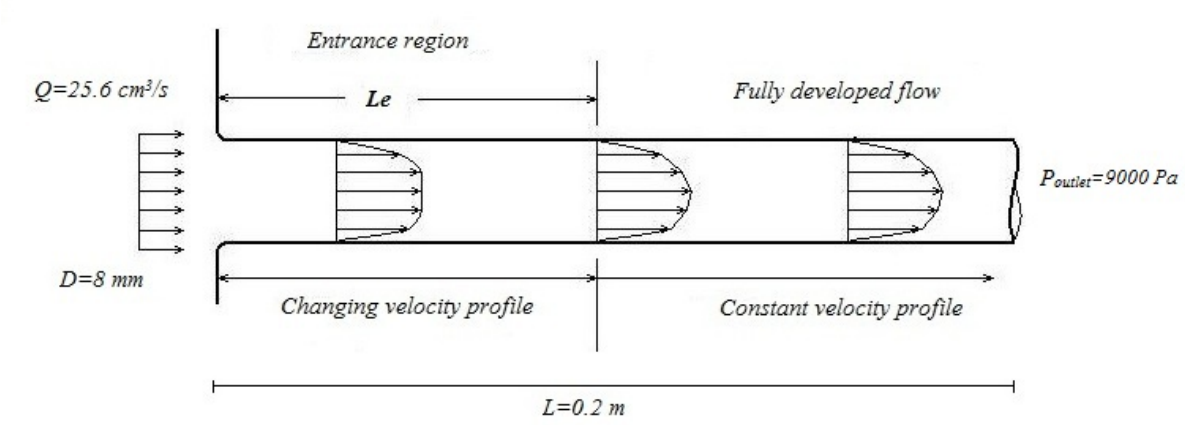


Figure 4.1: Flow through a circular pipe with a constant inlet velocity

As it can be seen, when the inlet flow enters the pipe, the viscous boundary layers grow up downstream, slowing the axial flow on the wall while accelerating the central core to mantain continuity (Equation 4.2).

In a finite distance from the inlet (entrance length, L_e), the boundary layers join and the fluid becomes entirely viscous (fully developed flow). When it happens, the velocity profile becomes constant, the wall shear stress becomes constant and the pressure changes linearly with z. For laminar flow, the entrance length is commonly accepted as being:

$$\frac{L_e}{D} \approx 0.06\ Re \quad (laminar) \tag{4.1}$$

According to the hypotheses, if the pipe is very large and for $z > L_e$, the only non-zero component of the velocity is v_z. In this domain, there is an easy mathematical solution to describe the behaviour of the flow. Thus, the current CFD analysis will be useful to corroborate this anaylitical model and to figure out the behaviour of the flow for $z < L_e$, which is more complex to describe mathematically.

For $z > L_e$, the continuity equation in cylindrical coordinates is

$$\frac{1}{r}\frac{\partial}{\partial r}(rv_r) + \frac{1}{r}\frac{\partial v_\theta}{\partial \theta} + \frac{\partial v_z}{\partial z} = 0 \Rightarrow \frac{\partial v_z}{\partial z} = 0 \Rightarrow v_z = v_z(r) \tag{4.2}$$

The momentum equation for v_z,

$$\frac{\partial v_z}{\partial t} + v_r\frac{\partial v_z}{\partial r} + v_\theta\frac{1}{r}\frac{\partial v_z}{\partial \theta} + v_z\frac{\partial v_z}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial z} + \frac{\mu}{\rho}\left[\frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial v_z}{\partial r}) + \frac{1}{r^2}\frac{\partial^2 v_z}{\partial \theta^2} + \frac{\partial^2 v_z}{\partial z^2}\right] \tag{4.3}$$

is reduced to

$$\frac{\mu}{r}\frac{d}{dr}\left(r\frac{dv_z}{dr}\right) = \frac{dp}{dz}$$

and as $p = p(z)$ according to the momentum equation for v_r, the right hand side of the equation is constant and < 0.

Integrating twice, applying the boundary condition $v_z = 0$ in $r = \frac{D}{2}$ and knowing that the first constant must be 0 because $ln\ 0$ is a singularity in $r = 0$, the analytical solution takes the form

$$v_z = \left(-\frac{dp}{dz}\right)\frac{1}{4\mu}\left(\frac{D^2}{4} - r^2\right) \tag{4.4}$$

This is the general solution of the flow through a circular pipe for $z > L_e$, a parabolic velocity profile (Hagen-Poiseuille flow). Additionally, an interesting value to be computed is the maximum velocity corresponding to the vertex of the parabola, which is

$$v_{z_{max}} = \left(-\frac{dp}{dz}\right)\frac{D^2}{16\mu} \tag{4.5}$$

For internal flow, the transition from laminar to turbulent happens at a Reynolds number of about $Re \approx 2300$. Therefore, as the simulation is made for laminar flow, its physical properties will be set accordingly: an oil is used with $\rho = 900\ kg/m^3$ and $\mu = 0.0288$ Pa·s. Then the Reynolds number is

$$Re = \frac{\rho DV}{\mu} = 65.48$$

4.0.9 Pre-processing

The following codes contain the information to simulate the circular pipe with $Re = 65.48$ using icoFoam. The case directory is named circularPipe and will be located

within FoamCases. Its structure of directories and subdirectories is very similar to the one used in Chapter 2 and Chapter 3.

4.0.9.1 Mesh generation

The mesh of the circularPipe case includes a new feature: a block with fewer than 8 vertices will be used. There is an axi-symmetry in the case so it is not necessary to mesh the whole pipe but only a wedge of it. Since $\frac{\partial}{\partial \theta} = 0$ for the whole domain, the results of the simulation will be the same while the number of elements of the mesh will be lower. Figure 4.2 represents a schematic view of the geometry that it is going to be created:

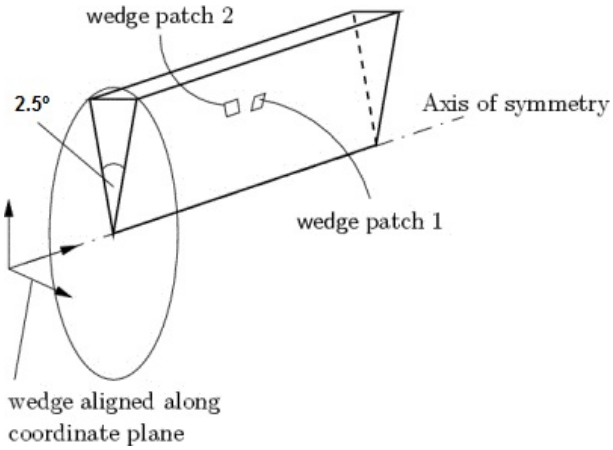
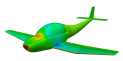


Figure 4.2: Scheme of the domain of the circularPipe case, extracted from [1]

The *blockMeshDict* contains:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 0.001;

```

```
19  vertices
20  (
21      (0 0 0)
22      (0 3.9990481 -0.0872595)
23      (0 3.9990481 0.0872595)
24      (200 0 0)
25      (200 3.9990481 -0.0872595)
26      (200 3.9990481 0.0872595)
27  );
28
29  blocks
30  (
31      hex (0 1 2 0 3 4 5 3) (50 1 100) simpleGrading (0.1 1 10)
32  );
33
34  edges
35  (
36     arc 1 2 (0 4 0)
37     arc 4 5 (200 4 0)
38  );
39
40  boundary
41  (
42      axis
43      {
44         type empty;
45         faces
46         (
47              (0 3 3 0)
48         );
49      }
50
51      inlet
52      {
53          type patch;
54          faces
55          (
56              (0 0 2 1)
57          );
58      }
59      wall
60      {
61          type wall;
62          faces
63          (
64              (2 5 4 1)
65          );
66      }
67      outlet
68      {
69          type patch;
70          faces
71          (
72              (3 4 5 3)
73          );
74      }
75
```



```
76      front
77      {
78          type wedge;
79          faces
80          (
81              (0 3 5 2)
82          );
83       }
84
85       back
86      {
87          type wedge;
88          faces
89          (
90              (0 1 4 3)
91          );
92       }
93  );
94
95  mergePatchPairs
96  (
97  );
98
99  // ************************************************************************* //
```

When running blockMesh, the created mesh is the following:

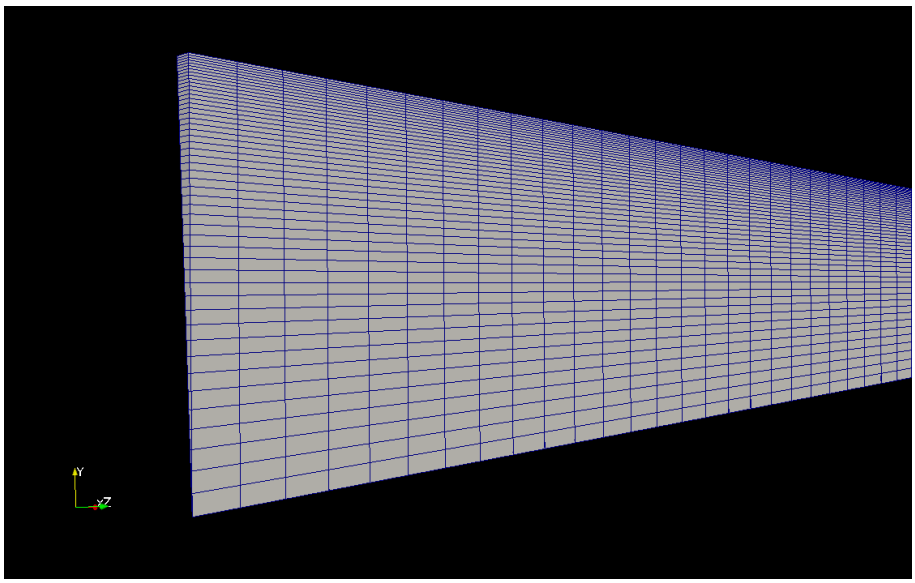


Figure 4.3: Initial mesh of the circularPipe case

First of all, it is important to comprehend the collapsing of the vertices. It can be seen at line 31 that vertices 0 and 3 are repeated in an adequate position within the

parentheses to indicate that the two vertices that would be occupying this position have been collapsed.

Secondly, in the patches definition, the same vertices 0 and 3 are joined creating an empty patch forming the axis of the pipe. The patch type wedge is used in two faces (front and back at lines 76 and 85) to indicate that an axi-symmetry exists and there are no physical walls (wedge patch 1 and wedge patch 2 in Figure 4.2).

Caution:

> The user should have noted that the vertices are written such that the wedge represents a 2.5º section of the pipe. To avoid errors, any edge of the wedge may coincide with the axis of the global coordinate system

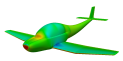
There is a grading towards the wall to compute a more exact value of wall shear stress and towards the inlet where the flow is not fully developed and therefore the velocity profile changes.

4.0.9.2 Boundary and initial conditions

Advice:

> When solving circulating internal flow, the most common boundary conditions are to fix an inlet velocity and an outlet pressure (or viceversa)

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.1.1                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       volScalarField;
13     object      p;
14 }
15 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
16
17 dimensions      [0 2 -2 0 0 0 0];
18
19 internalField   uniform 0;
20
21 boundaryField
22 {
23     axis
24     {
```



```
25          type            empty;
26      }
27
28      inlet
29      {
30          type            zeroGradient;
31      }
32
33      wall
34      {
35          type            zeroGradient;
36      }
37
38      outlet
39      {
40          type            fixedValue;
41          value           uniform 10;
42      }
43
44      front
45      {
46          type            wedge;
47      }
48
49      back
50      {
51          type            wedge;
52      }
53  }
54
55  // ************************************************************************* //
```

```
1   /*--------------------------------*- C++ -*----------------------------------*\
2   | =========                 |                                                 |
3   | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4   |  \\    /   O peration     | Version:  2.1.1                                 |
5   |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6   |    \\/     M anipulation  |                                                 |
7   \*---------------------------------------------------------------------------*/
8   FoamFile
9   {
10      version     2.0;
11      format      ascii;
12      class       volVectorField;
13      object      U;
14  }
15  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
16
17  dimensions      [0 1 -1 0 0 0 0];
18
19  internalField   uniform (0.50929 0 0);
20
21  boundaryField
22  {
23      axis
24      {
```

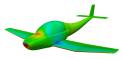
```
25          type            empty;
26      }
27
28      inlet
29      {
30          type            fixedValue;
31          value           uniform (0.50929 0 0);
32
33      }
34
35      wall
36      {
37          type            fixedValue;
38          value           uniform (0 0 0);
39      }
40
41      outlet
42      {
43          type                inletOutlet;
44          inletValue uniform  (0.50929 0 0);
45          value uniform       (0.50929 0 0);
46      }
47
48      front
49      {
50          type            wedge;
51      }
52
53      back
54      {
55          type            wedge;
56      }
57  }
58
59  // ************************************************************************* //
```

The inlet velocity has been computed as:

$$V = \frac{Q}{\pi \cdot (D/2)^2} = \frac{25.6 \times 10^{-6}}{\pi \cdot (4 \times 10^{-3})^2} = 0.50929\ m/s$$

In the **U** dictionary one finds a new type of boundary condition: inletOutlet. It switches **U** and p between fixedValue and zeroGradient: the first when the flow is ingoing and the second when it is outgoing.

Example: By applying it in the circularPipe case, the outlet velocity will always be adapted to the conditions according to zeroGradient except if there is an inflow at the outlet patch. If this happens, to this inflow velocity it is assigned the value specified under the inletOutlet instruction and so the flow will always be outgoing.



4.0.9.3 Physical properties

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              nu [ 0 2 -1 0 0 0 0 ] 0.000032;

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

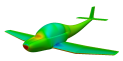
```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      RASProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

RASModel        laminar;

turbulence      off;

printCoeffs     off;

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

4.0.9.4 Control

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     icoFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         0.2;

deltaT          0.00005;

writeControl    timeStep;

writeInterval   20;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

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

4.0.9.5 Discretization and linear-solver settings

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
```



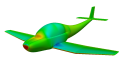
```
5   |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6   |    \\/     M anipulation  |                                                 |
7   \*---------------------------------------------------------------------------*/
8   FoamFile
9   {
10      version     2.0;
11      format      ascii;
12      class       dictionary;
13      location    "system";
14      object      fvSchemes;
15  }
16  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18  ddtSchemes
19  {
20      default         Euler;
21  }
22
23  gradSchemes
24  {
25      default         Gauss linear;
26      grad(p)         Gauss linear;
27  }
28
29  divSchemes
30  {
31      default         none;
32      div(phi,U)      Gauss linear;
33  }
34
35  laplacianSchemes
36  {
37      default         none;
38      laplacian(nu,U) Gauss linear corrected;
39      laplacian((1|A(U)),p) Gauss linear corrected;
40  }
41
42  interpolationSchemes
43  {
44      default         linear;
45      interpolate(HbyA) linear;
46  }
47
48  snGradSchemes
49  {
50      default         corrected;
51  }
52
53  fluxRequired
54  {
55      default         no;
56      p               ;
57  }
58
59  // ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-07;
        relTol          0;
    }

    U
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-06;
        relTol          0;
    }
}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
}

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

Advice:

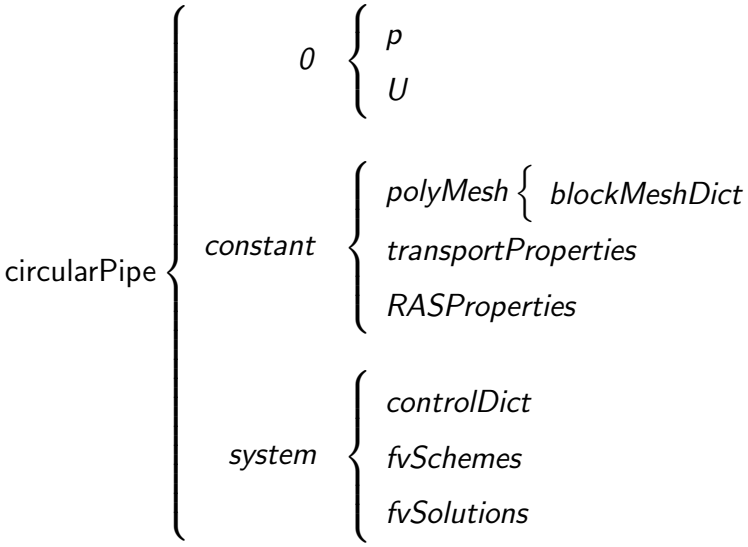
In the *fvSolution* file of the circularPipe case, the absolute tolerances are set lower than other cases. The tolerances are indicative of the error that the user *accepts* in the iterative resolution procedure. The lower the tolerances, the more accurate the results (but lower the simulation speed)

Advice:



Below the solver definition, it has to be specified PISO or SIMPLE and its characteristics. These algorithms are iterative procedures for solving equations for velocity and pressure, PISO being used for transient problems and SIMPLE for steady-state. When using SIMPLE, nCorrectors has to be set to 1, whereas when using PISO, it should be higher than 1 but typically not more than 4

At the end of the pre-processing of the circularPipe case, the scheme of directories and sub-directories is the following:



4.0.10 Post-processing

4.0.10.1 Results of the simulation

The velocity field at the inlet of the pipe (non-developed flow):

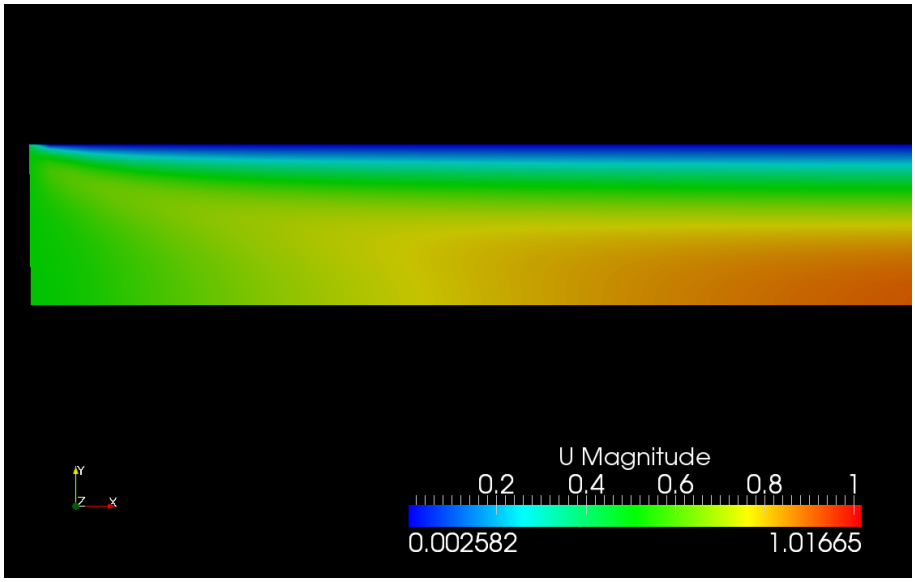


Figure 4.4: Velocity field at the inlet of the pipe in the circularPipe case (m/s)

The velocity field at the outlet of the pipe (developed flow):

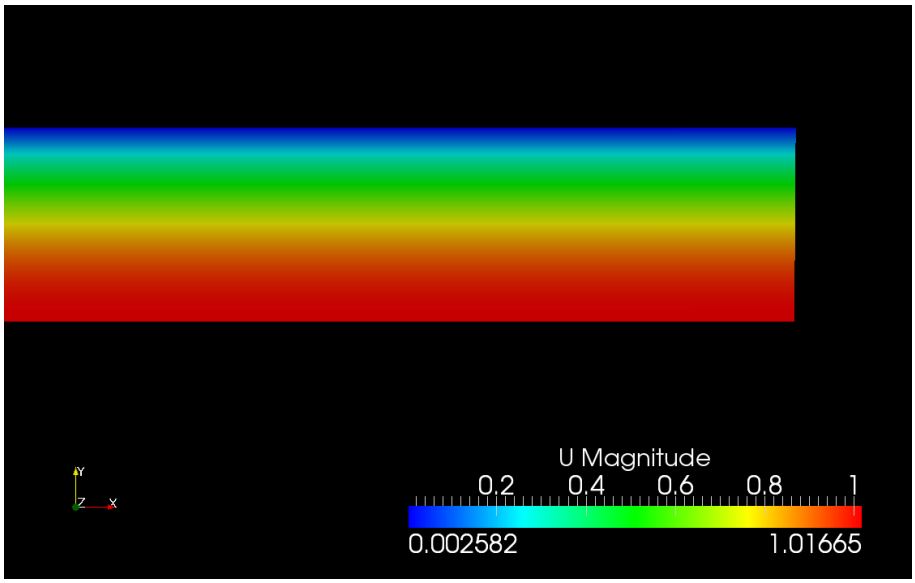


Figure 4.5: Velocity field at the outlet of the pipe in the circularPipe case (m/s)

The pressure field along the pipe:

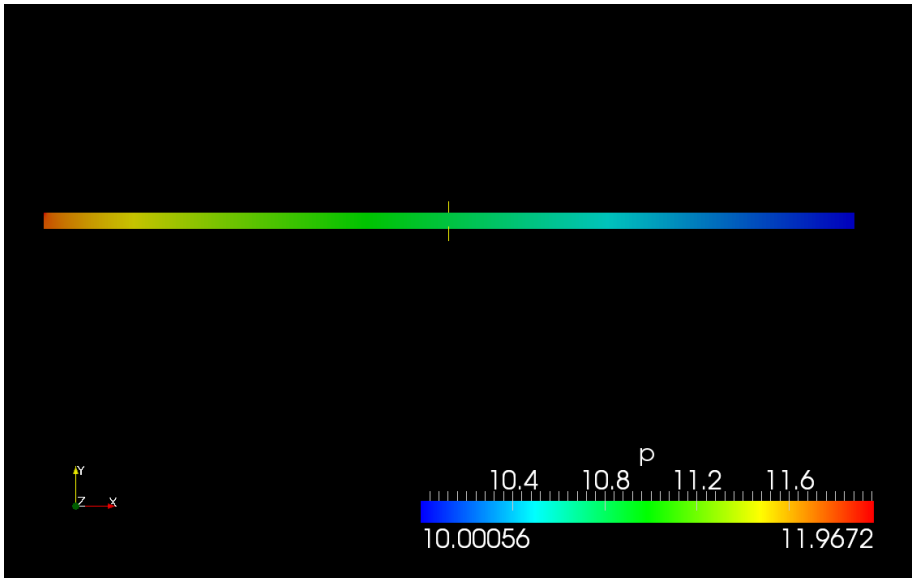


Figure 4.6: Pressure field in the circularPipe case (m^2/s^2)

The streamlines at the inlet of the pipe (non-developed flow):

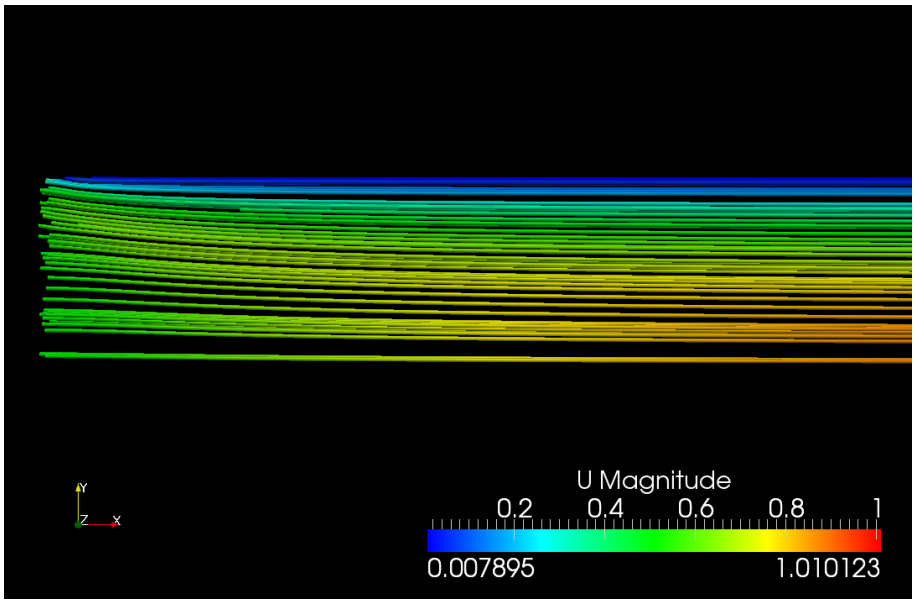


Figure 4.7: Streamlines of the flow at the inlet of the pipe in the circularPipe case (m/s)

The streamlines at the outlet of the pipe (developed flow):

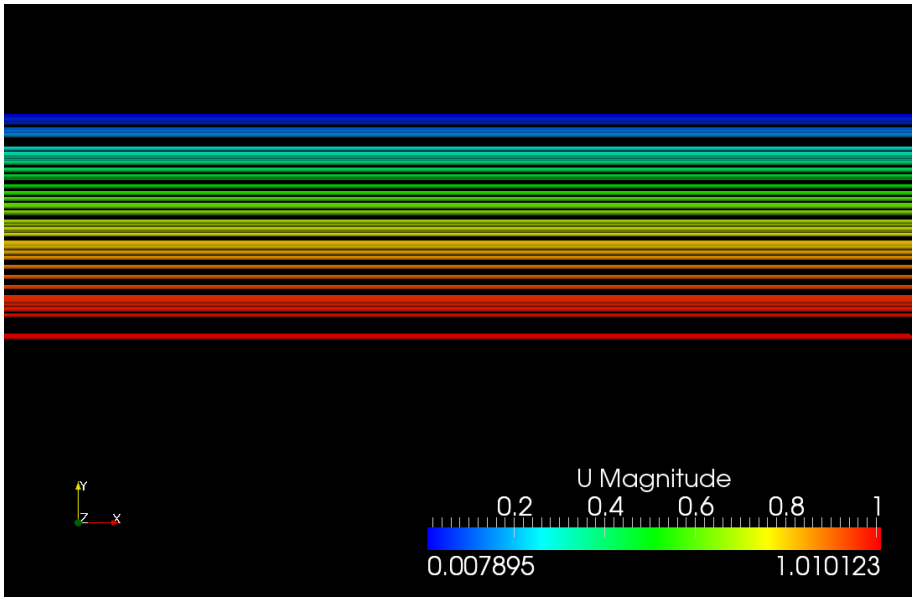


Figure 4.8: Streamlines of the flow at the outlet of the pipe in the circularPipe case (m/s)

To observe if the analytical equation of the velocity for $z > L_e$ (developed flow) shown in Section 4.0.8 matches with the results of the simulation, $|\mathbf{U}|$ at outlet as a function of r has been plot:

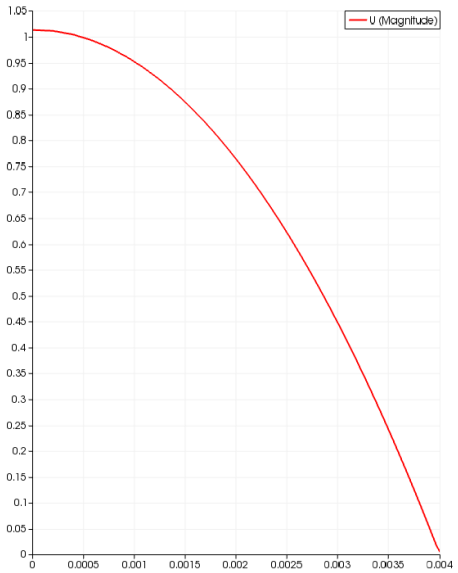
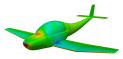


Figure 4.9: Plot of $|\mathbf{U}|$ (ordinate axis) as a function of r (abscissa axis) at the outlet of the pipe in the circularPipe case



As it can be seen, the velocity profile is a branch of a parabola. According to Equation 4.5, the maximum velocity at a section with developed flow is $v_{z_{max}} = 1.0190$ m/s. According to the results of the simulation, the maximum velocity is $|\mathbf{U}| = 1.0167$ m/s. It represents a relative error of $e_r = 0.226\%$.

Caution:

> Unlike in the *plane-Poiseuille flow* case, $\frac{dp}{dz}$ cannot be computed as $\frac{p_{outlet} - p_{inlet}}{L}$ because in the region of non-developed flow, the pressure gradient is not linear. Figure 4.10 can be used to compute it

According to the initial explanations, the pressure of the domain is linear for $z > L_e$ but not for $z < L_e$. This trend can be observed in the results of the simulation:

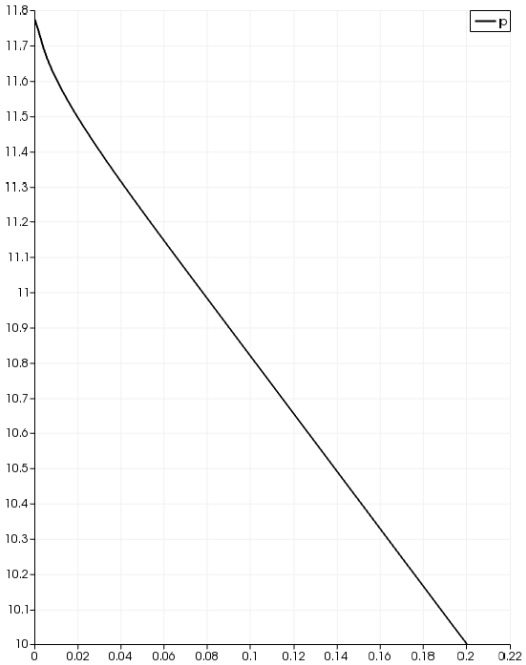


Figure 4.10: Plot of p (ordinate axis) as a function of z (abscissa axis) in the circularPipe case

Finally, it is shown the plot of the wall shear stress as a function of z:

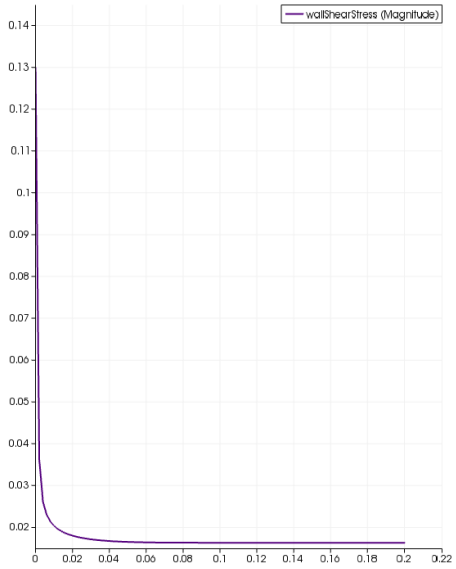


Figure 4.11: Plot of τ_w (ordinate axis) as a function of z (abscissa axis) in the circularPipe case

As it can be seen, the wall shear stress (τ_w) turns to constant for $z > L_e$ (obviously, inasmuch as the velocity profile turns to constant too). This plot allows the user to observe an approximate value of L_e according to the numerical simulation and compare it to Equation 4.1.

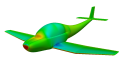
Caution:

> The wall shear stress has to be plotted and read when the patch wall is selected

4.0.11 Additional utilities

4.0.11.1 Computation of average field values at patches

In the current case, there is an imposed inlet velocity and an imposed outlet pressure. However, one variable that would be interesting to compute is the inlet pressure that exists in the pipe according to such boundary conditions. Knowing this factor may help the user in the computation of some useful values such as the pressure drop. There is a tool in *OpenFOAM®* to compute average values of a concrete field in any patch of the domain (previously defined in *blockMeshDict*). It is the patchAverage utility, which must be followed by the name of the field and the patch where it is calculated. For instance, to compute the average pressure at the inlet



while redirecting it to a file, type:

```
patchAverage p inlet > inletPressure
```

There, it is possible to observe that the solution stabilizes at $t = 0.159$ s and that the average value of the pressure at inlet is $p = 11.8239 \cdot 900 = 10641.51$ Pa. Therefore, the pressure drop within the pipe due to the viscous forces is 1641.51 Pa.

The same utility can be used to compute average vector fields such as $|\mathbf{U}|$. For instance, to corroborate that the continuity equation is fulfilled, type:

```
patchAverage U outlet > outletVelocity
```

By doing this, a file is generated with the average velocity vector at outlet. It can be seen that the average velocity is maintained between the inlet and the outlet.

4.0.11.2 Read field values with ParaView

Althoug when running icoFoam all the field data are stored within the subdirectories of the case, it is possible to read concrete field values while using ParaView. It is useful to obtain rapid data from the cells of the domain during the analysis of the results.

To do it, the user has to launch ParaView, select the last time of the simulation and check that the field or fields whose data are to be read are activated (in this example, p and $\mathbf{U}$). Then click on the rectangular icon (red circle at the top right corner of Figure 4.12) and select Spreadsheet View. It appears the information of the whole domain. To select only a specific cell or set of cells, click on the icon shown in the second red circle. Then, after clicking the icon surrounded by the purple circle, select the region of the geometry whose data are to be read.

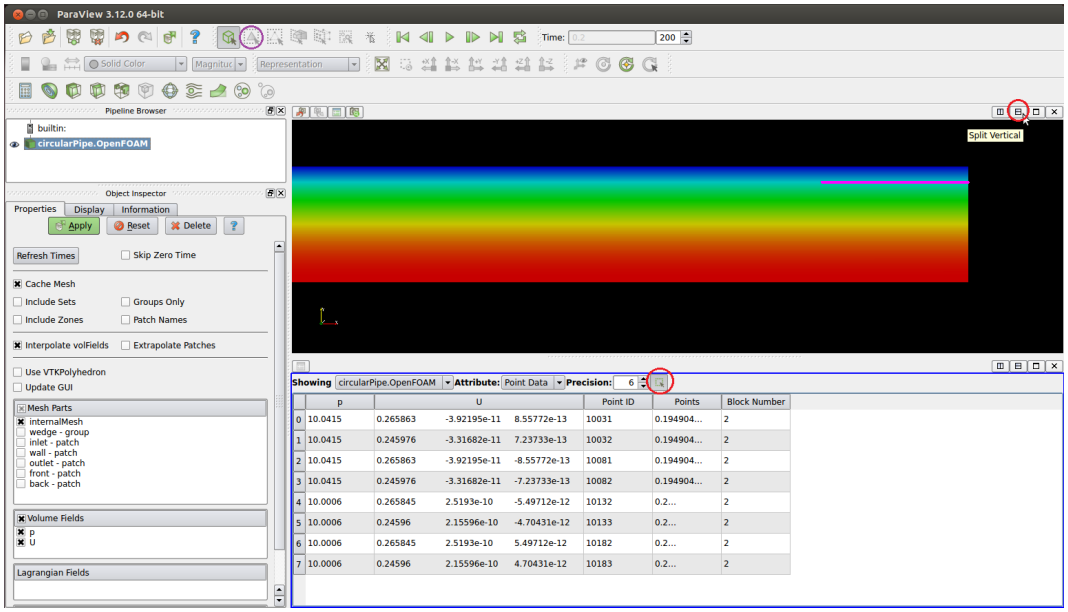


Figure 4.12: Menu to read field data in ParaView

4.0.11.3 Plot the residuals of the simulation

A fundamental point in the numerical simulations is the convergence of the solution. The resolutive procedure is an iterative process and by definition values are changing from one iteration to the next. Every numerical solution contains errors, but it is important to understand how big it is compared to the magnitude of the variable. If the solution does not converge, normally unphysical results are obtained.

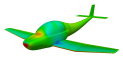
One way of measuring the convergence is with the residuals. It represents the absolute error in the solution of a particular variable. The system iterates until the residuals achieve the values of the tolerances defined in *controlDict*.

In the current section, it is shown how to plot these residuals. It may help the user in understanding if the solution has converged, and if so, how fast has it been.

Caution:

> A converging solution does not necessarily guarantee that the results of the simulation are correct. Factors such as a coarse mesh or a bad problem definition may cause the solver to produce inaccurate results

The graphical results of the residuals are going to be obtained by plotting the values within *log.icoFoam*, the file containing the information redirected while icoFoam was running. If it is not yet created, type:



```
icoFoam > log.icoFoam
```

The residuals are going to be plotted with Gnuplot by executing a Script. Copy within the case in a gedit sheet named executePlotResiduals the following instructions:

```
set logscale y
set title "Residuals"
set ylabel 'Residual'
set xlabel 'Iteration'
plot "< cat log.icoFoam | grep 'Solving for Ux' | cut -d' ' -f9" title 'Ux' with
    lines,\
"< cat log.icoFoam | grep 'Solving for Uy' | cut -d' ' -f9" title 'Uy' with lines,\
"< cat log.icoFoam | grep 'Solving for Uz' | cut -d' ' -f9" title 'Uz' with lines,\
"< cat log.icoFoam | grep 'Solving for p' | cut -d' ' -f9 | tr -d ',' " title 'p'
    with lines
pause 1000
```

Make the Script executable by typing:

```
sudo chmod a+x executePlotResiduals
```

Execute it to plot the results with Gnuplot by typing within the case:

```
gnuplot executePlotResiduals
```

The plots of the residuals are shown in Figures 4.13 and 4.14 (they are presented separately for a better understanding):

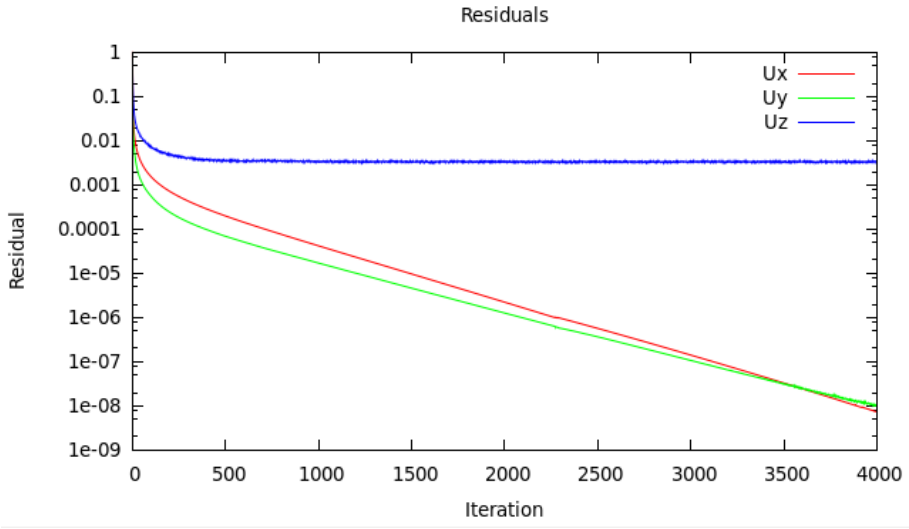


Figure 4.13: Residuals of the velocity in the circularPipe case

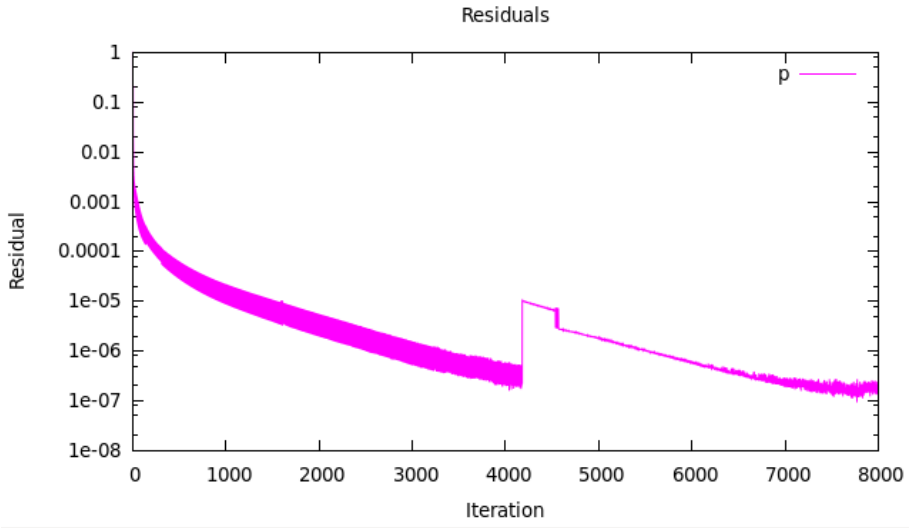


Figure 4.14: Residuals of the pressure in the circularPipe case