3. Bidimensional laminar flow around a circular cylinder

3.1 Description of the case

The second tutorial is based on the analysis of external flow. More specifically, the study encompasses the modelling of a bidimensional laminar flow around a circular cylinder. The analytical solution of the same case modeled with potential flow is known and easy to interpret; however the physical mechanism which rules *real* flow around a circular cylinder as well as its mathematical treatment are highly complex. Some examples are the boundary layer detachment and oscillatory effects that without a CFD study would be very difficult to determine either analytically or experimentally. The case includes simulations made with two different Reynolds numbers to observe the behaviour of the flow in such different regimes.

3.2 Hypotheses

- Incompressible flow
- Laminar flow
- Newtonian flow
- Bidimensional flow ($\frac{\partial}{\partial z} = 0$)
- Negligible gravitatory effects

3.3 Physics of the problem

The problem encompasses a circular cylinder with a diameter of $D = 0.1$ m submerged in a stream with an undisturbed speed of $V = 5$ m/s and ambient pressure. The problem statement is shown at Figure 3.1.

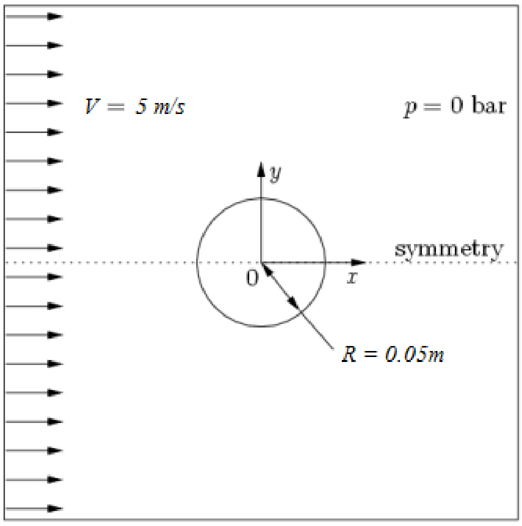


Figure 3.1: Flow around a circular cylinder

Unlike Chapter 2, in the current chapter there is no easy analytical solution to describe the behaviour of the fluid (except for potential flow, which is an ideal case). However, it is necessary to keep in mind the main equations and dimensionless numbers involved in the problem:

The continuity equation,

$$\nabla \cdot \mathbf{U} = 0 \tag{3.1}$$

The momentum equation,

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{U} \cdot \nabla \mathbf{U} = -\frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2 \mathbf{U} \tag{3.2}$$

The Reynolds number,

$$Re = \frac{D|\mathbf{U}|}{\nu} \tag{3.3}$$

The Strouhal number (which describes oscillating flow mechanisms),

$$St = \frac{wD}{2\pi|\mathbf{U}|} \tag{3.4}$$

The drag coefficient (dimensionless force parallel to the flow),

$$C_D = \frac{F_D}{\frac{1}{2}\rho|\mathbf{U}|^2 S} \tag{3.5}$$

The behaviour of the flow around a circular cylinder is very complex and highly dependent on the Reynolds number. The boundary layer in laminar regimes detaches at an angle of $\theta = 82^{\circ}$ as a consequence of the existence of an adverse pressure gradient. Furthermore, due to the symmetry of the cylinder, a curious phenomenon appears named *von Kármán street*, consisting of alternate vortices emitted by the cylinder. The vortices are detached periodically and their frequence is directly related to the Strouhal number.

For external flow, the transition from laminar regime to turbulent regime occurs at $Re \approx 3 \times 10^5$. However, the very complex structure of vortex shedding deserves special mention; the relation between the behaviour of the flow and the Reynolds number is shown at Figure 3.2:

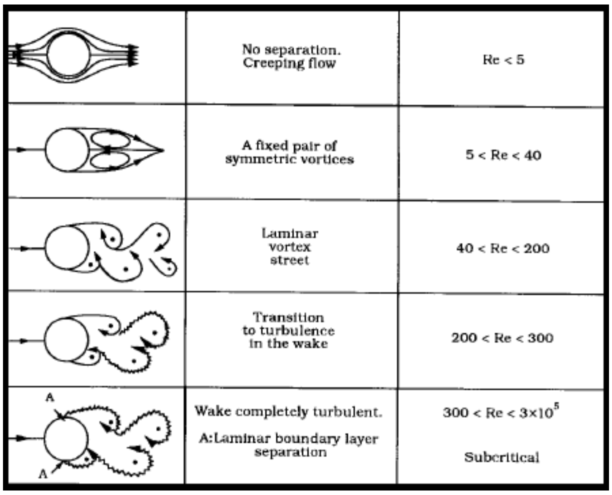


Figure 3.2: Flow structure depending on the Reynolds number, extracted from [2]

As the simulation is going to be run with laminar flow, the Reynolds number is not going to be set higher than 195. This will allow to observe the von Kármán street and its transition while guaranteeing that the whole fluid domain remains laminar. Additionally, another simulation with $Re = 30$ will be run to check if the results of the CFD simulation adapt to the scheme shown at Figure 3.2.

The drag coefficient is a function of the Reynolds number and decreases when the regime turns to turbulent. It can be seen at Figure 3.3.

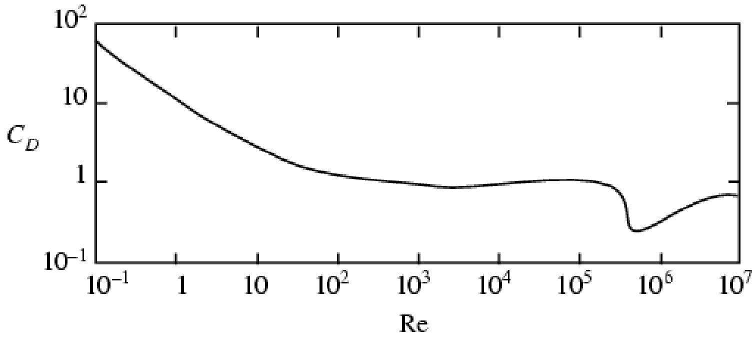


Figure 3.3: Drag coefficient as a function of the Reynolds number in an infinite circular cylinder

3.4 Pre-processing with $Re = 195$

The following codes contain the information to simulate the cylinder with $Re = 195$ using icoFoam. The case directory is named cylinder195 and will be located within FoamCases. Its structure of directories and subdirectories is very similar to the one used in Chapter 2.

3.4.1 Mesh generation

The mesh for the study of the cylinder is not going to be uniform. Some areas of the domain need to contain a higher cell density (mainly the walls of the cylinder and its prolongation). Consequently, it is necessary to divide the domain in different blocks as Figure 3.4 shows.

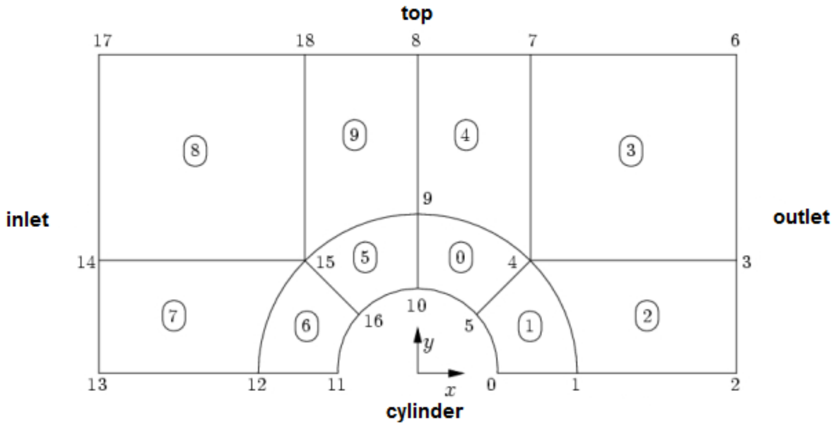


Figure 3.4: Half of the scheme used for the creation of the mesh, extracted from [1]

In Figure 3.4 there is only half of the mesh, which is contained between $z = -0.05$ m and $z = 0.05$ m. The numbers indicate the numbering used for the vertices and the blocks in the *blockMeshDict* file.

In the *bidimensional cylinder* case, *constant/blockMeshDict* must contain the following instructions:

```
/*--------------------------------*- 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;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 0.1;

vertices
(
   (0.5 0 -0.5)
   (1 0 -0.5)
   (10 0 -0.5)
   (10 0.707107 -0.5)
   (0.707107 0.707107 -0.5)
   (0.353553 0.353553 -0.5)
   (10 2 -0.5)
   (0.707107 2 -0.5)
   (0 2 -0.5)
```

```
    (0 1 -0.5)
    (0 0.5 -0.5)

    (-0.5 0 -0.5)
    (-1 0 -0.5)
    (-2 0 -0.5)
    (-2 0.707107 -0.5)
    (-0.707107 0.707107 -0.5)
    (-0.353553 0.353553 -0.5)
    (-2 2 -0.5)
    (-0.707107 2 -0.5)

    (0.5 0 0.5)
    (1 0 0.5)
    (10 0 0.5)
    (10 0.707107 0.5)
    (0.707107 0.707107 0.5)
    (0.353553 0.353553 0.5)
    (10 2 0.5)
    (0.707107 2 0.5)
    (0 2 0.5)
    (0 1 0.5)
    (0 0.5 0.5)

    (-0.5 0 0.5)
    (-1 0 0.5)
    (-2 0 0.5)
    (-2 0.707107 0.5)
    (-0.707107 0.707107 0.5)
    (-0.353553 0.353553 0.5)
    (-2 2 0.5)
    (-0.707107 2 0.5)


    (10 -0.707107 -0.5)
    (0.707107 -0.707107 -0.5)
    (0.353553 -0.353553 -0.5)
    (10 -2 -0.5)
    (0.707107 -2 -0.5)
    (0 -2 -0.5)
    (0 -1 -0.5)
    (0 -0.5 -0.5)

    (-2 -0.707107 -0.5)
    (-0.707107 -0.707107 -0.5)
    (-0.353553 -0.353553 -0.5)
    (-2 -2 -0.5)
    (-0.707107 -2 -0.5)

    (10 -0.707107 0.5)
    (0.707107 -0.707107 0.5)
    (0.353553 -0.353553 0.5)
    (10 -2 0.5)
    (0.707107 -2 0.5)
    (0 -2 0.5)
    (0 -1 0.5)
    (0 -0.5 0.5)
```

```

    (-2 -0.707107 0.5)
    (-0.707107 -0.707107 0.5)
    (-0.353553 -0.353553 0.5)
    (-2 -2 0.5)
    (-0.707107 -2 0.5)
);

blocks
(
    hex (5 4 9 10 24 23 28 29) (80 20 1) simpleGrading (10 1 1)
    hex (0 1 4 5 19 20 23 24) (80 20 1) simpleGrading (10 1 1)
    hex (1 2 3 4 20 21 22 23) (200 20 1) simpleGrading (1 1 1)
    hex (4 3 6 7 23 22 25 26) (200 40 1) simpleGrading (1 1 1)
    hex (9 4 7 8 28 23 26 27) (20 40 1) simpleGrading (1 1 1)
    hex (16 10 9 15 35 29 28 34) (20 80 1) simpleGrading (1 10 1)
    hex (11 16 15 12 30 35 34 31) (20 80 1) simpleGrading (1 10 1)
    hex (12 15 14 13 31 34 33 32) (20 20 1) simpleGrading (1 1 1)
    hex (15 18 17 14 34 37 36 33) (40 20 1) simpleGrading (1 1 1)
    hex (9 8 18 15 28 27 37 34) (40 20 1) simpleGrading (1 1 1)


    hex (40 45 44 39 53 58 57 52) (20 80 1) simpleGrading (1 10 1)
    hex (0 40 39 1 19 53 52 20) (20 80 1) simpleGrading (1 10 1)
    hex (1 39 38 2 20 52 51 21) (20 200 1) simpleGrading (1 1 1)
    hex (39 42 41 38 52 55 54 51) (40 200 1) simpleGrading (1 1 1)
    hex (44 43 42 39 57 56 55 52) (40 20 1) simpleGrading (1 1 1)
    hex (48  47 44 45 61 60 57 58) (80 20 1) simpleGrading (10 1 1)
    hex (11 12 47 48 30 31 60 61) (80 20 1) simpleGrading (10 1 1)
    hex (12 13 46 47 31 32 59 60) (20 20 1) simpleGrading (1 1 1)
    hex (47 46 49 50 60 59 62 63) (20 40 1) simpleGrading (1 1 1)
    hex (44 47 50 43 57 60 63 56) (20 40 1) simpleGrading (1 1 1)
);

edges
(
    arc 0 5 (0.469846 0.17101 -0.5)
    arc 5 10 (0.17101 0.469846 -0.5)
    arc 1 4 (0.939693 0.34202 -0.5)
    arc 4 9 (0.34202 0.939693 -0.5)
    arc 19 24 (0.469846 0.17101 0.5)
    arc 24 29 (0.17101 0.469846 0.5)
    arc 20 23 (0.939693 0.34202 0.5)
    arc 23 28 (0.34202 0.939693 0.5)
    arc 11 16 (-0.469846 0.17101 -0.5)
    arc 16 10 (-0.17101 0.469846 -0.5)
    arc 12 15 (-0.939693 0.34202 -0.5)
    arc 15 9 (-0.34202 0.939693 -0.5)
    arc 30 35 (-0.469846 0.17101 0.5)
    arc 35 29 (-0.17101 0.469846 0.5)
    arc 31 34 (-0.939693 0.34202 0.5)
    arc 34 28 (-0.34202 0.939693 0.5)


    arc 0 40 (0.469846 -0.17101 -0.5)
    arc 40 45 (0.17101 -0.469846 -0.5)
    arc 1 39 (0.939693 -0.34202 -0.5)
```

```
    arc 39 44 (0.34202 -0.939693 -0.5)
    arc 19 53 (0.469846 -0.17101 0.5)
    arc 53 58 (0.17101 -0.469846 0.5)
    arc 20 52 (0.939693 -0.34202 0.5)
    arc 52 57 (0.34202 -0.939693 0.5)
    arc 11 48 (-0.469846 -0.17101 -0.5)
    arc 48 45 (-0.17101 -0.469846 -0.5)
    arc 12 47 (-0.939693 -0.34202 -0.5)
    arc 47 44 (-0.34202 -0.939693 -0.5)
    arc 30 61 (-0.469846 -0.17101 0.5)
    arc 61 58 (-0.17101 -0.469846 0.5)
    arc 31 60 (-0.939693 -0.34202 0.5)
    arc 60 57 (-0.34202 -0.939693 0.5)
);

boundary
(

    top
    {
        type symmetryPlane;
        faces
        (
            (7 8 27 26)
            (6 7 26 25)
            (8 18 37 27)
            (18 17 36 37)
        );
    }

    bottom
    {
        type symmetryPlane;
        faces
        (
          (49 50 63 62)
          (50 43 56 63)
          (43 42 55 56)
          (42 41 54 55)
        );
    }

    inlet
    {
        type patch;
        faces
        (
            (14 13 32 33)
            (17 14 33 36)

            (46 13 32 59)
            (46 49 62 59)
        );
    }

    outlet
    {
```

```
        type patch;
        faces
        (
            (2 3 22 21)
            (3 6 25 22)

            (38 51 21 2)
            (41 54 51 38)
        );
    }

    cylinder
    {
        type wall;
        faces
        (
            (10 5 24 29)
            (5 0 19 24)
            (16 10 29 35)
            (11 16 35 30)

            (48 11 30 61)
            (45 48 61 58)
            (40 45 58 53)
            (0 40 53 19)
        );
    }

    frontAndBack
    {
        type empty;
        faces
        (
           (5 10 9 4)
           (24 23 28 29)
           (0 5 4 1)
           (19 20 23 24)
           (1 4 3 2)
           (20 21 22 23)
           (4 7 6 3)
           (23 22 25 26)
           (4 9 8 7)
           (28 23 26 27)
           (16 15 9 10)
           (35 29 28 34)
           (12 15 16 11)
           (31 30 35 34)
           (13 14 15 12)
           (32 31 34 33)
           (14 17 18 15)
           (33 34 37  36)
           (15 18 8 9)
           (34 28 27 37)

           (45 40 39 44)
           (58 57 52 53)
           (40 0 1 39)
```

```
            (53 52 20 19)
            (39 1 2 38)
            (52 51 21 20)
            (39 38 41 42)
            (52 55 54 51)
            (44 39 42 43)
            (57 56 55 52)
            (47 48 45 44)
            (60 57 58 61)
            (12 11 48 47)
            (31 60 61 30)
            (13 12 47 46)
            (32 59 60 31)
            (49 46 47 50)
            (62 63 60 59)
            (50 47 44 43)
            (63 56 57 60)
        );
    }
);

mergePatchPairs
(
);

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

It can be seen that as the cylinder has curved edges, it is necessary to use the arc instruction to obtain the circular geometry. This instruction must be followed by the labels of the connected vertices and an interpolation point contained in the trajectory of the arc.

In the current *blockMeshDict* there are four different types of patches: *wall*, *empty*, *symmetryPlane* and *patch*. All were used in the *plane-parallel plates* case except *symmetryPlane*, which is used in the top and bottom patches of the current case to indicate that there are no physical walls in the top and bottom borders; the flow must behave as if the domain would extend infinitely in the y-direction.

After running blockMesh and checking the results with checkMesh it is possible to observe the mesh with ParaView.

Caution:

> It is necessary to have the majority of the required files for the simulation within the case directory to create the mesh. Wait until the three main directories (*0*, *constant* and *system*) and their files are ready to run blockMesh. Otherwise, it is also possible to use a solved case as a *dummy* file to run blockMesh

The mesh of cylinder195 is shown at Figures 3.5, 3.6 and 3.7:

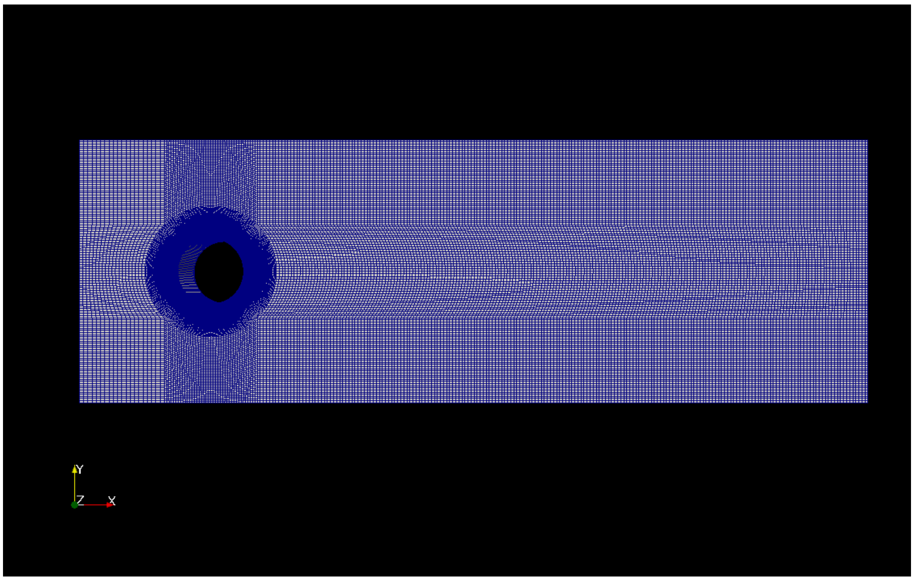


Figure 3.5: Mesh of the *bidimensional cylinder* case

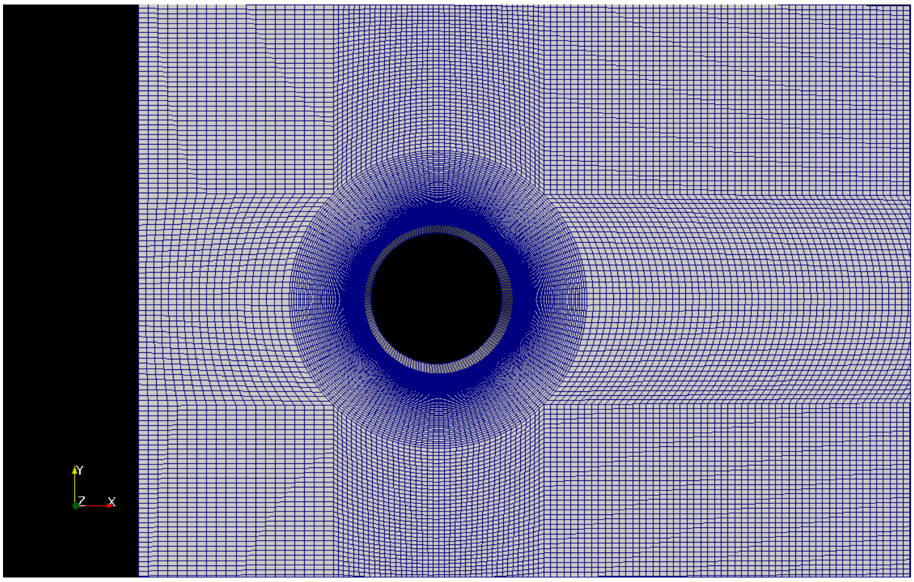


Figure 3.6: Detail of the mesh of the *bidimensional cylinder* case

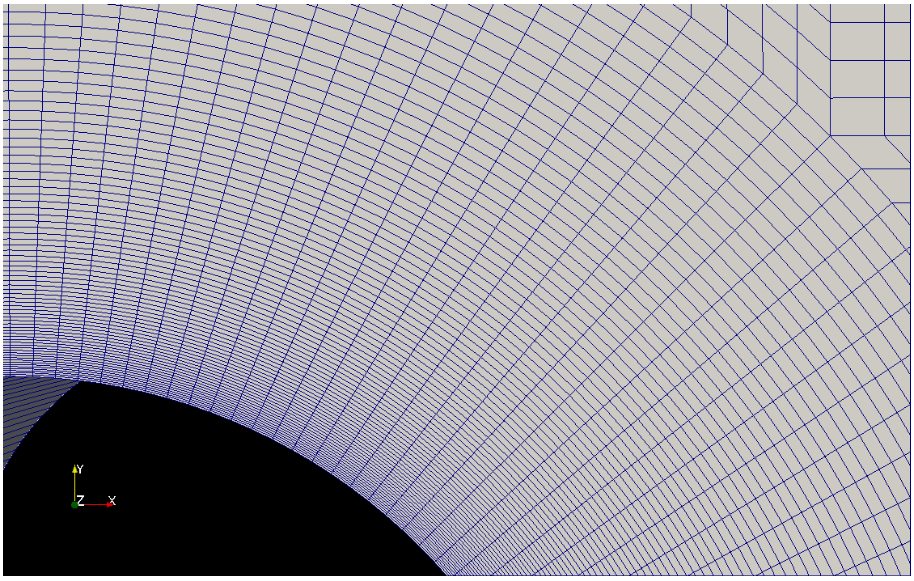


Figure 3.7: Detail of the mesh gradation on the walls of the *bidimensional cylinder* case

Advice:

> On the walls and downstream the cylinder it is necessary to have a high refined mesh. It will allow the user to compute the drag coefficient and to observe the von Kármán street both with high accuracy. However, if the simulation is too much time-consuming or it is done with a Reynolds number such that no vortices are generated, it is recommended to reduce the refinement of the mesh

3.4.2 Boundary and initial conditions

The files (located in *0*) containing the information related to the pressure and the velocity fields are the following:

```
/*--------------------------------*- 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       volScalarField;
```

```
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    top
    {
        type            symmetryPlane;
    }

    bottom
    {
        type            symmetryPlane;
    }

    inlet
    {
        type            freestreamPressure;
    }

    outlet
    {
        type            freestreamPressure;
    }

    frontAndBack
    {
        type            empty;
    }

    cylinder
    {
        type            zeroGradient;
    }
}

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

```
/*--------------------------------*- 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       volVectorField;
    object      U;
```

```
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField         uniform (5 0 0);

boundaryField
{
    top
    {
        type              symmetryPlane;
    }

    bottom
    {
        type              symmetryPlane;
    }

    inlet
    {
        type              freestream;
        freestreamValue   uniform (5 0 0);
    }

    outlet
    {
        type              freestream;
        freestreamValue   uniform (5 0 0);
    }

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

    frontAndBack
    {
       type               empty;
    }
}

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

The freestreamPressure condition is acting as zeroGradient but with a more accurate physical behaviour. freestream acts as fixedValue when the flow is ingoing, and as zeroGradient when it is outgoing. This kind of boundary conditions are widely used for external flow simulations.

3.4.3 Physical properties

Within constant one finds the information related to the kinematic viscosity and the RASProperties file to compute the wall shear stress or other utilities requesting the RAS dictionary.

```
/*--------------------------------*- 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      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              nu [0 2 -1 0 0 0 0] 2.564103e-03;

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

```
/*--------------------------------*- 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;

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

3.4.4 Control

```
/*--------------------------------*- 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    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application          icoFoam;

startFrom            startTime;

startTime            0;

stopAt               endTime;

endTime              1.75;

deltaT               0.00001;

writeControl         timeStep;

writeInterval        1000;

purgeWrite           0;

writeFormat          ascii;

writePrecision       6;

writeCompression     off;

timeFormat           general;

timePrecision        6;

runTimeModifiable    true;

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

Advice:

The case reaches steady before $t = 1.75$ s. Despite it, endTime has been set to 1.75 to widely appreciate the von Kármán street and its periodical distribution. As the mesh is very refined and the solver is transitory, the simulation may be very slow

3.4.5 Discretization and linear-solver settings

```
/*--------------------------------*- 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    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss linear;
}

laplacianSchemes
{
    default         none;
    laplacian(nu,U) Gauss linear orthogonal;
    laplacian((1|A(U)),p) Gauss linear orthogonal;
}

interpolationSchemes
{
    default           linear;
    interpolate(HbyA) linear;
}

```

```
snGradSchemes
{
    default         orthogonal;
}

fluxRequired
{
    default         no;
    p               ;
}

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

```
/*--------------------------------*- 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      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

    U
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0;
    }
}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 3;
    pRefCell        0;
    pRefValue       0;
}
```

```

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

Advice:

> As it can be seen, the instruction nNonOrthogonalCorrectors has been set to 3 instead of 0 (as it was in the *plane-parallel plates* case). It is so because when checking the mesh with checkMesh, it is possible to observe that there are *mesh non-orthogonalities*. Although globally the mesh is OK to be run, it helps in obtaining more physically accurate results

At the end of the pre-processing, the structure of directories and subdirectories within cylinder195 should be as follows:

- cylinder195
  - *0*
    - *p*
    - *U*
  - *constant*
    - *polyMesh*
      - *blockMeshDict*
    - *transportProperties*
    - *RASProperties*
  - *system*
    - *controlDict*
    - *fvSchemes*
    - *fvSolutions*

3.5 Post-processing

3.5.1 Results of the simulation with $Re = 195$

The evolution over time of the velocity is as follows:

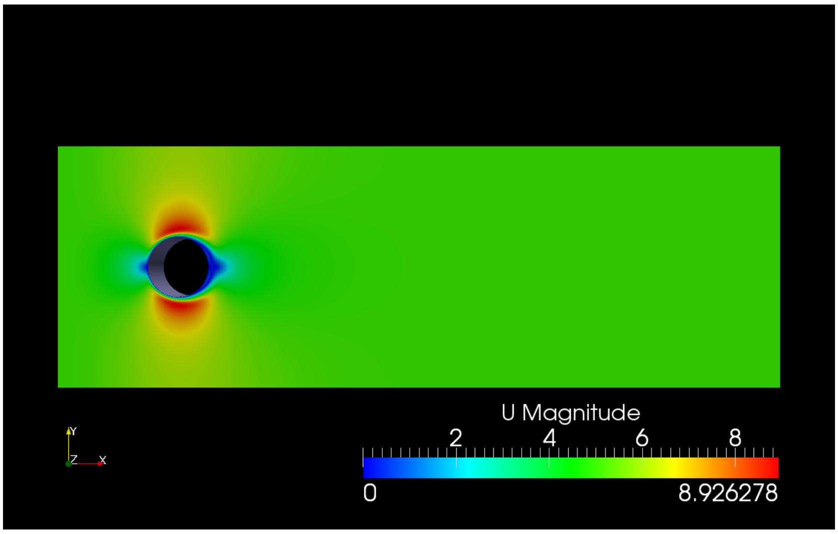


Figure 3.8: Velocity field around the bidimensional cylinder at $t = 0.01$ s (m/s)

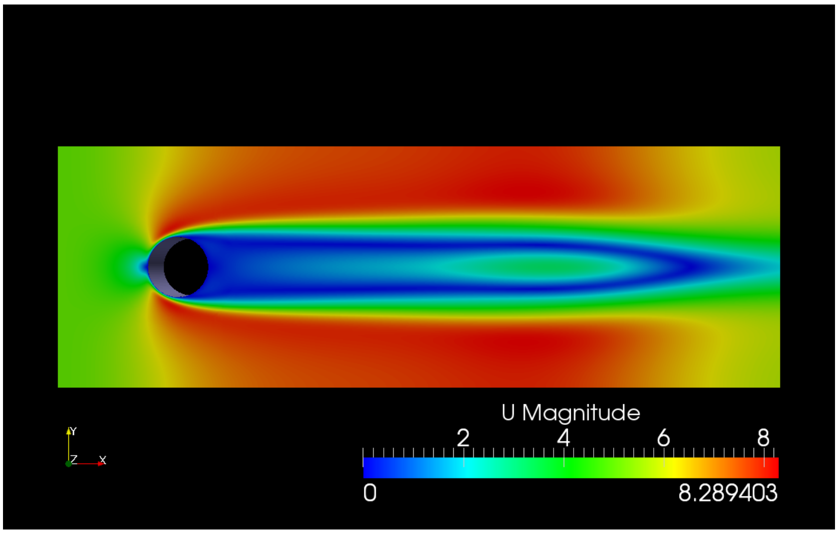


Figure 3.9: Velocity field around the bidimensional cylinder at $t = 0.6$ s (m/s)

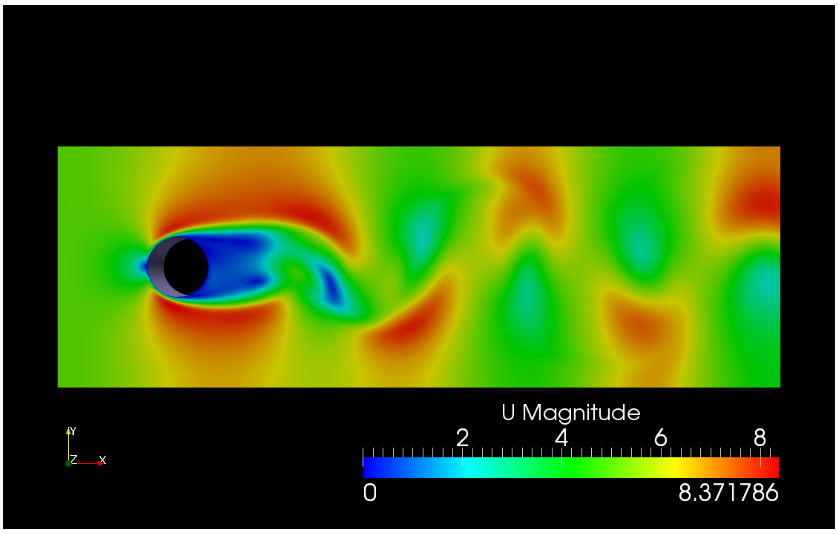


Figure 3.10: Velocity field around the bidimensional cylinder at $t = 1.13$ s (m/s)

The evolution over time of the pressure is as follows:

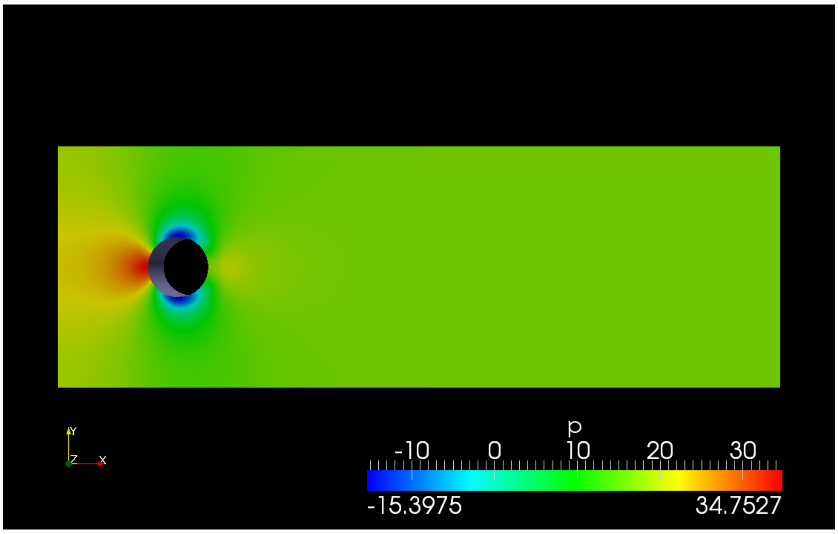


Figure 3.11: Pressure field around the bidimensional cylinder at $t = 0.01$ s (m^2/s^2)

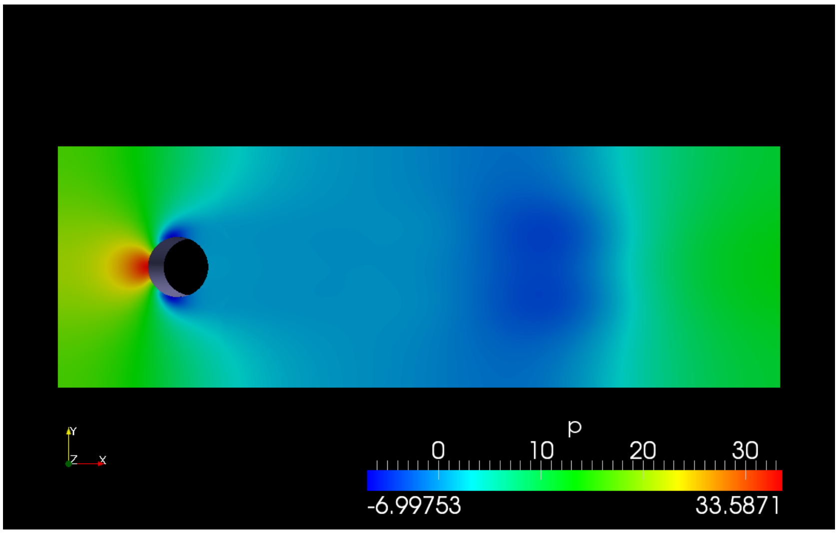


Figure 3.12: Pressure field around the bidimensional cylinder at $t = 0.6$ s (m^2/s^2)

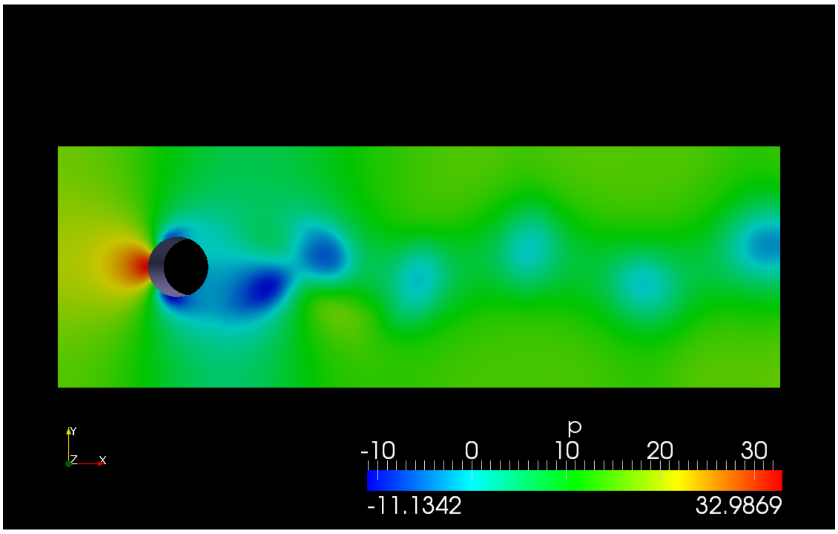


Figure 3.13: Pressure field around the bidimensional cylinder at $t = 1.13$ s (m^2/s^2)

The evolution over time of the streamlines is as follows:

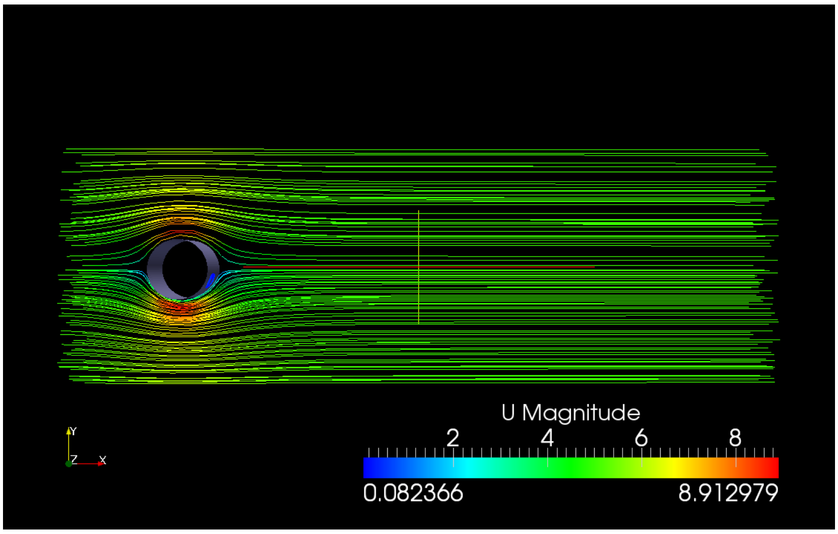


Figure 3.14: Streamlines around the bidimensional cylinder at $t = 0.01$ s (m/s)

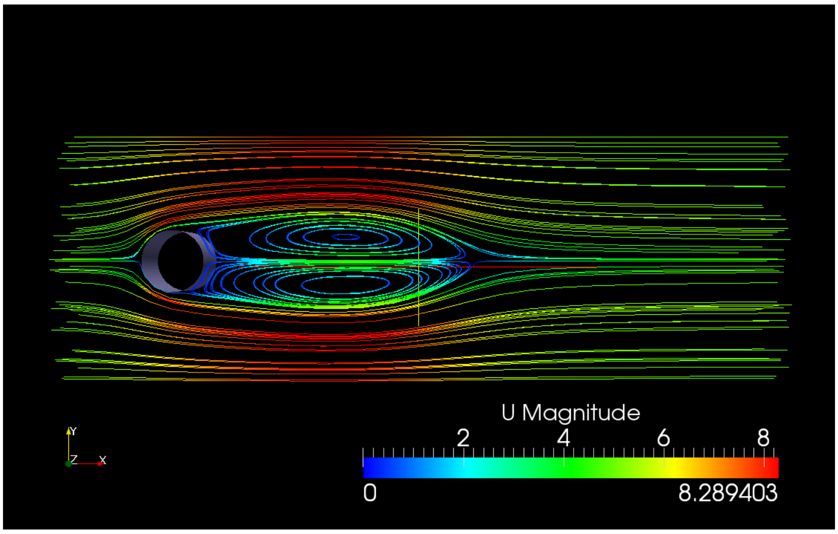


Figure 3.15: Streamlines around the bidimensional cylinder at $t = 0.4$ s (m/s)

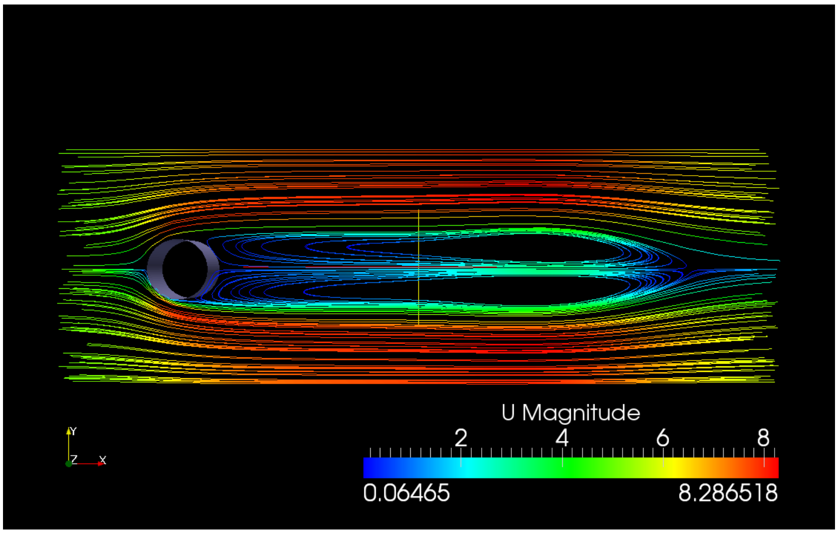


Figure 3.16: Streamlines around the bidimensional cylinder at $t = 0.6$ s (m/s)

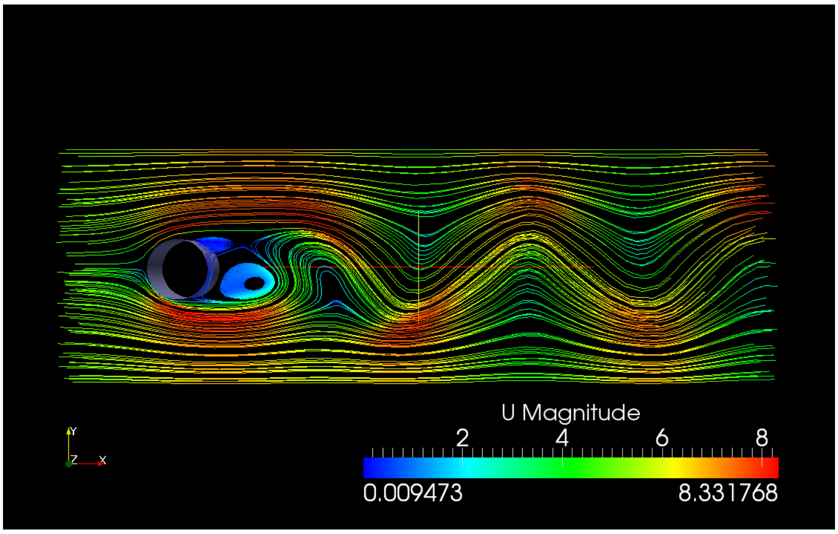


Figure 3.17: Streamlines around the bidimensional cylinder at $t = 1.13$ s (m/s)

A detail of the vector field near the wall of the cylinder is shown to appreciate the boundary layer detachment:

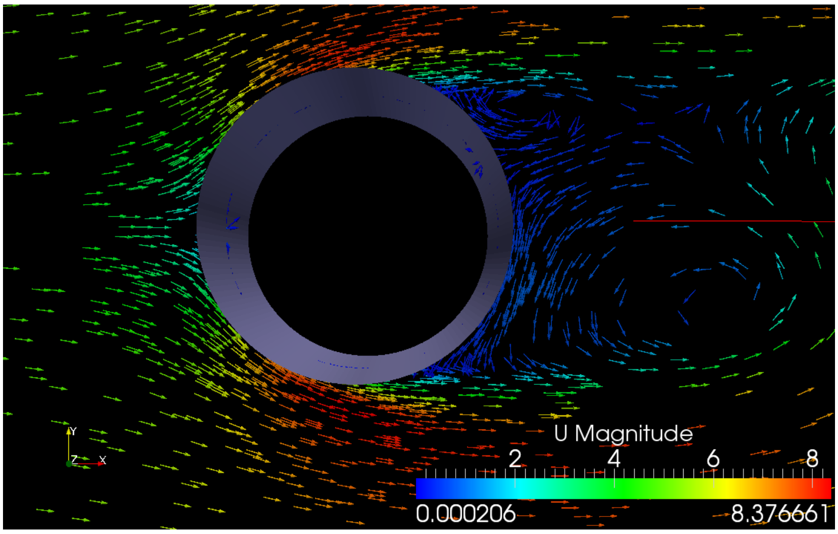


Figure 3.18: Velocity vectors around the bidimensional cylinder at $t = 1.13$ s (m/s)

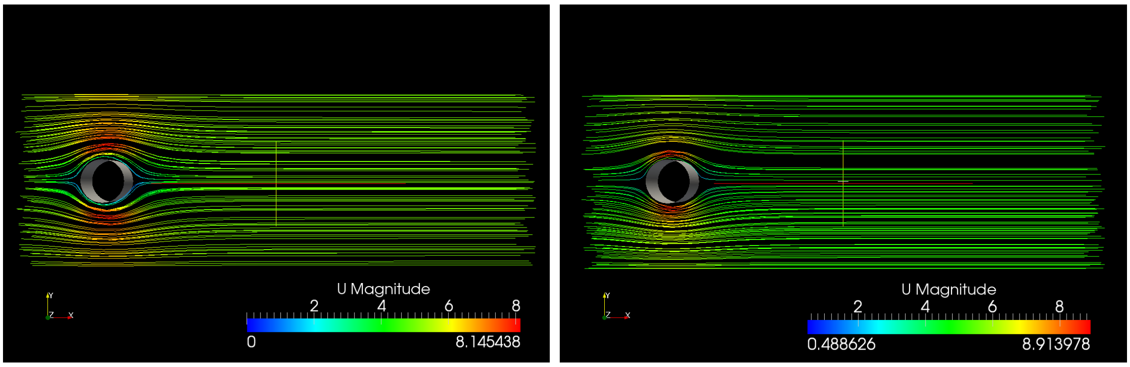
Advice:

A way to introduce the shape of the cylinder to provide more realism to the results is shown in Section 3.6.5

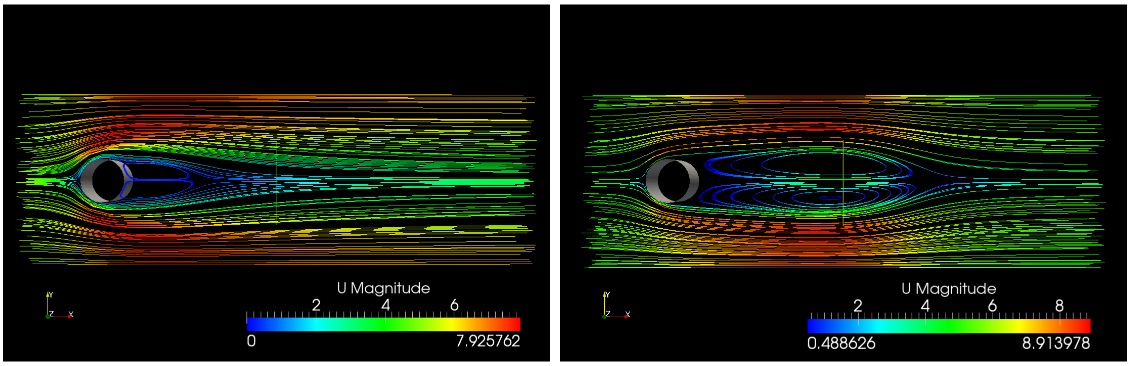
3.5.2 Comparative between cases with $Re = 30$ and $Re = 195$

Additionally, the simulation has been done for another different Reynolds number ($Re = 30$). The only instruction that needs to be changed in the *OpenFOAM*® code is for instance the kinetic viscosity in *constant/transportProperties*. According to Equation 5.3, by maintaining the value of the inlet velocity while increasing the kinetic viscosity it is possible to simulate the bidimensional cylinder in such regime (case cylinder30):

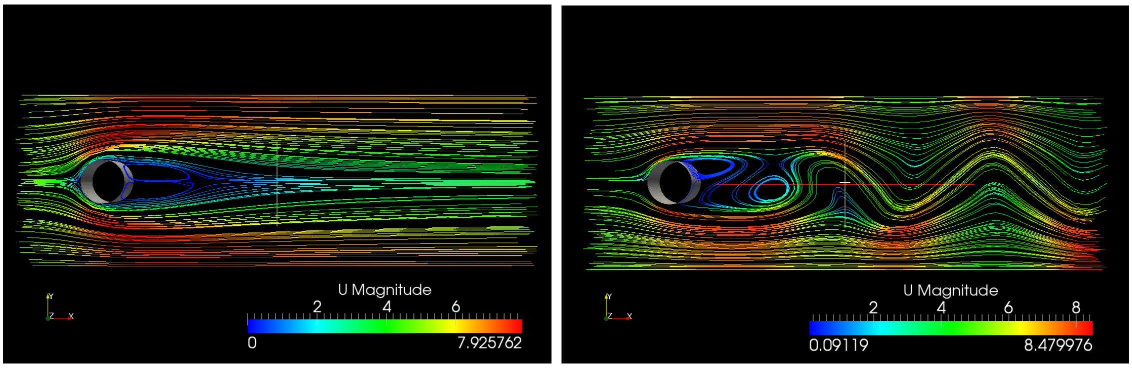
$Re = 30$ **$Re = 195$**



$t = 0.01$ s



$t = 0.4$ s



$t = 1$ s

As it was shown in Section 3.3, for $Re = 30$ the alternate vortices are not detached.

3.6 Additional utilities

3.6.1 Vorticity

As with the computation of the flow rate and the wall shear stress, it is also possible to compute the vorticity in the fluid field by using *OpenFOAM*® utilities. The vorticity is a pseudovector field that describes the local spinning motion of a fluid near some point, as would be seen by an observer located at that point and traveling along with the fluid. Mathematically, the vorticity is the curl of the velocity field:

$$\vec{w} = \nabla \times \mathbf{U}$$

The vorticity of a two-dimensional flow is always perpendicular to the plane of the flow. It plays a relevant role in the current chapter due to the existence of vortices generated by the cylinder forming the von Kármán street.

To execute it, type within the case directory

```
vorticity
```

Within the directories of each time step it has appeared a new file. Then, to observe the vorticity field, open ParaView and select it by clicking the `vorticity` box contained within Volume Fields. For the cylinder195 case, the results are:

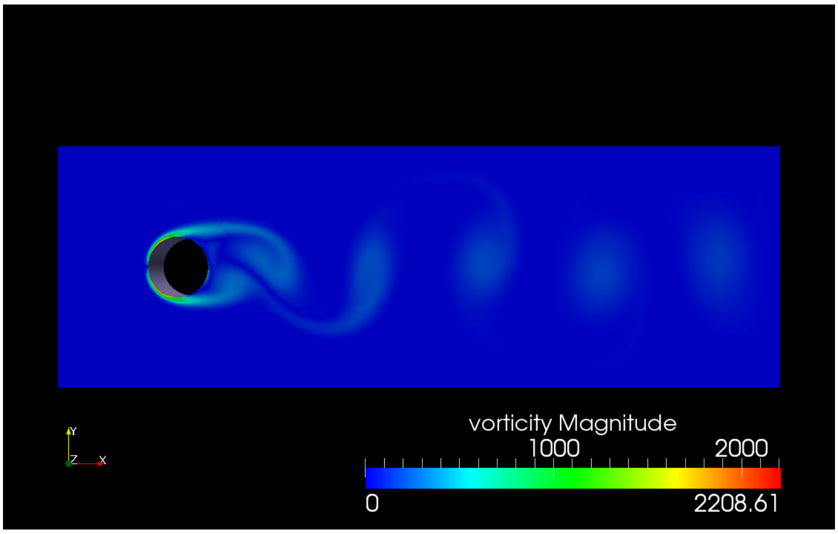


Figure 3.19: Vorticity field around the bidimensional cylinder at $t = 1.75$ s

It offers a clear-cut representation of the von Kármán vortices at $Re = 195$.

3.6.2 Computation of the aerodynamic coefficients

One of the main interests when studying external flows is the computation and analysis of the aerodynamic forces that the fluid exerts to solid objects. The drag is the force parallel to the flow velocity and the lift is the force perpendicular to the flow velocity. The dimensionless drag force is presented in Equation 5.6 and for an infinite circular cylinder its dependence on the Reynolds number is shown at Figure 3.3.

With *OpenFOAM*®, to compute the aerodynamic forces that a fluid exerts on solid walls, add the following code after the last instruction of controlDict:

```
functions
(
forces
{
type forces;
functionObjectLibs ("libforces.so");
patches (cylinder); // Patch where the force exerted by the fluid is calculated
pName p;
UName U;
rhoName rhoInf;
rhoInf 1000; // Reference density of the fluid
CofR (0 0 0); // Origin for moment calculations
outputControl timeStep; // Time criterion used to print the results
outputInterval 100; // How often (according to outputControl) the results are
    printed
}
forceCoeffs
{
type forceCoeffs;
functionObjectLibs ("libforces.so");
patches (cylinder); // Patch where the force exerted by the fluid is calculated
pName p;
UName U;
rhoName rhoInf;
rhoInf 1000; // Reference density of the fluid
CofR (0 0 0); // Origin for moment calculations
liftDir (0 1 0);
dragDir (1 0 0);
pitchAxis (0 0 1);
magUInf 5; // Free stream velocity
lRef 0.1; // Reference length (diameter of the cylinder)
Aref 0.01; // Reference area (cross sectional area of the cylinder)
outputControl timeStep; // Time criterion used to print the results
outputInterval 100; // How often (according to outputControl) the results are
    printed
}
```

```
);
```

Advice:

> For incompressible cases, the value of rhoInf is irrelevant for the computation of the dimensionless coefficients

Now, when the case is rerun, a new directory named *postProcessing* appears next to *0*, *constant* and *system*. This directory contains two subdirectories with information concerning the evolution of the aerodynamic forces and moments and their dimensionless coefficients.

3.6.3 Plotting the results with Gnuplot

Once the aerodynamic forces have been computed with the instruction shown in Section 3.6.2, it is useful to plot the results. Besides showing the behaviour of the forces with time, it allows an understanding of the convergence (or divergence) of the case. For the *bidimensional cylinder* case, it is possible to claim that if the drag coefficient converges then the case converges too.

First of all the user has to have Gnuplot installed. It is a portable command-line driven graphing utility for Linux, MS Windows, OSX and many other platforms. It is widely used to plot data obtained with *OpenFOAM*®. First, access the file containing the required data:

```
cd FoamCases/cylinder195/postProcessing/forceCoeffs/0
```

Secondly, execute Gnuplot by typing:

```
gnuplot
```

Finally plot the values of the drag coefficient (third column) in front of the time (first column) by typing:

```
plot './forceCoeffs.dat' u 1:3 w l
```

The plot is:

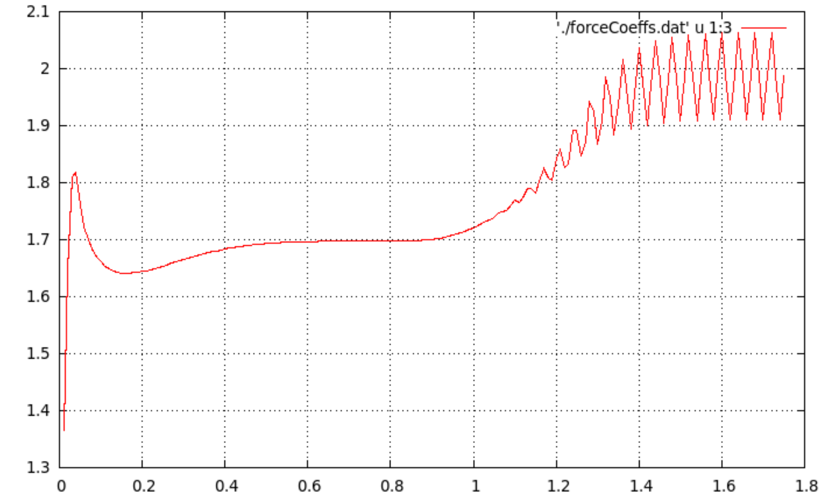


Figure 3.20: Drag coefficient (ordinate axis) of the bidimensional cylinder at $Re = 195$ in front of time (abscissa axis)

It is also possible to plot the values of the lift coefficient (fourth column) in front of the time (first column) by typing:

```
plot './forceCoeffs.dat' u 1:4 w l
```

The plot is:

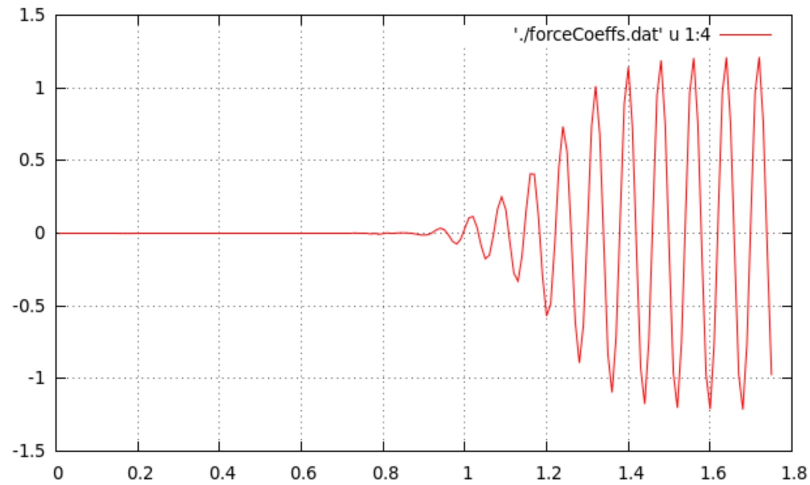


Figure 3.21: Lift coefficient (ordinate axis) of the bidimensional cylinder at $Re = 195$ in front of time (abscissa axis)

3.6.4 Computation of the stream function

As it was explained in Section 2.6.2, the streamlines offer a clear understanding of the behaviour of the flow; they represent the trajectories of particles in steady fluids. Related to it, there exists a scalar function (stream function) such that the flow velocity components can be expressed as the derivatives of this function, also being used to plot the streamlines. Mathematically it is related to the velocity as:

$$u = \frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x}$$

Since streamlines are tangent to the velocity vector of the flow, the value of the stream function must be constant along a streamline.

To obtain the stream function of the velocity, type:

```
streamFunction
```

To view the results with ParaView, it is necessary to select the `streamFunction` box located within Point Fields. Here are the results:

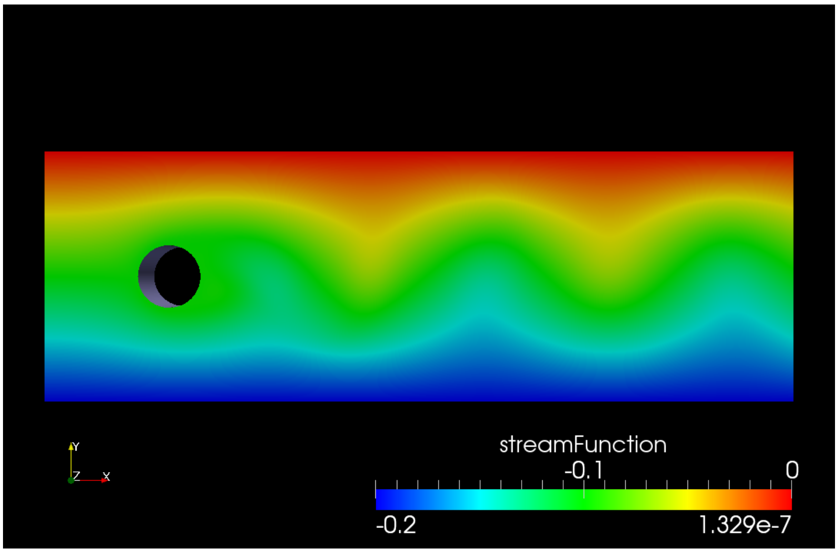


Figure 3.22: Stream function of the velocity of the *bidimensional cylinder* case for $Re = 195$ at $t = 1.75$ s

As it can be seen, it follows the same trend as the streamlines but with a continuous appearance. The colour indicates the value of the stream function in a particular point (note that this utility generates a point field), taking the $\psi = 0$ streamline as the one at the top patch of the domain.

It can be proved that the volumetric flow rate between two streamlines is equal to the difference between their stream functions. This helps in the validation of the streamFunction utility. At the inlet:

$$Q = V \cdot S = 5 \cdot 0.4 \cdot 0.1 = 0.2\ m^3/s$$

$$Q = \psi_n - \psi_0 = 0.2\ m^3/s$$

3.6.5 Convertion to VTK

It is possible to convert data from *OpenFOAM*® to VTK format. For instance, access cylinder195 and type:

```
foamToVTK
```

A new directory appears within the case containing the VTK data. Since it is a worldwide used format, it is also possible to open it with ParaView.

Example: Once in ParaView, click on the "open" icon, access the VTK directory of the case and click on "cylinder". Within the Pipeline Browser, a new module has appeared. It is the cylinder patch whose shape can be used to give more realism to the results of the simulation. This same procedure can be carried out for each one of the defined patches of the case.