Computational Fluid Dynamics
Assignment 3

Søren Asp Nissen, AU683660
Anton Østergaard Thorsen, AU681040
Claes Heide Pinnerup, AU643777

May 19, 2024

AARHUS UNIVERSITY

TABLE OF CONTENTS

Page

1 | Task 1.1: Controlling the mesh

To begin the simulation of a flow over a cylinder, the first task was to create mesh using the supplied `blockMeshDict` file. Fig. 1 is a visualization of this mesh:

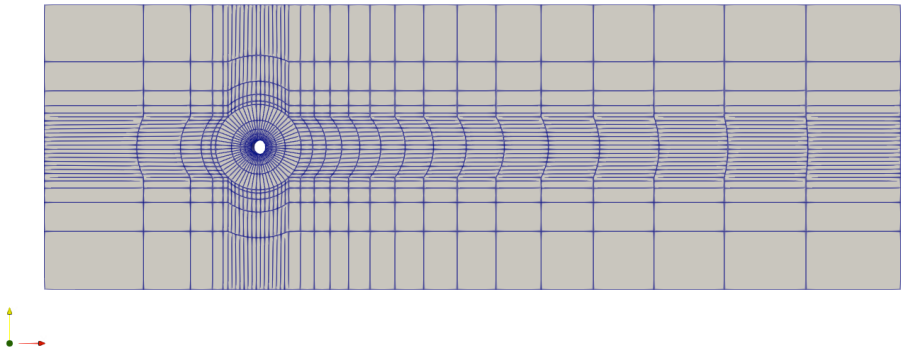

Figure 1: Visualization of mesh using Paraview

This mesh could be refined, by editing the section of `blockMeshDict` which controls the number of subdivisions of each section of the mesh:

```
blocks
(

    hex (4 5 1 0 24 25 21 20) (5 5 1)       simpleGrading (0.05 15 1)
    hex (5 6 2 1 25 26 22 21) (15 5 1)      simpleGrading (1 15 1)
    hex (6 7 3 2 26 27 23 22) (15 5 1)      simpleGrading (8 15 1)
    hex (12 13 5 4 32 33 25 24) (5 15 1)    simpleGrading (0.05 1 1)

    hex (13 10 8 5 33 30 28 25) (10 15 1)   simpleGrading (0.005 1 1)
    hex (8 9 6 5 28 29 26 25) (15 10 1)     simpleGrading (1 200 1)
    hex (11 14 6 9 31 34 26 29) (10 15 1)   simpleGrading (200 1 1)
    hex (13 14 11 10 33 34 31 30) (15 10 1) simpleGrading (1 0.005 1)

    hex (14 15 7 6 34 35 27 26) (15 15 1)   simpleGrading (8 1 1)
    hex (16 17 13 12 36 37 33 32) (5 5 1)   simpleGrading (0.05 ...
        0.06 1)
    hex (18 19 15 14 38 39 35 34) (15 5 1)  simpleGrading (8 0.06 1)
    hex (17 18 14 13 37 38 34 33) (15 5 1)  simpleGrading (1 0.06 1)

);
```

Here, the first parenthesis represents the vertex indices defining each mesh block. The second parenthesis defines the number of subdivisions along each axis. For the first row, for example, `(5 5 1)` results in the block being divided into 5 segments along the x-axis, 5 segments along the y-axis, and only one segment along the z-axis. Thus, a larger number of subdivisions can be used to refine the mesh. The final parenthesis defines how the cells are graded along each block, and thus how the cells are expanded or compressed along the different axes. This, however, will not be changed in this

project.

An example of a refined mesh is shown in Fig. 2:

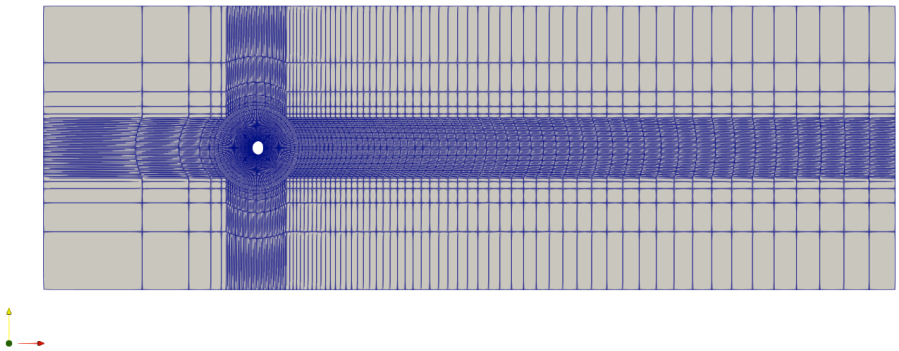

Figure 2: Visualization of refined mesh with 12100 cells

2 | Task 1.2: Setup of simulation

To simulate the flow past the cylinder, with Re = 100, the kinematic viscosity or the flow velocity can be adjusted such that:

$$\mathrm{Re} = \frac{UL}{\nu} = 100 \tag{1}$$

Where L is the characteristic length, which in this case is the cylinder diameter, U is the flow velocity and ν is the kinematic viscosity. The Reynolds number can now be adjusted in OpenFOAM, by adjusting the appropriate settings. With a cylinder diameter of $D = 1\,\mathrm{m}$, and an inlet velocity of $1\,\mathrm{m/s}$, the kinematic viscosity is chosen to be $\nu = 0.01$. The flow velocity is chosen in the `0` folder, where velocity initial conditions are stored, in the file `U`. For the inlet, the velocity is set to be `(1 0 0)`, i.e. $1\,\mathrm{m/s}$ in the x-direction. Simoultaneously, no slip conditions are set at the walls of the cylinder at `walls`, zero-gradient at the `outlet` and slip on the `sides` of the domain. All of this is specified with the code:

```
internalField uniform (0 0 0);

boundaryField
{
    walls
    {
        type noSlip;
    }
    inlet
    {
        type fixedValue;
        value uniform (1 0 0);
    }
    outlet
```

```
15      {
16          type zeroGradient;
17      }
18      sides
19      {
20          type slip;
21      }
22  }
```

In the folder `constant`, the kinematic viscosity is adjusted in the file `transportProperties`, with the code:

```
1  transportModel Newtonian;
2
3  nu nu [ 0 2 -1 0 0 0 0 ] 0.01;
```

Where `transportModel Newtonian` specifies that the fluid in the simulation should follow a Newtonian model of viscosity. the final number `0.01` specifies the value of the kinematic viscosity, in SI-units, while the bracket `nu [0 2 -1 0 0 0 0]` defines the unit from the fundamental dimensions `[mass, length, time, temperature, electric current, amount of substance, luminous intensity]`, such that the unit becomes $[\text{length}]^2[\text{time}]^{-1}$ i.e. m^2/s. As such the correct Reynolds number is achieved.

The pressure initial and boundary conditions can be specified in the file `p` in a similar way to `U`.

```
1  dimensions [0 2 -2 0 0 0 0];
2
3  internalField uniform 0;
4
5  boundaryField
6  {
7      walls
8      {
9          type zeroGradient;
10     }
11     inlet
12     {
13         type zeroGradient;
14     }
15     outlet
16     {
17         type fixedValue;
18         value uniform 0;
19     }
20     sides
21     {
22         type zeroGradient;
23     }
24 }
```

For the `controlDict` file, which controls simulation parameters, output settings, etc., the following setup is used:

```
application icoFoam;

startTime 0;

endTime 200;

deltaT 0.02;

writeControl adjustableRunTime;

writeInterval 1;

writeFormat ascii;

writePrecision 6;

writeCompression off;

timeFormat general;

timePrecision 6;

runTimeModifiable yes;

adjustTimeStep yes;

maxCo 0.5;

libs( "libforces.so" "libOpenFOAM.so"
"libfieldFunctionObjects.so");

functions
{
    #includeFunc solverInfo;

    readFields
    {
        functionObjectLibs( "libfieldFunctionObjects.so" );
        type readFields;
        fields (p U);
    }

    forces
    {
        type forceCoeffs ;
        functionObjectLibs( "libforces.so" );
        outputControl outputTime ;
        writeInterval 0.01;
        patches ( "walls" );
        pName p;
        UName U;
        rho rhoInf ;
```

```
53          log true ;
54          rhoInf 1;
55          liftDir (0 1 0);
56          dragDir (1 0 0);
57          CofR (0 0 0);
58          pitchAxis (0 1 0);
59          magUInf 1.0;
60          lRef 1.0;
61          Aref 1.0;
62      }
63  }
```

The important parts of the code are explained below:

- `application icoFoam;` defines which solver is used for solving the problem. In this case *icoFoam* is used, which is a solver for incompressible laminar flows that, unlike simpleFoam, is used for unsteady flows. With a Reynolds number of Re = 100, which is well inside the laminar regime, this solver should be adequate.
- `startTime 0;` specifies the startTime to be 0 s
- `endTime 200;` specifies the endtime to be 200 s
- `deltaT 0.02;` defines the timestep to be 0.02 s. This can be adjusted when checking for timestep independence
- `adjustTimeStep yes` allows OpenFoam to automatically adjust the timestep, such that the Courant number is below 0.5, defined at `maxCo 0.5`

in the functions that evaluate the lift and drag force, in the lower part of controlDict, the following setup is used:

- `patches ( "walls" )` specifies the patches where we want C_d and C_l to be evaluated, in this case at the cylinder walls.
- `rho rhoInf` sets the fluid density, which is later specified to be 1 at `rhoInf 1`
- `liftDir (0 1 0)` and `dragDir (1 0 0)` specify that the lift should be evaluated along the y-axis, and the drag along the x-axis

The used `fvSchemes` setup is as follows:

```
1  ddtSchemes
2  {
3      default backward;
4  }
5
6  gradSchemes
7  {
8      default none;
9      grad(p) Gauss linear;
10     grad(U) Gauss linear;
11     snGradCorr(U) Gauss linear;
12     snGradCorr(p) Gauss linear;
13 }
```

```
14
15 divSchemes
16 {
17     default none;
18     div(phi,U) Gauss linear;
19 }
20
21 laplacianSchemes
22 {
23     default none;
24     laplacian(nu,U) Gauss linear corrected;
25     laplacian((1|A(U)),p) Gauss linear corrected;
26 }
27
28 interpolationSchemes
29 {
30     default linear;
31     interpolate(U) linear;
32 }
33
34 snGradSchemes
35 {
36     default corrected;
37 }
38
39 fluxRequired
40 {
41     default no;
42     p ;
43 }
```

The important parts of the code are explained below:

- `default backwards` under `ddtSchemes`, specifies that a 'backward' scheme is used for time derivatives
- under `gradSchemes`, no scheme is chosen as default in `default none`, but the Gauss linear scheme (which is 2nd order) is used for both the gradient in the pressure field, the velocity field, and for the surface normal gradient corrections in `grad(p)`, `grad(U)` `snGradCorr(U)` and `snGradCorr(p)`
- `divSchemes`: similarly for the divergence, there is no default scheme, but Gauss linear scheme is used for divergence in flux (`phi`) and velocity (`U`)

The following setup is used for the `fvSolution` file, which specifies solver settings:

```
1 solvers
2 {
3     p
4     {
5         solver PCG;
6         tolerance 1e-9;
7         relTol 0.01;
8         preconditioner DIC;
9         cacheAgglomeration true;
```

```
10          nCellsInCoarsestLevel 10;
11          agglomerator faceAreaPair;
12          mergeLevels 1;
13      }
14      pFinal
15      {
16          $p;
17          relTol 0;
18      }
19      U
20      {
21          solver smoothSolver;
22          smoother symGaussSeidel;
23          tolerance 1e-05;
24          relTol 0.1;
25      }
26      UFinal
27      {
28          $U;
29          relTol 0;
30      }
31  }
32
33  PISO
34  {
35      nNonOrthogonalCcorrectors 0;
36      nCorrectors 2;
37  }
```

Explanation of the setup:

- under `p`, the solver is chosen to be preconditioned conjugate gradient in `solver PCG;`. The tolerance is set to 10^{-9}, and the solver stops if the change in pressure is below this value.
- the solver used for the velocity field `U`, is chosen to be `smoothSolver`, and it uses a Gauss-Seidel smoother. The tolerance is here set to `1e-05`.
- under `PISO`, the number of correctors, i.e. the number of times the pressure is corrected within a single time step, is set to 2.

3 | Task 1.3: Lift and drag

After running the simulation, data for the lift and drag coefficients is extracted and plotted. Fig. 3 shows the time series of the lift coefficient, where it is seen to fluctuate quite heavily, after approximately 60 seconds. This is due to the development of vortex shedding. In Fig. 4, the time series for the drag coefficient is shown. In the latest part of the time series, the drag coefficient is also seen to fluctuate slightly around $C_d = 1.4$, but these fluctuations are not nearly as pronounced as for the lift coefficient. Both of these plotted time series, are for the mesh-independent and time step-independent setup, which will be further discussed in Section 4.

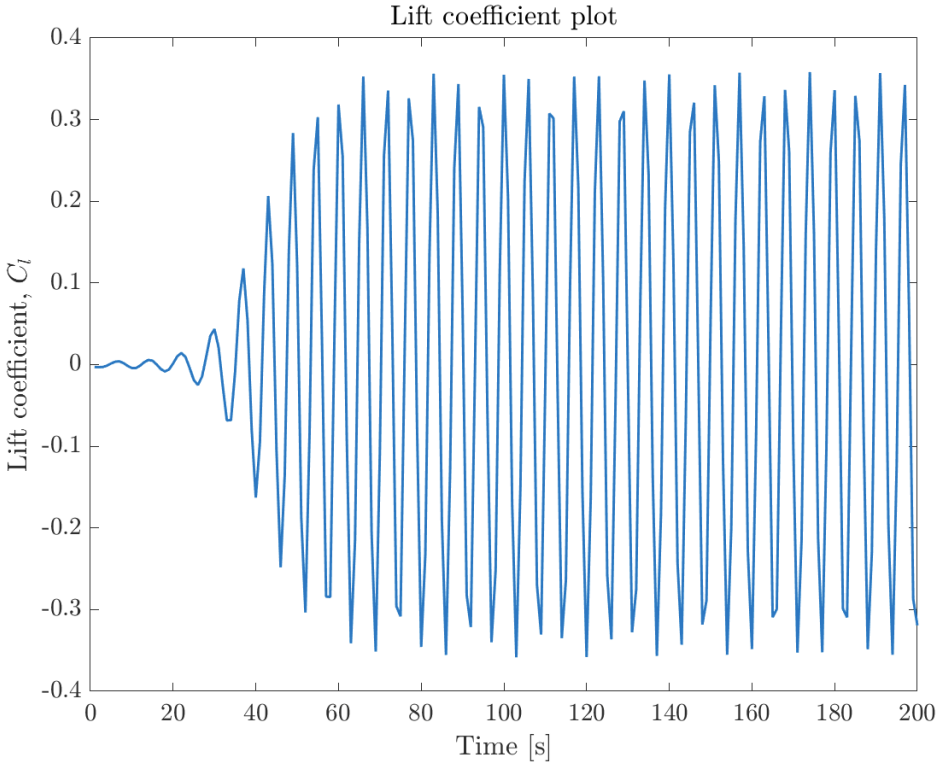

Figure 3: Lift coefficient plot

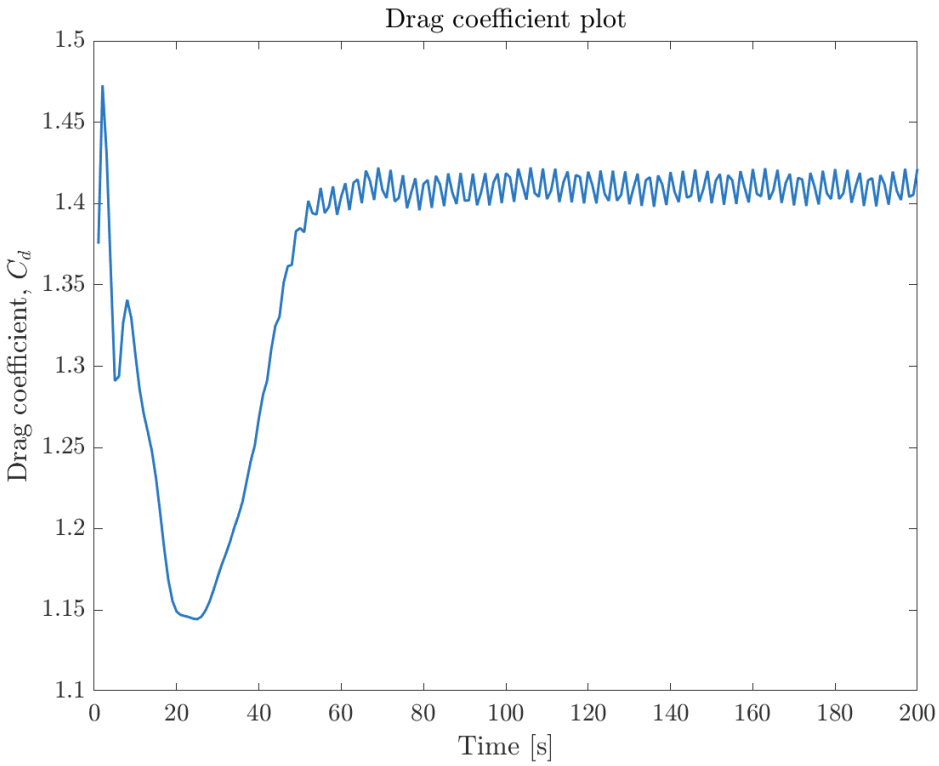

Figure 4: Drag coefficient plot

In addition to the lift and draft plot, the residuals were also extracted, and plotted in

Fig. 5:

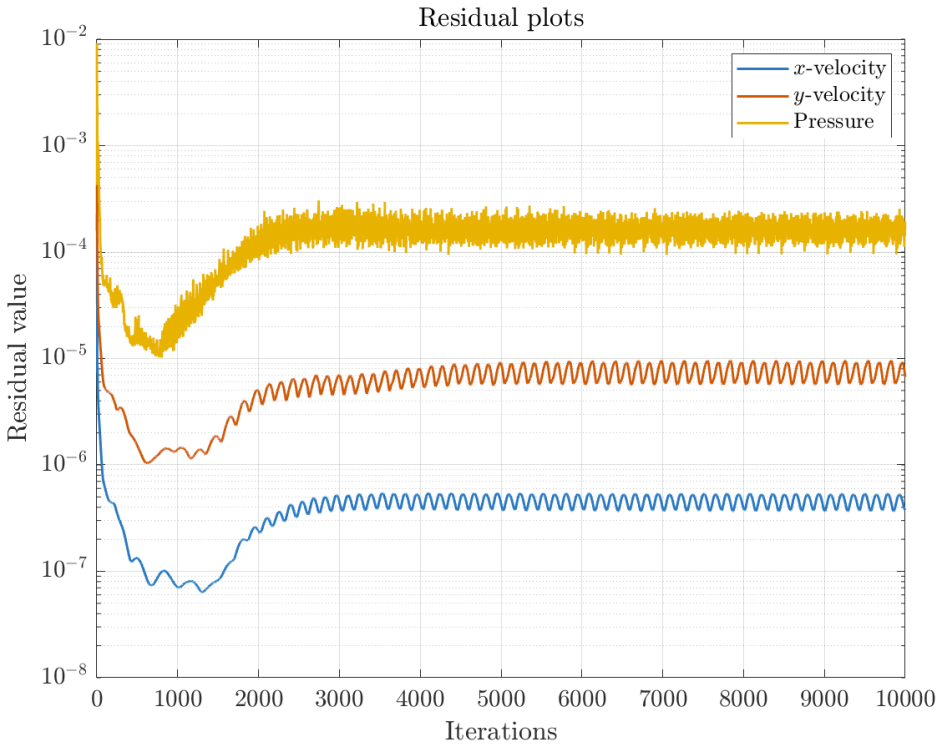

Figure 5: Residual plot

Also, the residuals show that the simulation has been succesful. For pressure, the residuals should be below $\sim 10^{-3}$, while the residuals for the velocities should be below $\sim 10^{-5}$.

4 | Task 1.4: Independency

When setting up the simulation, it was also necessary to check for mesh independence. This was done by refining the mesh, and re-running the simulation while checking the change in the maximum lift coefficient and maximum drag coefficient. In the refinement process, there was added emphasis on refining the mesh in the area of vortex shedding.

Firstly the mesh dependency analysis:

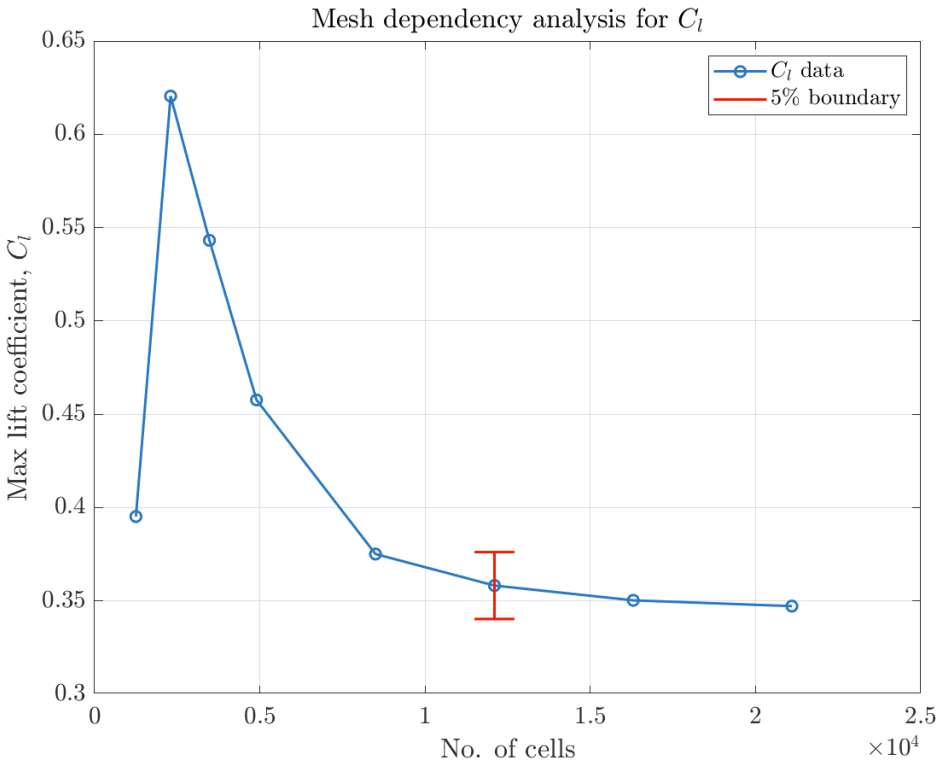

Figure 6: Mesh dependency for lift coefficient for ΔT=0.02 s and nCorr=2

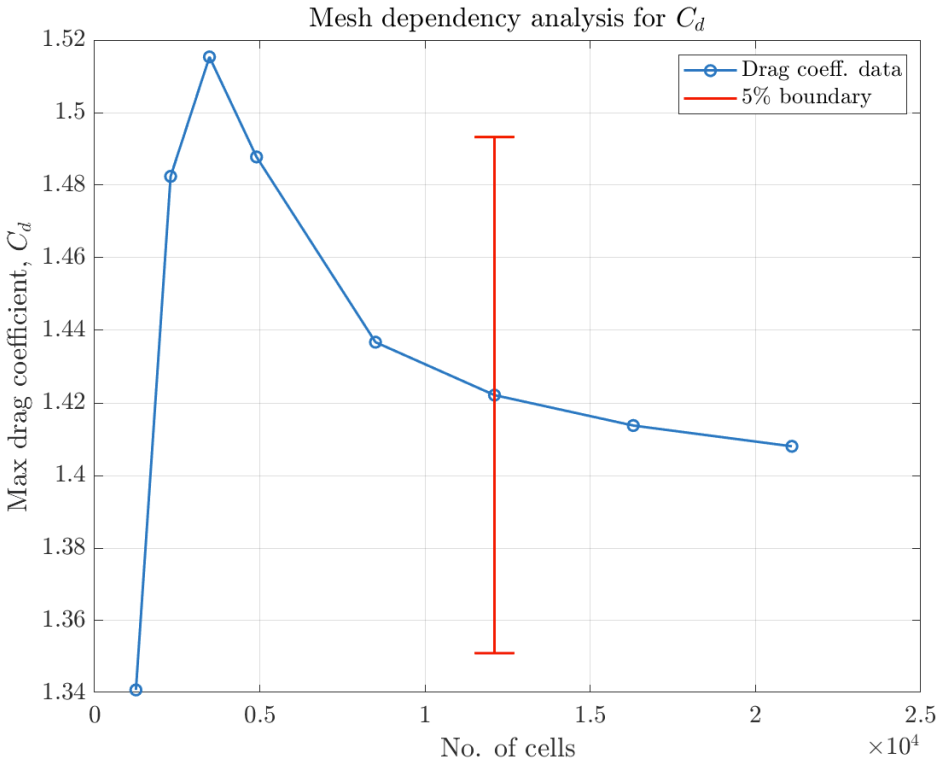

Figure 7: Mesh dependency for drag coefficient for ΔT=0.02 s and nCorr=2

From Figs. 6 and 7, it can be seen that approximately doubling the number of cells from 12100 to 21100 changes the values of C_l and C_d less than 5%, thus in this case the

mesh with 12100 cells is deemed sufficiently accurate. As such increasing the number of cells would mainly just result in unnecessary large computational times. Note that the mesh with 12100 cells is depicted in Fig. 2, which is a nice mesh as it has a max skewness of just 1.1 and a max non-orthogonality of just 43.9.

Next time-step dependency was analyzed, in a similar fashion:

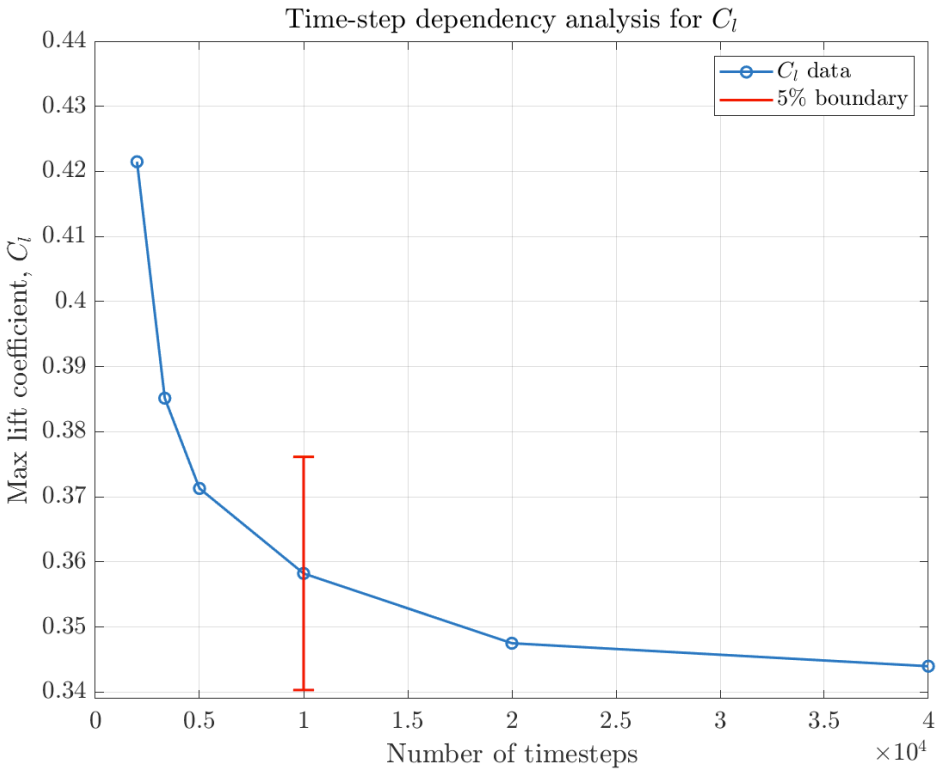

Figure 8: Time step dependency for lift coefficient for 12100 cells and nCorr=2

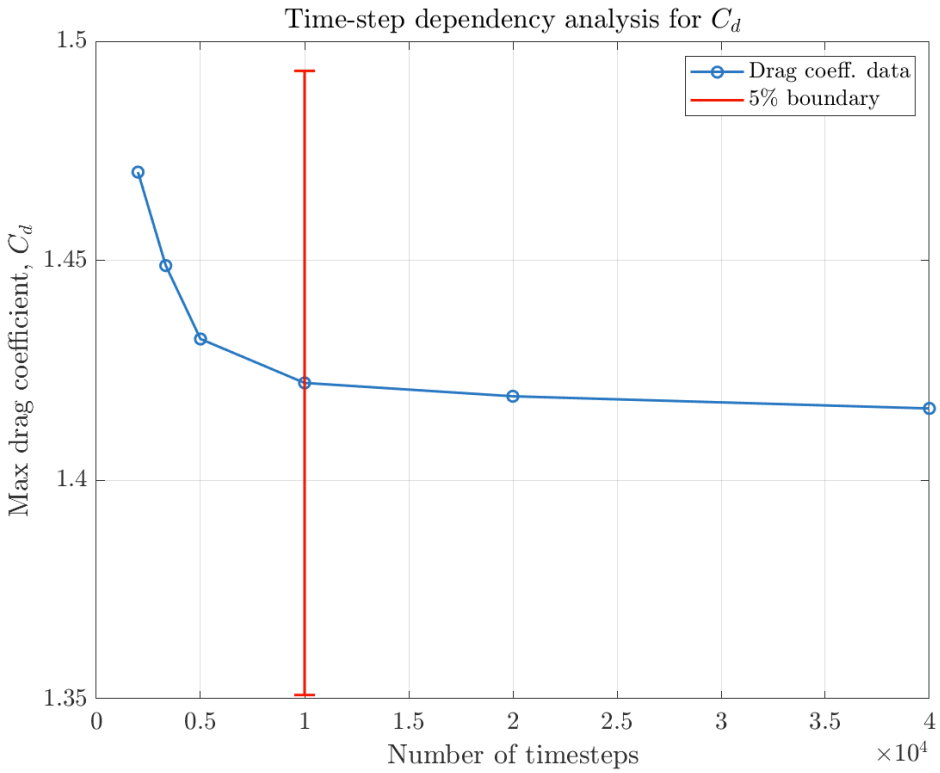

Figure 9: Time step dependency for drag coefficient for 12100 cells and nCorr=2

Similarly to the mesh dependency analysis, when doubling the number of time steps from 10000 to 20000 (and even 40000) the drag and lift coefficients change less than 5%, and thus increasing the number of time steps beyond 10000 is deemed unnecessary. Note that 10000 timesteps correspond to $\Delta T = 0.02\,s$.

Additionally, the necessary number of correctors, which are the number of inner loops in the PISO algorithm, has been investigated as well:

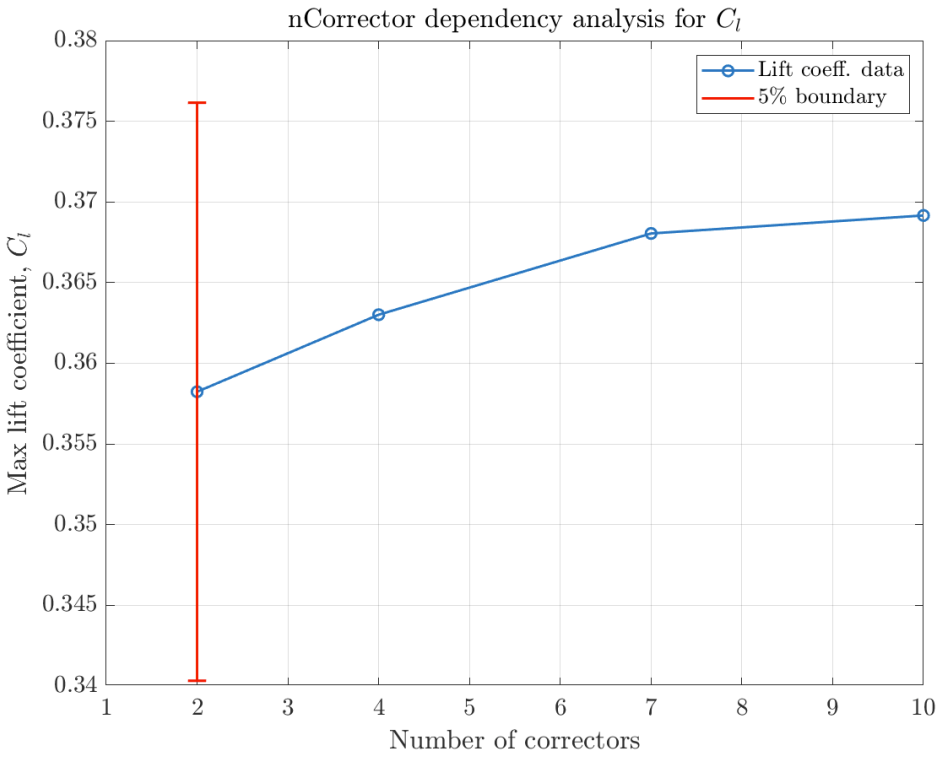

Figure 10: Correcctor dependency for lift coefficient for ΔT=0.02 s and 12100 cells

From Fig. 10 it is found that 2 corrector steps produce sufficiently accurate results.

The max lift coefficient with 2 number of correctors, with step size $\Delta T = 0.02\,s$ and a mesh consisting of 12100 cells is $C_l = 0.358$, and the max drag coefficient is $C_d = 1.422$

5 | Task 1.5: Strouhal number

An oscillating flow past a body can be characterized by the dimensionless Strouhal number St. A larger St indicates a higher frequency relative to the flow velocity while, conversely, a smaller value indicates a lower relative frequency. One way to calculate the Strouhal number is provided in Eq. 2.

$$St = \frac{f \cdot L}{U} \tag{2}$$

In Eq. 2 f is the oscillation frequency, L is the characteristic length, and U is the flow velocity. As mentioned in Section 2 the characteristic length for our case is the cylinder diameter, $D = 1\,m$, and the flow velocity is $U = 1\,m/s$.

The oscillation frequency can be derived from the flow simulation results by use of a Fourier Transform. In this case a Fast Fourier Transform, or FFT, method is used to derive the dominant frequency of the flow oscillations.

The oscillation frequency is $f = 0.1881\,Hz$. With an inlet flow velocity of $U = 1\,m/s$ and a characteristic length/cylinder diameter of $L = 1\,m$, the Strouhal number is then:

$$St = \frac{0.1881\,\mathrm{Hz} \cdot 1\,\mathrm{m}}{1\,\mathrm{m/s}} = 0.1881 \tag{3}$$

The Strouhal number can also be calculated from empirical relations between the Strouhal number and Reynolds number. One such relation is:

$$St(Re) = 0.2684 - \frac{1.0356}{\sqrt{\mathrm{Re}}} \tag{4}$$

Thus, for a Reynolds number of 100, we get the approximated Strouhal number using Eq. 4:

$$St(Re = 100) = 0.2684 - \frac{1.0356}{\sqrt{100}} = 0.18284 \tag{5}$$

This has a deviation from the strouhal number determined from the oscillation frequency obtained from the CFD simulations, using Eq. 2, of just 2.7 %. Thus, this relation seems to fit the obtained results.

6 | Task 1.6: Validation

In this section, the obtained Strouhal number and drag and lift coefficients are validated, by comparison to the literature. In Table 1 drag and lift coefficient results from 5 separate sources are displayed along with the results obtained in this paper.

Results at Re = 100	C_d	C_l
D. Russel and Z. Wang [6]	1.38 ± 0.007	± 0.322
D. Calhoun and Z. Wang [2]	1.35 ± 0.014	± 0.30
M. Braza, P. Chassaing, and H. Hinh [1]	1.36 ± 0.015	± 0.25
J. Choi, R. Oberoi, J. Edwards, and J. Rosati [3]	1.34 ± 0.011	± 0.315
J. Guerrero [5]	1.39 ± 0.012	± 0.333
Our Results	1.422	± 0.358

Table 1: Comparison of drag and lift coefficients, i.e. C_d and C_l, results with literature

As is evident from Table 1 the obtained drag coefficient, C_d, is a little higher compared to the literature results, while the magnitude of C_d fluctuations is very similar. The lift coefficient, C_l, is also somewhat higher than in the literature. Both coefficients, however, seem satisfactory. As described in Section 4, this small error might only partly stem from mesh resolution or the number of time steps, as the results converge above those results. Instead, these small systematic errors might be due to small variations in the given flow parameters. In the literature Durbin et al. (2007) [4], obtains $St = 0.17$ at Re = 100 with $C_l = 0.35$. This Strouhal number is similar but smaller than both the results $St = 0.1881$ and $St = 0.18284$, obtained using Eq. 2

and Eq. 4 respectively. The Strouhal number obtained using Reynolds should be the same in both cases as the Reynolds are the same. The *St*-result obtained using Eq. 2 in this paper is, however closer to that result. The difference might, like with the coefficients, indicate some small discrepancy in parameters.

7 | Task 1.7: Velocity field and streamlines

In Fig. 11 the velocity field contours in the x-direction are depicted, for the validated case with 12100 cells, 2 correctors, and ΔT=0.02 s, showing the vortex-shedding phenomenon in the wake of the cylinder.

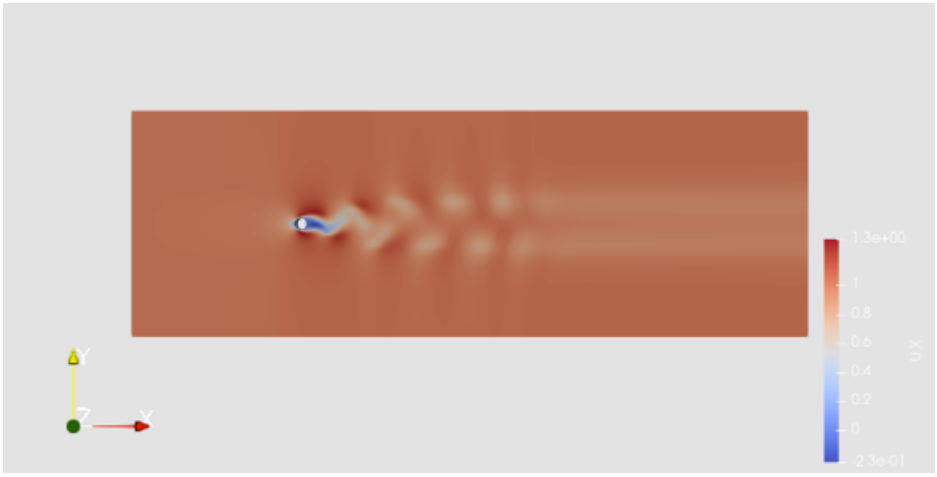

Figure 11: Velocity field contours in x-direction

This phenomenon can be further visualized from the streamlines around the cylinder, as depicted in Fig. 12:

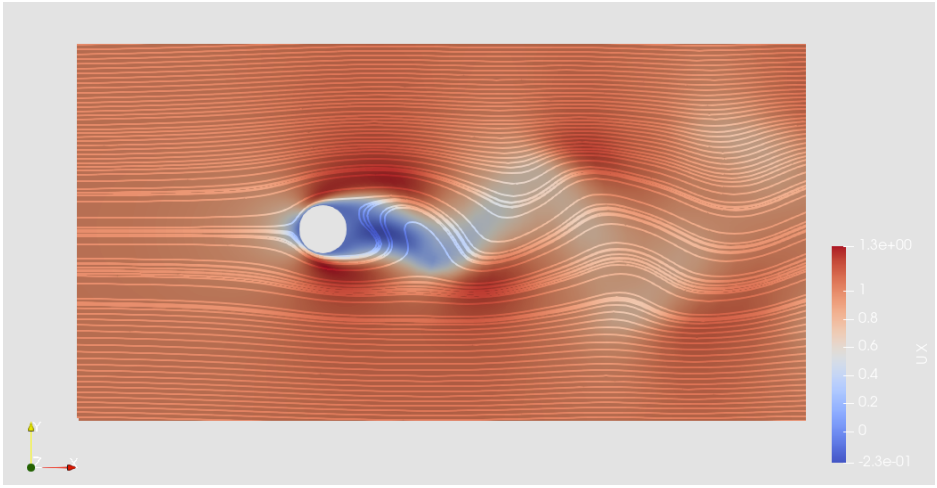

Figure 12: Streamlines around cylinder

Bibliography

[1] M Braza, PHHM Chassaing, and H Ha Minh. "Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder". In: Journal of fluid mechanics 165 (1986), pp. 79–130.

[2] Donna Calhoun. "A Cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions". In: Journal of computational physics 176.2 (2002), pp. 231–275.

[3] Jung-Il Choi et al. "An immersed boundary method for complex incompressible flows". In: Journal of Computational Physics 224.2 (2007), pp. 757–784.

[4] Paul A. Durbin and Gorazd Medic. "Intermediate Reynolds Numbers". In: Fluid Dynamics with a Computational Perspective. Cambridge University Press, 2007, pp. 132–166.

[5] Joel Guerrero. "Numerical simulation of the unsteady aerodynamics of flapping flight". In: Department of Civil, Environmental, Architectural Engineering (2009).

[6] David Russell and Z Jane Wang. "A Cartesian grid method for modeling multiple moving objects in 2D incompressible viscous flow". In: Journal of Computational Physics 191.1 (2003), pp. 177–205.