CFD in OpenFOAM

An entry-level project, based on a case study of a backward-facing step. This project is for those interested in learning how to automate OpenFOAM simulations using `Bash`.

Matej Tomić - *MSc* in ME
`matej.tomic@gmail.com`

Contents

List of Figures

List of Tables

1 Introduction

This project will guide the reader on how to write small, but powerful, `Bash` scripts in order to automate the workflow of CFD simulations. The CFD case in the project is relatively-speaking simple, so the `Bash` scripts generated here can be a good foundation for more complex cases. A backward-facing step will be the computational domain, and the results obtained will be compared to a scientific paper making it possible to validate numerical results obtained with an experiment.

1.1 Motivation

Since I made a switch to OpenFOAM, I noticed it inherently required a smarter approach for performing CFD simulations. For those like me that became acquainted with OpenFOAM only after using ”conventional” software packages with a GUI, it seemed weird to a only have collection of dictionaries that are manipulated by a text editor and a CLI. Thankfully, we live in the day and age where internet gives us copious amounts of information[1]. So, learning OpenFOAM should now be easier than before. I decided to give the OpenFOAM tutorials a chance and began descending into the rabbit hole. Soon enough, I noticed that a lot of my time was spent preparing cases, running them, exporting files, etc., which was clearly hindering the progress since I wasn't focusing so much on actual CFD. A steep learning curve combined with the fact that I was working in an unfamiliar environment of Linux, made me realize I was spending lots of time reading threads on forums concerning topics unrelated to CFD. In reality, I was subconsciously overwhelmed by the fact that my initial plan of engaging in numerical simulations only, will have to be further slowed down due to the lack of knowledge of Linux, Unix, Vim and others. This hit hard, because now my time resources needed to be further allocated for learning additional unfamiliar concepts. As if there weren't plenty already!

1.2 Goals

The goal of this project is ultimately to go through the crucial steps that enable the user to automate tedious and repetitive tasks, while setting up and simulating a 2D backward-facing step simulation in OpenFOAM. The automation will greatly speed up and help us:

1. **perform a mesh-independence study**
2. and **investigate different turbulence models**.

The backward-facing step induced phenomena in a flow field is a classic study case and is chosen because of its relative ”simplicity” that relieves us of dealing with more complicated flow phenomena and geometries. Such a project will enable users to thoroughly investigate important niches of setting up CFD simulations in OpenFOAM, and automating a big part of the process so that more time can be devoted to, for example post-processing and qualitative assessment of the key flow features. The driving idea is for the reader to gain a robust overview of what the automation process entails, so that the concepts can be scaled-up and transferred to any case of choice.

[1] A possible downside being that one might find it hard to focus on a single topic.

1.3 Covered topics

It is important to clearly state the topics that will be covered in this project, so that the reader can decide if this project is of any significance for them. In essence, in order to perform CFD simulations the user usually has to go through pre-processing, solving the case, and post-processing. This is visually shown in figure 1.1, where the circular arrow inside the triangle symbolises an inherently iterative process.

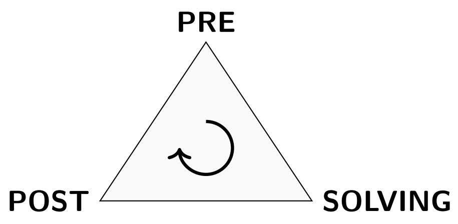


Figure 1.1: A typical CFD workflow.

If one were to monitor the time is spent on each of these steps, the biggest amount would go to the pre-processing. Pre-processing usually comprises the following tasks:

1. **modeling the geometry in CAD**,
2. **cleaning up the dirty geometry**[2],
3. **writing scripts to automate repetitive tasks**,
4. **meshing and mesh-independence studies**
5. **choosing solver parameters**.

Setting up a case properly will take the most amount of time. In the long run it has proven very well worth investing such time, so that the rest of the job can be performed with relative ease. Since the project geometry is relatively simple in our case, there will be no need of cleaning up the geometry and simplifying it. Furthermore, the geometry can be created using `blockMeshDict`, without the need for CAD. So, to summarize, this project will cover the last 3 enumerated items.

1.4 Prerequisites

Aside from the software needed to replicate this project, there are some prerequisites regarding the reader's qualifications. If the reader is capable of running tutorial cases provided by OpenFOAM, that is more than enough. If not, the reader will have difficulties following along, since this project is not self-contained, meaning the reader will have to search for missing information in other materials.

[2]Often times also simplifying the complex geometry which doesn't have a noticeable influence on the flow features.

1.5 Additional information

All of the cases used here and more material can be found on my GitHub repository. Every tool/software used in this project was of open source type. Here they are listed, and it is also noted which purpose they served.

1. **OpenFOAM 9** - a C++ toolbox used for numerical simulations,
2. **ParaView** - a powerful tool used for post-processing,
3. `Bash` - command language for job control, used for automating repetitive tasks,
4. **LibreOffice Calc** - spreadsheet software used for storing raw simulation data,
5. `GNU Octave` - high-level programming language, used for generating vector graphics plots from raw simulation data,
6. **Vim** - text editor of choice used for writing scripts
7. and **LaTeX** - document preparation software used for compiling this final document.

The reader is of course free to substitute in their software of preference.

Computational resources are often a topic that is hard to avoid while dealing with numerical simulations. Everything concerning this project was performed on a dinosaur laptop[3] with the following performance specifications.

Component	**Specifications**
CPU	Intel Core i5-4200U @ 1.6 GHz
RAM	8 GB DDR3
GPU	Integrated Intel HD Graphics 4400
HDD	SSD 480 GB SATA III
OS	Ubuntu 22.04.2 LTS

Table 1.1: Computational resources used in the project.

In addition, care was taken to avoid approaching any red zones while meshing.

[3]To encourage everyone and to discourage excuses.

2 The simulation

In this project, transient and incompressible simulations of the backward-facing step will be carried out. The variations in x-component of velocity - u will be examined with respect to:

1. **different mesh configurations**
2. and **different turbulence models - RANS and LES**.

Mesh will be generated with the help of OpenFOAM's `blockMesh` algorithm. Turbulence models $k-\varepsilon$ and $k-\omega$ SST are the backbones of the industry and is the reason why I decided to employ them specifically. Additionally, since the numerical results will be compared with the experiments, I decided to compare them with the Smagorinsky model, a model from the LES family. In this way, the fluctuations in field variables will not be totally smeared, which would be the case with RANS. It is often handy to classify the flow beforehand. Since the process medium in our flow is air, this is a problem of gas dynamics. The flow can further be classified, according to fluid mechanics as:

1. **internal**,
2. **incompressible**,
3. **transient**,
4. **advection-dominant**
5. and **adiabatic**.

It is clear that the flow is internal because the process medium is constricted to the physical boundaries of the domain - in our case a backward-facing step. Velocity will be chosen so that the variations in the pressure field are negligible. Quantities are free to change not only with space, but with time as well. The fluid behavior is thereby considered transient, instead of steady. Advection-dominant flows are flows where the dominant transport mechanism is the downstream movement of the fluid, opposed to the symmetrical process of diffusion being the driving force of transport. Péclet number can be used to discern which mode is dominant.

$$\mathrm{Pe} = \frac{\text{advective transport rate}}{\text{diffusive transport rate}}$$

2.1 Setting up the simulation

In this subsection, the following topics will be covered:

1. **creating the geometry**,
2. **meshing the geometry**
3. and **configuring the simulation parameters**.

2.1.1 Geometry and meshing

As mentioned previously, I will use OpenFOAM's native `blockMesh` algorithm. A 2D representation of our computational domain is shown in figure 2.1, along with the orientation with respect to the coordinate system.

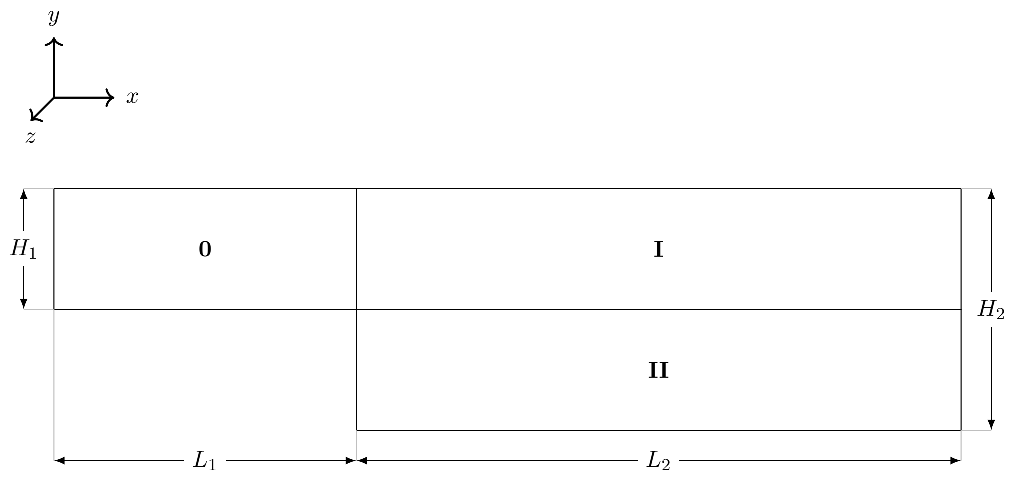


Figure 2.1: The computational domain and its characteristic dimensions.

Dimensions of the computational domain are be based on the dimensions provided in the paper, from section 4.

Dimension	**Value** [m]
Height H_1	0.0052
Height H_2	0.0104
Length L_1	0.2
Length L_2	0.5

Table 2.1: Dimensions of the computational domain.

The following meshing parameters are chosen for each of the 4 configurations.

Mesh configuration	**x1Cells**	**x2Cells**	**y1Cells**	**y2Cells**	**y3Cells**	**Total**
1	100	250	20	20	20	12000
2	120	300	24	24	24	17280
3	144	360	29	29	29	25056
4	173	432	35	35	35	37332

Table 2.2: Mesh configuration parameters used in the `blockMeshDict`.

The quantity **xiCells** represents the number of cells in the i^{th} block along the x-axis, where $i = 1, 2$. Same applies to **yjCells**, only this time along the y-axis, where $j = 1, 2, 3$. Column **Total**, from table 2.2, represents the total number of cells in the computational domain.

2.1.2 Configuring simulation parameters

The velocity vector will be denoted with $\mathbf{U}$. Due to the nature of the computational domain, it is obvious that the flow direction is aligned with the positive direciton of the x-axis, figure 2.1.

$$\mathbf{U} = u\mathbf{i} + v\mathbf{j} + w\mathbf{k}$$

Therefore, only the x-component of the velocity vector will be relevant when assigning the inlet velocity BC. In order to validate the results, I will be referencing a paper from section 4. In the paper, the authors state that for the given geometry, the following values of Reynolds number serve as flow regime classifiers.

Fluid flow regime	Reynolds number
Laminar	$\mathrm{Re} < 1200$
Transitional	$1200 < \mathrm{Re} < 6600$
Turbulent	$\mathrm{Re} > 6600$

Table 2.3: Qualitative classification of flow regimes for the given domain parameters.

For our case, velocity value of $u = 1.558269$ [m/s] will be used as the BC value. This corresponds to the Reynolds number of 1095 and a laminar flow regime according to table 2.3. Values that were unknown were treated by adopting default values of those quantities. Such quantities are presented in the table below.

Physical quantity	Value
Kinematic viscosity - ν	$1.48 \cdot 10^{-5}$ [m^2/s]
Turbulence intensity - I	5%
Model coefficient for turbulent viscosity - C_μ	0.09 [-]

Table 2.4: Assumed values of unknown quantities.

Knowing the value of velocity, parameters such as TKE - k, TKE dissipation - ε and the specific TKE dissipation rate - ω can be determined. OpenFOAM user guide was consulted, and the expressions used are provided below.

$$k = 1.5 \cdot (I \cdot |\mathbf{U}|)^2; \quad \varepsilon = \frac{C_\mu^{0.75} \cdot k^{1.5}}{D}; \quad \omega = \frac{\sqrt{k}}{C_\mu^{0.25} \cdot D}$$

In our case, the hydraulic diameter - D is defined by $D = 2H_1 = H_2$. This is adopted from the scientific paper.

Physical quantity	Value
TKE - k	0.0091 [m^2/s^2]
TKE dissipation rate - ε	0.0340 [m^2/s^3]
specific TKE dissipation rate - ω	16.7519 [1/s]

Table 2.5: Calculated quantities.

There were no other major changes to the `fvSchemes` and `fvSolver` settings. The simulation was set to last for a total of $t = 0.5$ [s], with a time step size of $\Delta t = 5 \cdot 10^{-5}$ [s]. According to the given velocity - u, flow passed only 1.11 times through the domain.

2.2 Running the simulation

All of the simulations are meant to be started from the CLI. The following commands should be executed in order to run each of the scripts, in the chronological order corresponding to figure 2.2.

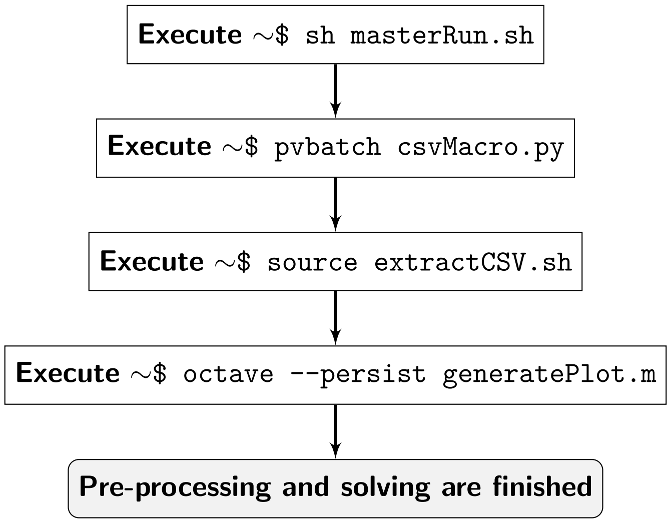

Execute ~$ sh masterRun.sh
Execute ~$ pvbatch csvMacro.py
Execute ~$ source extractCSV.sh
Execute ~$ octave --persist generatePlot.m
Pre-processing and solving are finished

Figure 2.2: A flowchart depiction of how to run the simulations.

It can be seen here that there will be a total of 5 scripts, including the `run.sh` script which is executed automatically by the `masterRun.sh` script. Figure 2.3 tells us that the majority of these scripts are written in `Bash` command language.

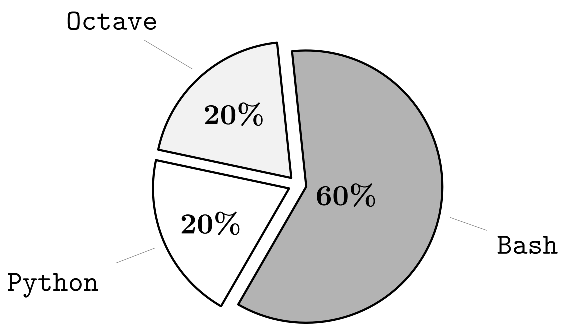


Figure 2.3: Overview of programming languages used.

`Python` could have been used for plotting the graphs as well, but I chose `Octave` since I am more comfortable with it. `Bash` scripts here are shown without comments, but they aren't complicated so it will be easy to understand each of the lines. More depth on the scripts is provided in the README file of the GitHub repository.

2.2.1 Bash scripts

Let's take a preliminary look at the contents of the scripts. The main script is called `masterRun.sh`. A simple script which goes into each of the different folders containing cases and runs the `run.sh` script. Upon the end of one `run.sh` script, the `masterRun.sh` starts a new one.

Listing 1: `masterRun.sh` script

```
#!/bin/bash
for i in kEps_config1 kEps_config2 kEps_config3 kEps_config4; do
        (
        cd $i;
        sh run.sh;
        cd ..;
        )
done
```

In this case, the loop will contain the names of the folders where the cases are, and where the `run.sh` scripts are found.

Up next is the `run.sh` script. It is an upgrade to the standard variant `run.sh`, that can be found in the tutorial cases. This script contains the commands for creating a bunch of logs, plotting the residuals, converting the files to VTK file format and exporting the residuals in .png format. I have also included a monitor of the Courant number.

Listing 2: `run.sh` script

```
#!/bin/bash
foamListTimes -rm
blockMesh | tee log.blockMesh
checkMesh | tee log.checkMesh
pyFoamPlotRunner.py --with-courant pisoFoam
pyFoamRedoPlot.py --pickle-file Gnuplotting.analyzed/pickledPlots --picture-prefix=kEps1095_
postProcess -func CourantNo
foamToVTK
killall gnuplot_x11
```

The `exportCSV.sh` script takes in all of the .csv files that are exported from ParaView, and combines the specified columns into one .ods file, which is later used for plotting in Octave. Before this script is run, it is necessary to export the .csv files from ParaView. This can be done manually, or with a `Python` macro.

Listing 3: `exportCSV.sh` script

```
#!/bin/bash
for i in {1..4}; do
    (
    cd kEps_config$i;
    for j in 019 023 029 055; do
        (
        awk -F',' -v OFS=, 'NR>1 {print $2}' \
        kEps_config${i}_${j}.csv >> ../forOctave_kEps1095_${i}_${j}.ods;
        )
    done
    )
done
paste *.ods >> forOctave_kEps1095.ods
find . -maxdepth 1 -type f -name '*.ods' ! -name 'forOctave_kEps1095.ods' -exec rm {} +
```

The `awk` command will ignore the 1st row of the .csv file and will take only the 2nd column. The line of code below the `awk` command will place that 2nd column into a new .ods file. The `paste` command will then concatenate all of the .ods files into one file. The last line of code in the `extractCSV.sh` script keeps only the final concatenated .ods file containing all of the columns of variable of interest.

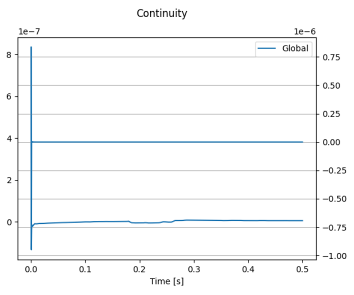
3 Post-processing

The following visual information will be presented:

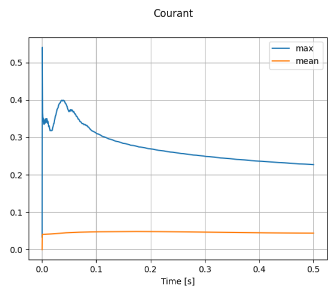
1. **residuals**
2. and **velocity variations** - as part of the mesh independence study.

3.1 Residuals

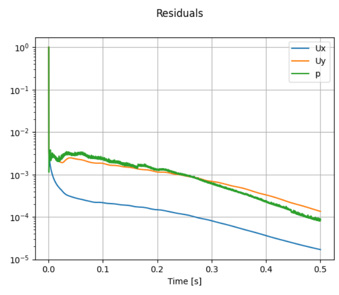
Residuals provided below are from the `Smagorinsky_config4` simulation. The residuals are plotted thanks to `pyFoam` software.



(a) Continuity residuals.



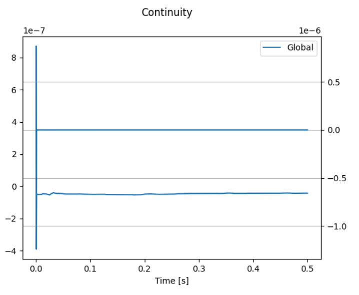
(b) Courant number evolution.



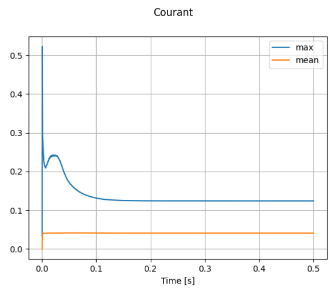
(c) Global residuals.

Figure 3.1: Residuals from the `Smagorinsky_config4` simulation.

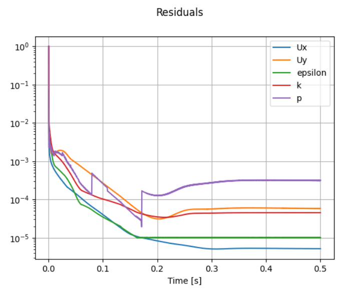
Residuals provided below are from the `kEps_config4` simulation.



(a) Continuity residuals.



(b) Courant number evolution.

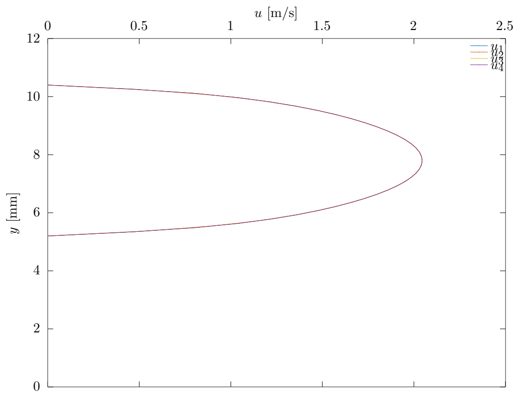


(c) Global residuals.

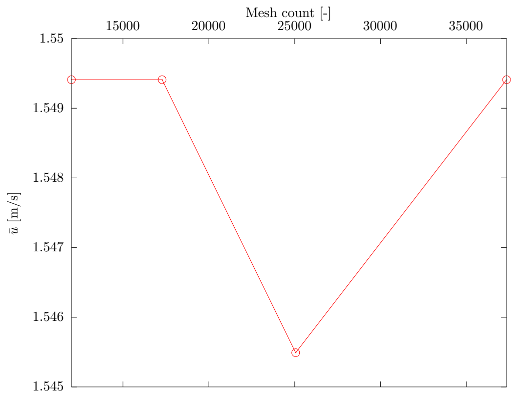
Figure 3.2: Residuals from the `kEps_config4` simulation.

3.2 Mesh independence study

These plots are generated with the aid of the `Octave` script. The only thing to note is the x/h notation, which represents the reattachment length to step height ratio. The reattachment length coordinates were adopted from the paper.

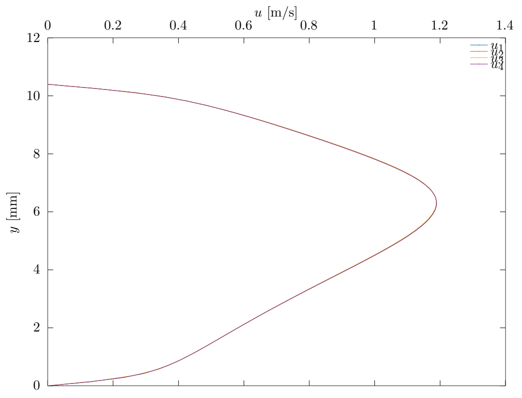


(a) Velocity profile at $x/h = -1.76$, for different mesh configurations.

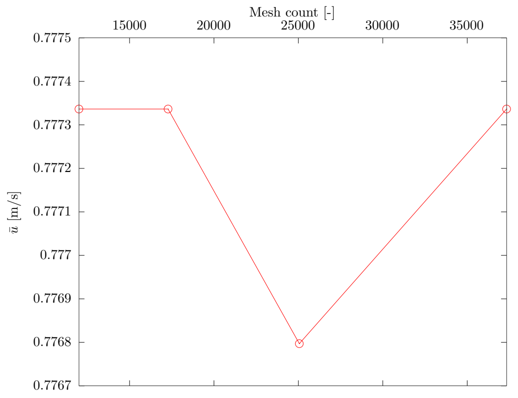


(b) Average velocity - $\bar{u}$ at $x/h = -1.76$, for different mesh configurations.

Figure 3.3: Velocity profile and average velocity for different mesh configurations at $x/h = -1.76$.



(a) Velocity profile at $x/h = 7.04$, for different mesh configurations.



(b) Average velocity - $\bar{u}$ at $x/h = 7.04$, for different mesh configurations.

Figure 3.4: Velocity profile and average velocity for different mesh configurations at $x/h = 7.04$

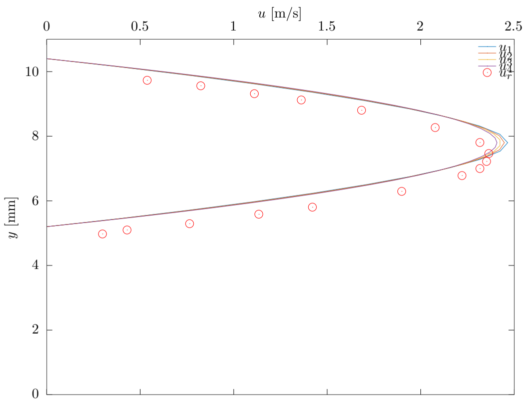
From figures 3.3 and 3.4, it can be noted that the variations in velocity with respect to the mesh are pretty much nonexistent. Therefore, the variations in velocity are independent of the mesh. These velocity profiles are the result of RANS, where the $k - \varepsilon$ turbulence model was used.

4 Validation of CFD with EFD

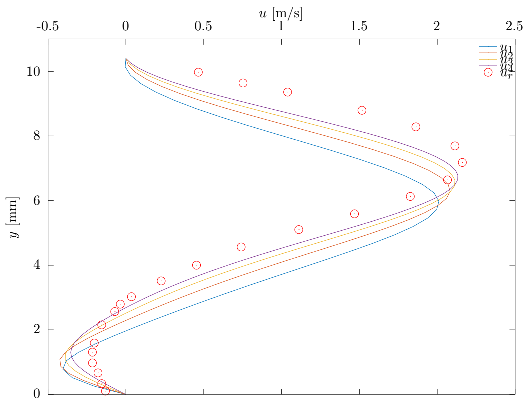
In order for the numerical simulations to be considered valid, a CFD user is often given two choices. To perform:

1. **EFD - experimental fluid dynamics**
2. and/or **hand calculations**.

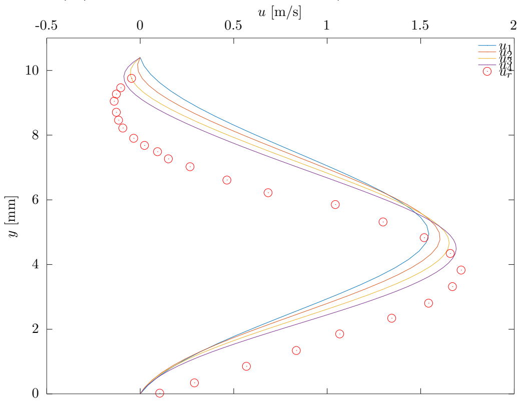
More often than not, the 2nd option is difficult to carry out, since only a variety of fluid flow problems can be solved analytically. The 1st option necessitates experience and equipment, and it is the next best thing other than the real-case scenario. For the purpose of this project, I will compare some of the numerical results with a scientific paper[4]. EFD velocity profiles are labeled with u_r in the legend, and are shown with red circles on the plots. Velocity profiles obtained from LES simulations are represented by continuous parabolas.



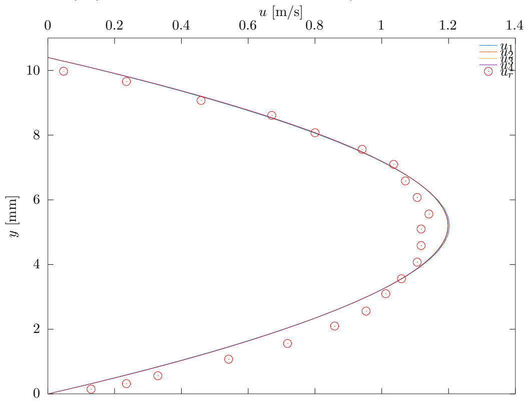
(a) Velocity profile at $x/h = -1.76$.



(b) Velocity profile at $x/h = 7.04$.



(c) Velocity profile at $x/h = 19.04$.



(d) Velocity profile at $x/h = 67.44$.

Figure 4.1: CFD results compared to EFD results, FIGURE 5 from the referenced paper.

From figure 4.1, a couple of trends can be observed. The plot from figure 4.1a describes a velocity profile before the step, and therefore the separation effects are not yet present. This

[4]Experimental and theoretical investigation of backward-facing step flow, B.F. Armaly, F. Durst, J.C.F. Pereira and B. Schönung. Institute of Hydromechanics, Section III: Mechanics of Turbulent Flows, University of Karlsruhe, Kaiserstraße 12, D-7500 Karlsruhe, F.R.G. The paper can be accessed using this link.

is the reason a general agreement between EFD and CFD is visible. The same can be said for figure 4.1d, however this time the velocity profile is measured long after the step, where the bulk flow becomes fully developed again. The other two figures, 4.1b and 4.1c, have a noticeable discrepancy in a quantitative sense. The trend from CFD is qualitatively still in agreement with EFD. However, discrepancies in figures 4.1b and 4.1c from CFD, can be explained by acknowledging that turbulence is inherently a 3D phenomenon. By disregarding the 3^{rd} dimension in our simulations, some of the effects present in the flow field haven't been taken into account. Authors from the paper confirm this by stating that values of $\text{Re} < 400$ and $\text{Re} > 6000$ yield a 2D flow, while a 3D flow is exhibited inbetween. Lastly, what are some improvements that could bridge the gap between CFD and EFD in this case? Firstly by refining the mesh in LES, then by using high-resolution discretization schemes, potentially performing DNS in order to resolve the small eddies as well and extending the computational domain to the 3^{rd} dimension.

5 Acknowledgements and outro

I would personally like to thank the following contributors for inspiring this project, and for providing quality material from which I learned a lot.

Asmaa Hadane József Nagy Theodore Ong

I hope that the readers will find my material of use. The readers are more than welcome to send me an email in case of mistakes, remarks, and/or suggestions. Alternatively, you can reach out to me on LinkedIn as well! I look forward to hearing your comments and insights.