

MASTER'S THESIS

---

**ONSET  
OF  
TURBULENCE**

---



A DIRECT NUMERICAL SIMULATION STUDY OF  
TRANSITIONAL TURBULENCE USING THE FINITE  
ELEMENT METHOD

MADS HOLST AAGAARD MADSEN

"WHEN I MEET GOD,  
I AM GOING TO ASK HIM TWO QUESTIONS:  
WHY RELATIVITY? AND WHY **TURBULENCE**?  
I REALLY BELIEVE HE WILL HAVE AN ANSWER FOR THE FIRST."

- WERNER HEISENBERG, 1901-1976

---

# ONSET OF TURBULENCE

---

Author	Mads Holst Aagaard Madsen
Supervisor	Joachim Mathiesen, Associate Professor
Co-Supervisor	Gaute Linga, PhD-fellow



*The thesis concludes  
one academic semester of:*

**30 ECTS**  
*during spring 2016*

**Research Group**

Research department of Bio-Physics, Niels Bohr Institute

Submitted to the University of Copenhagen  
August 8, 2016

## ACKNOWLEDGEMENTS

First and foremost I would like to express my profound gratitude to my supervisor Joachim Mathiesen for introducing me to the fascinating physics of continuous matter. I also feel indebted to PhD-fellow Gaute Linga for his general assistance throughout the research. Without their guidance and helpful discussions, this work could not have been done. Furthermore, I would like to thank M. Misztal and M. Svenningsen as well as A. Garcia and R. Martin for their cooperation and sparring concerning the simulations carried out in the present work. Finally, I thank my friends and family for invaluable inputs.



# CONTENTS

<b>I</b>	<b>Introduction</b>	<b>1</b>
<b>1</b>	<b>Motivation</b>	<b>2</b>
1.1	The thesis in short . . . . .	3
1.1.1	Thesis structure . . . . .	4
1.1.2	Notation . . . . .	5
<b>2</b>	<b>Setup &amp; Solvers</b>	<b>6</b>
2.1	FEniCS . . . . .	6
2.1.1	Oasis . . . . .	7
2.2	Lattice Boltzmann Method . . . . .	7
<b>3</b>	<b>Contributions</b>	<b>9</b>
3.1	Motivation revisited . . . . .	10
<b>II</b>	<b>Theory of Continuous Matter</b>	<b>12</b>
<b>4</b>	<b>Equations of Fluid Dynamics</b>	<b>13</b>
4.1	Incompressibility . . . . .	13
4.2	Continuity equation . . . . .	14
4.3	Cauchy's equation . . . . .	14
4.4	The Navier-Stokes Equations . . . . .	15
4.4.1	Scaling of NS-equations . . . . .	15
4.4.2	Simplified setups . . . . .	17
<b>5</b>	<b>Ducts and pipes</b>	<b>20</b>
5.1	Laminar flow regime . . . . .	21
5.1.1	Entrance region . . . . .	21
5.1.2	Steady Hagen-Poiseuille flow . . . . .	23
5.2	Transitional flow regime . . . . .	25

5.3	Turbulent regime . . . . .	25
5.3.1	Shear stress . . . . .	26
5.3.2	Boussinesq hypothesis . . . . .	28
5.3.3	Layers of turbulent flow . . . . .	28
<b>6</b>	<b>Turbulence Theory</b>	<b>33</b>
6.1	Fully developed turbulence . . . . .	33
6.1.1	Specific rate of dissipation . . . . .	33
6.1.2	Kolmogorov's theory . . . . .	35
6.2	Reynolds-averaged Navier-Stokes equations . . . . .	36
6.2.1	Reynolds stress tensor . . . . .	37
<b>7</b>	<b>Onset of Turbulence</b>	<b>39</b>
7.1	Historical overview . . . . .	39
7.1.1	Dimensionless units . . . . .	40
7.1.2	Directed Percolation . . . . .	41
7.2	Front Dynamics . . . . .	43
7.2.1	Edge profiles . . . . .	45
7.2.2	Current FD status . . . . .	47
<b>III</b>	<b>The Finite Element Method</b>	<b>50</b>
<b>8</b>	<b>Computational method</b>	<b>51</b>
8.1	Finite Element Method . . . . .	51
8.1.1	Method of weighted residuals . . . . .	52
8.1.2	Weak formulation . . . . .	55
8.1.3	Piecewise Linearity . . . . .	55
8.1.4	Finite Element Formulation in 1D . . . . .	56
8.1.5	FEM example in 1D . . . . .	57
8.2	FEM generalization for higher dimensions . . . . .	58
8.2.1	Problem formulation . . . . .	59
8.2.2	Discretization . . . . .	59
8.2.3	Transient analysis . . . . .	62
<b>9</b>	<b>Fluid simulations</b>	<b>68</b>
9.1	Stokes flow solver . . . . .	68
9.1.1	Stokes flow . . . . .	68
9.1.2	Problem statement . . . . .	69
9.1.3	Steady Stokes flow system equation . . . . .	72
9.1.4	KKT Boundary conditions . . . . .	73

9.1.5	Pseudo-compressibility . . . . .	74
9.2	Stokes flow solver verification . . . . .	76
9.2.1	Solution assessment . . . . .	76
9.3	Navier-Stokes solver . . . . .	79
9.3.1	Stokes flow transient analysis . . . . .	79
9.3.2	Convection implementation . . . . .	80

## **IV Computational Fluid Dynamics 82**

<b>10</b>	<b>Flow simulations and analytical comparisons</b>	<b>83</b>
10.1	Mathematical framework of FEM . . . . .	83
10.2	Verification of FEniCS implementation . . . . .	85
10.3	Laminar pipe flow . . . . .	86
10.3.1	Weak Formulation . . . . .	86
10.3.2	Setup . . . . .	87
10.3.3	Solution to laminar pipe flow . . . . .	87
10.3.4	Solution assessment . . . . .	89
10.3.5	Complex meshes . . . . .	93
<b>11</b>	<b>Transient solvers</b>	<b>95</b>
11.1	Chorin's Method . . . . .	95
11.1.1	Coupled and segregated solvers . . . . .	96
11.1.2	FEM scheme . . . . .	96
11.1.3	Solver implementation . . . . .	97
11.1.4	Verification of fractional step solver . . . . .	98
11.1.5	Solution to transient pipe flow . . . . .	100
<b>12</b>	<b>Oasis implementation</b>	<b>104</b>
12.1	Solver and iteration scheme . . . . .	104
12.1.1	Incremental Pressure Correction Scheme . . . . .	104
12.2	Setup and verification . . . . .	107
12.2.1	Setup . . . . .	107
12.2.2	Verification . . . . .	108
<b>13</b>	<b>Initialization</b>	<b>113</b>
13.1	Setup . . . . .	113
13.2	Steady-state solver . . . . .	114
13.2.1	Stabilization . . . . .	114
13.2.2	Solver settings . . . . .	116
13.3	Assessment of steady-state . . . . .	120

13.4	Fenicstools . . . . .	123
13.5	Mesh considerations . . . . .	124
13.5.1	Mesh partitioning . . . . .	124
13.5.2	Mesh design . . . . .	125
13.5.3	Grid limitations . . . . .	126
<b>V</b>	<b>Onset of Turbulence</b>	<b>128</b>
<b>14</b>	<b>Turbulence simulation results</b>	<b>129</b>
14.1	Profile assessment . . . . .	130
14.1.1	Profiles . . . . .	130
14.1.2	Kinetic energy . . . . .	131
14.1.3	The law of the wall & the logarithmic law . . . . .	133
<b>15</b>	<b>LBM comparison</b>	<b>139</b>
15.1	Motivation for comparison . . . . .	139
15.2	Metrics of comparison . . . . .	140
15.3	High-performance computing . . . . .	140
15.4	Profile comparisons . . . . .	142
<b>16</b>	<b>Onset of Turbulence</b>	<b>146</b>
16.1	Front dynamics . . . . .	147
16.1.1	Strong slugs . . . . .	147
<b>17</b>	<b>Discussion &amp; Outlook</b>	<b>154</b>
17.1	Discussion . . . . .	154
17.1.1	Turbulence simulation results . . . . .	154
17.1.2	LBM comparison . . . . .	155
17.1.3	Front dynamics . . . . .	157
17.2	Ensuing research . . . . .	158
<b>18</b>	<b>Conclusion</b>	<b>160</b>
<b>A</b>	<b>Appendix</b>	<b>161</b>
A.1	Physics in ducts and pipes . . . . .	161
A.2	Transient Hagen-Poiseuille flow . . . . .	164
A.3	FEM examples using self-written code . . . . .	165
A.3.1	FEM 2D static example . . . . .	165
A.3.2	FEM 2D transient examples . . . . .	166
A.3.3	Stokes' problems and analytical comparison . . . . .	169
A.4	Flow around objects . . . . .	173

---

A.4.1	Importance of functionals . . . . .	173
A.4.2	Setup and variational problem . . . . .	174
A.4.3	Flow solutions around an airfoil . . . . .	176
A.4.4	Functional-based analysis . . . . .	176
A.5	Lid-driven cavity . . . . .	184
A.5.1	Setup . . . . .	184
A.5.2	Variational problem . . . . .	185
A.5.3	Newton's Method . . . . .	185
A.5.4	Solution to Lid-Driven Cavity . . . . .	186
A.6	Supplementary Oasis information . . . . .	187
A.6.1	Oasis verification . . . . .	187
A.6.2	Convection term . . . . .	187
A.6.3	Other initialization choices . . . . .	189
A.6.4	Kinetic energy plots for Oasis and LBM . . . . .	192
A.7	Scripts . . . . .	192
<b>Bibliography</b>		<b>197</b>

## ENGLISH ABSTRACT

The overall scope of this thesis is to investigate the front dynamics of localized turbulence using Direct Numerical Simulation (DNS) by solving Navier-Stokes equations for  $2400 \lesssim Re \lesssim 8300$ . Embedded within the work is a comparison of two computational methods as well as a comprehensive study of the Finite Element Method (FEM) and Computational Fluid Dynamics (CFD) exemplifying how to write a FEM-solver either from the ground up or by means of an automated program. The current state of the investigated research field, 'Onset of Turbulence', is addressed and the final results are put into this broader frame of reference thus filling yet another gap on the way to thoroughly understanding the elusive transition to fully developed turbulence.

## Part I

### INTRODUCTION

## MOTIVATION FOR PRESENT WORK

“Finally, there is a physical problem that is common to many fields, that is very old, and that has not been solved. It is not the problem of finding new fundamental particles, but something left over from a long time ago – over a hundred years. Nobody in physics has really been able to analyze it mathematically satisfactorily in spite of its importance to the sister sciences. It is the analysis of *circulating or turbulent fluids*.” [37, chapter 3, p. 15]

- **Richard P. Feynman**, 1964

American Nobel Prize Laureate in Physics



HY study the turbulent motion of fluids? Through the years the subject has occupied many a bright mind, including Feynman's. Such was his reverence for the topic that he dubbed Turbulence the 'most important unsolved problem of classical physics'. It has always seemed fascinating to me that such a topic has transfixed scientists for over a century. Now that the Age of the Computer has dawned upon us several decades ago the possibility for glimpses of what the answer to this problem might eventually look like increases day by day. It is therefore the scope of the present work, along with a general introduction of the Finite Element Method (FEM) and Computational Fluid Dynamics (CFD), to use Direct Numerical Simulations (DNS) in order to investigate the behavior of turbulent fluids. Furthermore, a comparison between the FEM and the Lattice Boltzmann Method (LBM) is sought in the turbulent regime and



finally, the simulation results of localized turbulence will be compared to those from relevant literature.

## 1.1 The thesis in short

There are five overall parts in the thesis. Part **I** is an introduction to the work in general. Here, key components such as programs and methods which are integral to the thesis will be described in chapter **2**. Furthermore, a chapter (**3**) is dedicated to point out exactly what the contributions are within the present work. Concludingly, the motivation is revisited in section (**3.1**).

Part **II** consists of four chapters and gives a brief summary of relevant fluid dynamical equations (**4**), turbulence theory (**5,6**) and most importantly an overview of the particular field of research to which we hope to contribute (**7**). The scope here is to introduce the physics we will deal with throughout the ensuing parts and to give the reader a general understanding of the physics at play. The overview (**7**) hopefully serves to render a clear idea of the current state of the research field. Thus, it is easier to know the relevance of the present work and to better place it in a larger context. Throughout the theoretical introduction figures exemplifying the phenomena described will be given using data from the simulations. This will hopefully allow for an easy, intuitive understanding that helps connect the somewhat dusty formulas with a feeling for the underlying physical mechanisms. It also seems prudent to mention here that a select number of quotes are given throughout the work. This is not 'name-dropping', but intended to give a glimpse of how bright minds through the century have perceived this fascinating topic.

Part **III** is a study of the FEM. The study has the general purpose of introducing the reader to the method employed throughout this work (**8**). Specifically, it will be shown how one can build a Navier-Stokes (NS) solver from the ground up and subsequently test it (**9**). The structure of part **III** is intended to have a natural progression. In chapter **8** we introduce the basic formalism in 1D (**8.1**) and then generalize it to higher dimensions (**8.2**). We then solve problems with increasing difficulty that are all relevant sub problems for the NS-equations. By solving Poisson's equation one learns how to deal with the diffusive term ( $\nabla^2 \mathbf{u}$ ) and the body force ( $\mathbf{f}$ ) in NS-equations. Here, both Dirichlet and Neumann boundaries are also handled. Subsequently, the transient term ( $\frac{\partial \mathbf{u}}{\partial t}$ ) is introduced with the heat equation. In chapter **9** the solver is tested on two famous test cases (Stokes' problems, **A.3.3**) and finally we introduce the notion of

a staggered grid to *couple* the velocity and the pressure (9.1). The final product is a *coupled* solver with linear shape functions for velocity and a pressure assumed constant over the element (P1-P0 Lagrange elements). There are numerous test cases throughout part III out of which we choose flow in a pipe as the final one since the majority of work has been carried out in pipes and ducts. Although irrelevant for the pipe flow setup, the convective term  $((\mathbf{u} \cdot \nabla)\mathbf{u})$  is finally dealt with by introducing Picard linearization and thus, we have touched upon every single term in the NS-equations and are in principle ready to dive into the arts of CFD with a solver written from scratch.

Part IV is an introduction to CFD using the software FEniCS (described below in section 2.1). Similarly to part III there are several test cases where integral components in CFD such as *error convergence* and *functional computation* are described (10). Subsequently, a *segregated* NS-solver is built and compared to transient Hagen-Poiseuille pipe flow to ensure its correctness (11). In chapter 12 the FEM-solver *Oasis* is introduced as it is the one we will use in part V. The final chapter (13) describes the important initialization step necessary to obtain our final results.

The last part (V) entails the application of FEniCS using the *Oasis*-solver. Specifically, DNS results of turbulence in a (deformed) duct are presented and compared to two famous laws of physics (presented in part II) governing the motions of turbulence (14). This is to ensure the functionality of the FEM-solver we use. Subsequently, a comparison to the LBM is carried out covering computational performance and statistics of turbulence (15). This comparison of computational methods aims to identify pros and cons of the two methods. Finally, the current state of the research field, 'Onset of Turbulence', (initially addressed in part II) is reviewed and the results from present work is placed in the greater framework of results known from the literature (17). Thus hopefully, we will further increase our insight into the transitional regime of turbulence.

### 1.1.1 Thesis structure

The thesis structure is representative of the work I have done during the semester; spring 2016. A rather large amount of time has been dedicated to a study of the FEM and CFD as well as the physics of continuous matter. The acquired insight has been instrumental to the success of the overall scope of the thesis: investigations into the research field 'Onset of Turbulence' by conducting DNS. The essence of these studies (part II, III and IV) have therefore been presented and they are just as relevant to the thesis as the remaining parts.

However, since the thesis is of a considerable length it has been designed with a substructure dedicated to accomplished fluid dynamicists and seasoned FEM practitioners who only wish to read the sections specifically pertaining to the investigations into 'Onset of Turbulence'. This is fully possible. Such readers are advised to follow the roadmap given below:

- A brief description of 'Onset of Turbulence' (chap. 7)
- Presentation of DNS results describing the characteristics of localized turbulence (chap. 16)
- An assessment of the results and how they relate to results from the literature (sec. 17.1.3)
- Ensuing research proposal based on our results (sec. 17.2)
- Conclusion (chap. 18)

Once again; this is only a roadmap for a focused *but partial* reading of the thesis.

### 1.1.2 Notation

I have tried to use the same notation throughout the different parts of the thesis. For interested readers consulting references for further explanation it can on the other hand also be helpful that the notation given in the present work somewhat resembles the one in the references. There are therefore a few changes from part to part in the thesis, e.g. the trial and test function in part III are denoted:  $\tilde{\mathbf{u}}$  and  $\mathbf{w}$ . The notation changes to  $u$  and  $v$  respectively in part IV and onwards. Save for the minor changes the notation will hopefully be self-consistent.

### Scripts and references

The considerable amount of scripts produced throughout the work in this thesis both for producing figures and statistical results are available upon request. Please contact `bk1886@alumni.ku.dk` for the desired code. One script however is provided in the appendix(A.7). It is with this script we generated the final turbulence data.

As for references, there is extensive citing throughout the thesis - much more than one would find in a scientific paper. This is for interested readers in need of further explanation or for those who want to check the background of a given formula.

## PROGRAMS, SOLVERS & SETUP

**T**HE following is a brief presentation of programs and solvers used in the present work. The descriptions should indeed be supplemented with further reading if one is to use any of the below-mentioned programs. It merely serves to acquaint the reader with acronyms such as FEniCS or Oasis as they will be present throughout the text.

### 2.1 FEniCS

FEniCS is a scientific computing platform that can be utilized to find efficient solutions to differential equations. It focuses on *automated* solutions to PDEs by employing FEM. As will be described in depth in part **III** and **IV** the FEM approach is to write up a variational problem which then leads to the assembly of a system equation. *Automated* means among other things that the assemble-step is done “behind the scenes” so to speak. One does not have to write an assemble function explicitly but can simply call the `assemble()` function. Another trademark of FEM is the decomposition of the domain into small elements which is particularly useful for non-trivial boundaries. Thus, flow simulations of fluids can be obtained even for complex geometries. After having derived the weak formulation one can simply feed FEniCS the variational problem to obtain a solution. FEniCS is a cooperative project between universities counting Cambridge and Chicago amongst them. Obtaining and using the software is fairly straightforward at <http://fenicsproject.org/>.

### 2.1.1 The generic fractional-step Oasis solver

The Optimized And Stripped Solver (Oasis) that we will use in our turbulence investigation is developed by M. Mortensen, K. Valen-Sendstad and J. Bø. Below, a short description is given. For further information on Oasis Refs. [16, 23, 24, 25] should be consulted. It is this solver that will be implemented and tested in chapter 12 before we use it to obtain the final turbulence simulation results in part V. Oasis is an open-source state-of-the-art FEM NS-solver written in Python. It can run with Message Passing Interface (MPI) which allows the user to solve large problems by running the code in parallel. It is put together by FEniCS blocks and interfaces (in our case) with the linear backend PETSc which likewise is state-of-the-art [50]. Installation is done simply by cloning to the git. In chapter 11 we show how to build a completely well-functioning NS-solver; a very manageable task once you get the hang of it. The reason for using Oasis is that it has been optimized in numerous ways. The solver has been shown to scale weakly up to 256 CPUs (using P2-P1) which is one of the main reasons we have chosen to use it. For the complete list of precautions taken to optimally speed up Oasis we refer to Ref. [16].

#### Setup

The turbulence simulations will be carried out in a deformed duct of dimensions  $[1 \times 1 \times 40]$  where the deformation consists of 4 bumps in the  $xz$ -plane. The design of the mesh will be described in more detail later. Here, it shall suffice to mention that by using periodic boundaries we get an 'infinite' mesh which allows us to observe structures in the fluid over long distances.

## 2.2 Lattice Boltzmann Method

The Lattice Boltzmann Method (LBM) is a rather new CFD method that instead of solving the NS-equations simply solves the Boltzmann-equations. The idea here is to model the fluid as a set of particles and then compute their interaction and thereby their propagation on a computational mesh. Standard Lattice Boltzmann schemes have uniform regular grids which are aligned with some *characteristic* directions for velocity. This means discretization and particle-velocity are coupled. Newer LBMs are *off-lattice* to better mirror boundaries. There are obviously many versions of the method and throughout the thesis whenever we refer to LBM it is with the implementation at the Niels Bohr Institute (NBI)

at University of Copenhagen in mind. This LB implementation is based on unstructured grids which means the space and velocity discretization are *uncoupled*. The implementation is presented in Ref [48] as having first-order accurate temporal discretization,  $\mathcal{O}(\Delta t^1)$ , and second-order accurate spatial discretization,  $\mathcal{O}(\Delta x^2)$ . The current version has been improved to  $\mathcal{O}(\Delta t^2)$  which mirrors the settings we will use for Oasis.

## CONTRIBUTIONS WITHIN THE PRESENT WORK

**T**HE thesis has key contributions distributed throughout the four final parts II-V. It is always important to discern what work has actually been done by an author and when references should be attributed instead. In general, more or less all I know of FEM, CFD and the physics of continuous matter is from the listed Refs. Though naturally, the relevance of a given source varies immensely from section to section throughout the work. Therefore, as a rule of thumb, I will try to mention the most relevant source(s) at the beginning of each section.

In part II I give a (minimal) review addressing the current state of the research field 'Onset of Turbulence'. This is given in order to better relate the present work to other contributions.

In part III a coupled 2D solver is put together and tested against the analytical expression for pipe flow. The most relevant references are [54, 15, 11, 44] which describe all the techniques used. In other words; I have not invented the wheel in this section - I simply made it turn. Here, it is only natural to mention M. Misztal and M. Svenningsen. The latter had done a project on cell dynamics simulations and shared both code and thoughts with me. Despite the different setup it was an enlightening sparring. The former, being an author of several papers I used, kindly explained any lingering doubts. Thus, I feel indebted to both. The chosen solution is one out of many ways to solve the NS-equations and certainly serves as more than a mere exercise since it is futile to think one can control state-of-the-art solvers without having profound understanding of the

underlying processes based on first-hand experience.

In part **IV** I show how a segregated 3D solver using Chorin's splitting method can be assembled and tested against the analytical expression for transient Hagen Poiseuille flow. Here, Ref. [4] and the online **2D example** on Chorin's method have been a good starting point for putting together the presented solver. Subsequently, it is described how to utilize Oasis and how to test the correctness of a given implementation. Thus, in part **IV** it is both shown that it certainly is possible to put together 'one's own' NS-solver using FEniCS but also, that should one be in need of state-of-the-art performance it is feasible to implement high-performance solvers. Finally, I show how to further modify Oasis by implementing statistics modules or by setting up initialization profiles calculated from a coupled, iterative solver.

Part **V** is the culmination of the thesis where I show (1) how to assess data of turbulence using friction units, (2) how to compare computational methods and finally (3) how to produce turbulent results with relevance to the adherent research field.

### 3.1 Motivation revisited

Several engineering acquaintances posed the questions (when reading early drafts): Is there an actual experiment carried out - and is it even physically relevant? The answer is unequivocally NO and YES!

No, currently there is no physical, actual counterpart to the duct mesh we simulate in. NBI will gladly assume the role of the recipient to generous donations enabling the experiment which will start as soon as funding is acquired. Besides, both Oasis and the LBM implemented at NBI have been tested in various (physically relevant) setups: the LBM was tested on a real sample of a porous rock from Rødvig, Denmark. The sample was X-ray scanned to obtain a mesh [48]. Likewise, Oasis has already been used in the study of blood flow in intracranial aneurysms [47, 10]. Similarly, a geometry was obtained by a (CT) scan of the physical model. The purpose of part **V** is a *test* of Oasis and *comparison* to the LBM as well as to the broader literature. As such, the important thing is that the two solvers simply use the *same* mesh.

The thesis should be viewed as a part of a greater whole: During the fall of 2016 large scale turbulence simulations will be carried out at NBI. To this end a complementing solver to the in-house LBM solver is sought. Likewise, tentative investigations into the research field 'Onset of Turbulence' are needed to better optimize the design of the ensuing



project. This thesis provides all of the above and is thereby the foundation for the pending research.

## Part II

### THEORY OF CONTINUOUS MATTER

## EQUATIONS OF FLUID DYNAMICS

**T**HE continuum equations we will deal with are basically based on two pillars: one is the *conservation of mass* and the other *Newton's Second Law* applied to continuous matter. The latter is also sometimes called the *momentum balance*. The equations we end up with will describe fields for *mass density*,  $\rho$ , and *velocity*,  $\mathbf{u}$ . Despite the elegant simplicity, the continuum equations are only analytically solved in a limited amount of cases which of course is why we simulate them.

### 4.1 Incompressibility

Throughout this work we will only be dealing with incompressible fluids. When in need we alter the physics slightly by adding some *pseudo-compressibility*, but this is only to render our system solvable. Fluids are in general (practically) incompressible as long as the velocity is well below the speed of sound. Thus, any conclusions based on the ensuing simulations should only be drawn with this in mind. Obviously, true incompressibility would mean an infinite speed of sound so this is an approximation - but one of many one usually makes when simulating. Mass conservation is e.g. also not given in Relativity.

Truly incompressible fluids cannot accumulate in any region. This is stated for an arbitrary volume in the global incompressibility equation which locally has the well-known form seen below [20, eq. 12.4-5]:

$$\nabla \cdot \mathbf{u} = 0 \quad (4.1)$$

## 4.2 Continuity equation

In fluid dynamics the *continuity equation* is one of the central differential equations and can be seen below [20, eq. 12.12]. To derive it one simply writes up the *global equation of mass conservation* for a fixed volume:  $\frac{d}{dt} \int_V \rho dV = - \oint_S \rho \mathbf{u} \cdot d\mathbf{S}$ , and apply Gauss' divergence theorem using a fixed surface:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \quad (4.2)$$

The expression also follows from Reynolds Transport Theorem.

Using the material derivative we can restate the continuity eq. 4.2 as  $\frac{D}{Dt} \rho = -\rho \nabla \cdot \mathbf{u}$  which simplifies to the incompressibility eq. 4.1 when the material derivative is zero.

## 4.3 Cauchy's equation

Applying Newton's Second Law to a material particle: mass,  $\rho dV$ , times acceleration,  $\frac{D\mathbf{u}}{Dt}$ , equals force,  $d\mathbf{F}$ , will result in the Cauchy equation from 1827 if we divide with  $dV$  [20, eq. 12.26]:

$$\rho \left( \frac{D\mathbf{u}}{Dt} \right) = \mathbf{f}^* \quad (4.3a)$$

$$f_i^* = f_i + \sum_j \nabla_j \sigma_{ij} \quad (4.3b)$$

Above,  $\mathbf{f}^*$  is called the *force density* and is given in eq. 4.3b. As seen, it consists of *body forces*,  $f_i$ , and stress,  $\sigma_{ij}$ . This equation entails the dynamics for all continuous matter. To obtain the Navier-Stokes equations we only have to insert the stress tensor for fluids in the effective force density.

We now have the field equations of motion for  $\rho$  and  $\mathbf{u}$  in respectively eq. 4.2 and 4.3a. Should we know the fields  $\rho$  and  $\mathbf{u}$  at any time we can solve for the temporal derivative,  $\frac{\partial}{\partial t}$ , and iterate forward in time,  $t + \Delta t$ . To be fair, we have left out the field equation for the Eulerian displacement field (not to mention pressure, temperature or the constitutive force equation). The reason that the displacement field is not needed is that we are not dealing with elastic or viscoelastic materials that can remember their previous position.

## 4.4 The Navier-Stokes Equations

The most important equation for simulations of flow is Navier-Stokes equation describing incompressible, isotropic and homogeneous fluids:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}, \quad \nabla \cdot \mathbf{u} = 0 \quad (4.4a)$$

$$\sigma_{ij} = -p \delta_{ij} + \mu (\nabla_i u_j + \nabla_j u_i) \quad (4.4b)$$

Having already derived the Cauchy equation we only need the expression for the Newtonian stress tensor seen in eq. 4.4b which by insertion gives NS-equations. We write in plural since naturally eq. 4.4a gives equations for  $u_x, u_y, u_z$  and  $p$  [20, eq. 15.20-21].

Notice that the body force,  $\mathbf{f}$ , is really force per unit mass, i.e. an *acceleration field* (the *true body force density*,  $f_i$  divided by *mass density*  $\rho$ ). However, for all practical purposes the term is often referred to simply as body forces and thus, we will keep  $\mathbf{f}$  as the corresponding symbol. In e.g. simulation tutorials, one will often not find the  $1/\rho$  factor in front of the pressure term. This is merely a convenience - they simply consider the equations with 'unit fluid density' [17, p. 395] [20, p. 334].

### 4.4.1 Scaling of NS-equations

To be able to compare simulations with work done by others it is important to know how to *classify* flows. This is because simulating fluids with very different parameters will obviously result in very different behavior or motion of the fluid. The most widely used tool for such flow classification is the *Reynolds number*,  $Re$ , which describes how lively a fluid is. The general idea for flow classification is to non-dimensionalize the NS-equations and end up with one dimensionless parameter,  $Re$ , which can be used as a metric when assessing how similar flows actually are. Two simulations of e.g. pipe flow with differing diameters will *hydrodynamically* be similar if the Reynolds numbers are close. The results will simply differ in length and velocity scale. We are in other words exploiting the flow similarity for congruent geometries. This has in relevant literature also been described as the study of *transformation properties*, *invariance properties* or simply *symmetries* [20, p. 255] [3, p. 56].

### NS-equations and forcing

For setups such as ours where the inlet and outlet boundaries of the duct are connected by periodic boundaries one obtains an 'infinite' duct. In this case, one cannot impose a pressure gradient but must instead use the force term to apply a *forcing* to drive the fluid. Therefore, we cannot leave out the body forces when rescaling (as otherwise is typically the case). Using the rescaling:

$$u'_i = \frac{u_i}{u_{avg}}, \quad x'_i = \frac{x_i}{L_0}, \quad p' = \left( \frac{L_0}{\rho \nu u_{avg}} \right) p, \quad f'_i = \frac{L_0^2}{\nu u_{avg}} f_i, \quad t'_i = \frac{\nu}{L_0^2} t \quad (4.5)$$

where  $L_0$  is a characteristic length of the setup,  $f_i$  are the components of force per unit mass and  $u_{avg}$  is a reference velocity, we can obtain the *non-dimensionalized* NS-equations:

$$\partial_t u_i + Re u_j \partial_j u_i = -\partial_i p + \nabla^2 u_i + f_i \quad (4.6a)$$

$$\partial_j u_j = 0 \quad (4.6b)$$

A. J. Chorin used this exact scaling in his paper where he presented the splitting method named after him [4]. As seen above we have dropped the primes after insertion but the constants,  $\nu$  and  $\rho$ , are gone. Instead the Reynolds number,  $Re$ , now figures.

A side note for interested readers checking the corresponding scaling in Chorin's 1966 paper: the discrepancy in force scaling *is* on purpose. It seems A. J. Chorin unintentionally has flipped his force-scaling [4, p. 1].

To be abundantly clear: in the present part (II) we give examples and to nurture the reader's intuition we use dimensions in these pedagogical examples. For part (III), (IV) and (V) there will *not* be units on e.g. time-step or length specifications since the mesh and all our parameters are unitless. Should one be interested in a particular, physically relevant measure then one can simply use the scalings given above. Our procedure is completely equivalent to the one seen in the paper presenting Oasis [16].

### The Reynolds number

As the only parameter left in the rescaled equations the Reynolds number is by far the most common tool for flow classification. For flow in pipes and ducts we have the three overall categories; Laminar, Transitional and Turbulent flow. 'Laminar' refers to the smooth, orderly motion seen for a fluid at low speed whereas 'turbulent' flow is disordered and marred by fluctuations and vortices. Each regime change is marked with a critical Reynolds number,  $Re_{cr}$ , that depends on the geometry and flow conditions. It is not always straightforward to determine these

(approximate) thresholds as many factors such as *surface roughness* can disturb the flow. In experiments, a laminar pipe flow for  $Re \approx 100.000$  has been maintained by avoiding such disturbances and using a (very) smooth pipe [20, p. 270]. Particularly, the intermediate regime can be tricky to characterize which is why simulations in the transitional regime for a setup with deformations are interesting.

The Reynolds number is based on the average velocity,  $u_{avg}$ , a characteristic length,  $L_0$ , and the kinematic viscosity,  $\nu$ . It expresses the ratio between *inertial* and *viscous* forces and is as mentioned dimensionless. Others like to think of the number as a ratio,  $\tau_v/\tau_a$ , of two timescales; one for advection,  $\tau_a = L_0/u_{avg}$ , and one for viscous forces,  $\tau_v = L_0^2/\nu$ .

$$Re = \frac{u_{avg} L_0}{\nu} \quad (4.7)$$

On a side note:  $L_0$  for pipes is the diameter,  $D$ . Osborne Reynolds however, actually used the radius (called  $c$ ) in his paper from 1883 when calculating the  $Re$  for glass-pipes. His definition back then was not 'Reynolds number' but simply:  $\frac{c\rho U}{\mu}$  [41, p. 938].

#### 4.4.2 Simplified setups

##### Stokes' problems

A particularly well-suited simulation setup for initial investigations is planar flow which due to symmetry only has one non-vanishing velocity component, e.g.  $u_x(y, t)$ . The NS-equations will in the absence of pressure and body forces reduce to a simple diffusion equation:

$$\frac{\partial u_x}{\partial t} = \nu \frac{\partial^2 u_x}{\partial y^2} \quad (4.8)$$

In particular, two problems related to the setup above are famous and known as *Stokes' first and second problem*. They form an excellent starting point for verifying a fluid solver which we have done for our self-written solver. Interested readers can consult sec. A.3.3.

##### Newtonian fluids

All fluids dealt with in this work are Newtonian which means there is a linear proportionality between the velocity gradient and the related stress. The material constant describing this relationship is of course the dynamic viscosity,  $\mu$ , and tells how much friction is related to neighboring layers moving at different speeds. Three everyday relevant values are

$\mu_{air} = 1.82E - 5 [Pa\ s]$ ,  $\mu_{water} = 1.00E - 3 [Pa\ s]$  and  $\mu_{blood} = 2.7E - 3 [Pa\ s]$  [20, tab. 15.1] and [20, p. 270] (blood is obviously not Newtonian!).

### Newton's law of viscosity

This dependency is encapsulated in *Newton's law of viscosity* often stated as  $\sigma_{ij}(x_j) = \mu \frac{du_i(x_j)}{dx_j}$ , which is geometry-dependent (planar flow) and is really a simplification of eq. 4.4b.

A simple test to check if we are actually simulating a Newtonian fluid is to simulate planar flow to 'infinite' time. Thus, the LHS of eq. 4.8 will vanish which means we expect the linear form:  $u_x(y) = A \cdot y + B$ . Remember, Newton's law of viscosity enters into the RHS of eq. 4.8 through the effective force density:

$$f_i^* = (1/\rho) \nabla_y \sigma_{xy} = \nu \frac{\partial^2 u_x}{\partial y^2}$$

and thus, the linear form is derived. Fortunately, our results from the self-written solver back up this expectancy and a linearly decreasing velocity field is seen in figure 4.1.



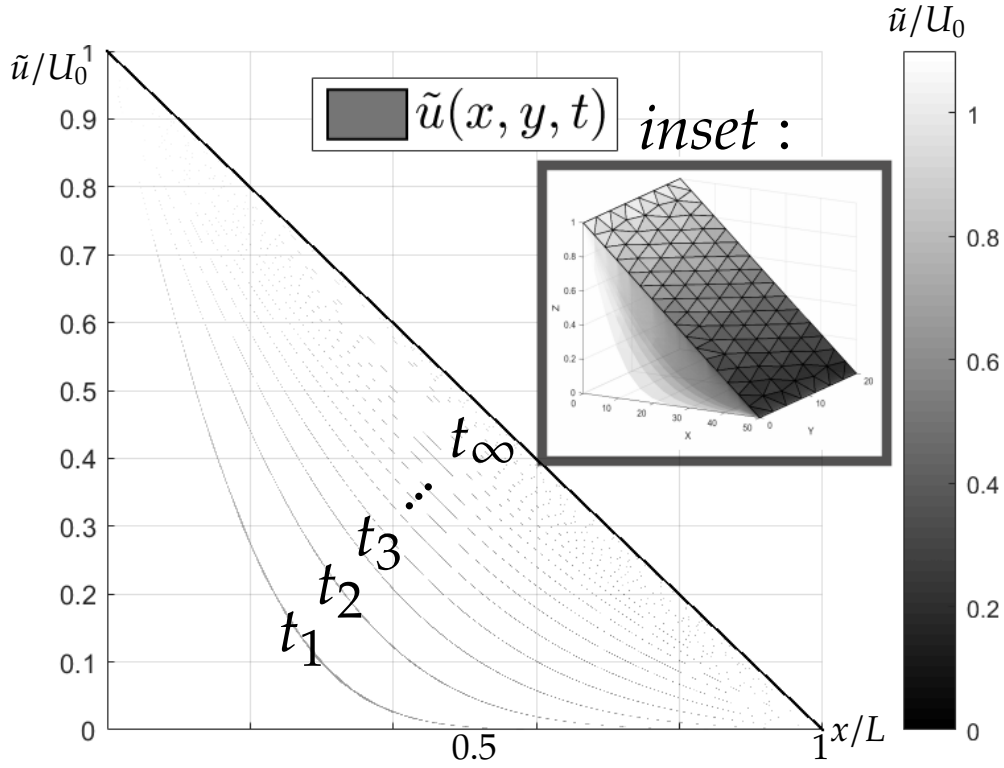



Figure 4.1: A simulation of the simplified version of the NS-equations, i.e. the diffusion equation, also used in Stokes' problems. As time goes toward infinity the simulation reaches a steady state where the stress is distributed throughout the fluid. The inset shows the simulation setup from an elevated angle. It is in essence a study of shear forces between infinite parallel plates: one at  $x = 0$  with constant speed,  $U_0$ , and one at  $L = 50$  not moving.

## DUCTS AND PIPES

ow that we have introduced the NS-equations that govern the physics we will turn to the particular setup of relevance for this work, which is pipes (circular cross-sectional area) and especially ducts (square cross-sectional area).

Usual focal points for flow in ducts and pipes are *friction* and its relation to *pressure loss*,  $\Delta P_L$ , and *head loss* over the length of the duct. In turn, one can then study the *pumping power requirement*, i.e. how much power must be provided to drive the flow. This is particularly relevant for our setup as it is how our simulations work: we pump energy into the system to maintain the turbulent motion of the fluid. The obvious question now is: How much energy should we pump into the system? This is particularly easy to determine for laminar flow and a very useful step before making the calculations in the turbulent regime. In the hope of keeping this part (II) somewhat concise a rather large chunk is presented in the appendix A.1 which interested readers are advised to consult. In it, one will find key concepts explained, such as: *pressure loss*, *head loss*, *pumping power*, *friction factors* and the *implicit Colebrook equation*.

Included in the following is: (5.1.1) An explanation of the Entrance Region to introduce the concept of different *layers* in the fluid followed by (5.1.2) a presentation of the analytical (laminar) base flow called Hagen-Poiseuille. Subsequently sections 5.2 and 5.3 introduce the transitional and turbulent regime respectively. The three main flow regimes can for pipes loosely be associated with the Reynolds numbers [20, p. 270]:

- Laminar;  $Re \leq 2300$
- Transitional;  $2300 \leq Re \leq 4000$
- Turbulent;  $4000 \leq Re$



## 5.1 Laminar flow regime

By far the easiest flow type to handle is the laminar flow. Once a transient NS-solver has been built there are several things one can investigate and compare to analytically known solutions to ensure that the solver is implemented in a proper way. As mentioned in part [IV](#) we will do just this by building and testing a segregated solver using Chorin's splitting method. Alternatively, we could have studied the entrance region to see the gradually increasing boundary layer and to compare the associated entry length to the theoretical value. However, since we will be dealing with periodic boundaries the effect of the entrance region is not of primary importance. Therefore, we have instead chosen to reproduce transient u-profiles of the Hagen-Poiseuille flow as a means of 'exact' recovery of the analytical solution.

It is on the other hand a very simple way to introduce one aspect of the boundary layer - which is central to this work - and therefore the topic is merited. Thus, the well-known study of the entrance region will only be touched upon briefly before we introduce the Hagen-Poiseuille flow for ducts and pipes.

### 5.1.1 Entrance region

A fluid with uniform velocity (plug flow) that enters a pipe will suddenly start to transition to a steady-state velocity profile for flow within pipes which is parabolic. This is due to the no-slip Boundary Condition (BC) at the wall which will stop all fluid particles in the outer layer touching the wall. Also, the adjacent layers will slow down due to the friction between layers in the fluid. This results in a velocity profile which is zero at the walls and maximum at the pipe center. We will actually see an increase in speed at the center since the *conservation of mass* demands a constant mass flow rate, *iii*. More precisely, it can be shown that the average velocity of the pipe flow is half the maximum. Setting up the simulation with imposed velocity inflow,  $u_{inlet} = 1$  m/s (giving an average flow rate of ditto) in a pipe with  $r = 2$  m and  $L = 3$  m with the kinematic viscosity,  $\nu = 0.2$  m<sup>2</sup>/s (much thicker than honey!) results in the plot in [figure 5.1](#). Luckily, we see that the maximum velocity reaches 2 m/s which is twice the average flow rate. The viscous shearing forces that shape the

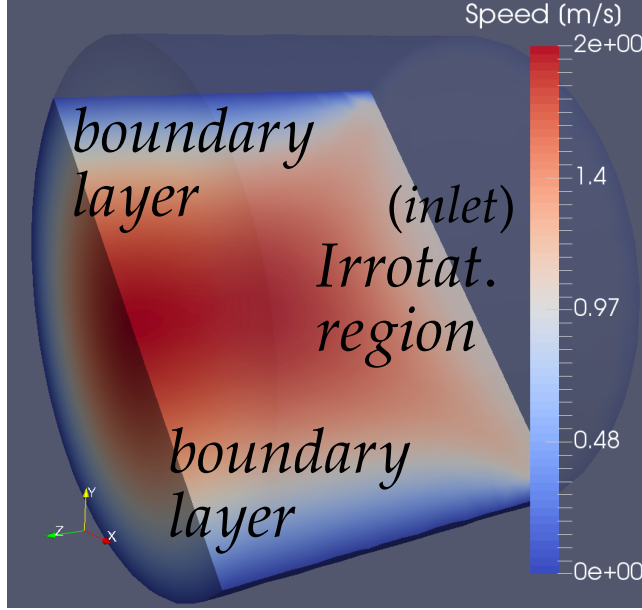


Figure 5.1: A simulation of the entrance region for pipe flow. The pipe has a radius of 2 m and a length of 3 m. The viscosity for the simulation was  $\nu = 0.2 \text{ m}^2/\text{s}$ . Thus, using eq. 5.2 we can estimate the entry length to  $L_{h,laminar} \cong (1/16) \cdot \frac{(1 \text{ m/s} \cdot 2 \text{ m})}{(0.2 \text{ m}^2/\text{s})} \cdot 4 \text{ m} = 2.5 \text{ m}$  which is two thirds of the shown length. By first glance the approximation does not seem to be far off.

profile from uniform to parabolic are present in the region known as the *boundary layer*. The other region also visible in figure 5.1 is the *irrotational flow region* where viscous forces are negligible. As seen, the boundary layer, despite first being negligible, increases in the flow direction. Once it reaches the center the flow is in a steady state. The region up until that point is called the *hydrodynamic entrance region*. Figure 5.1 is in other words a snapshot of exactly this region. The parameter defining the length of such an entrance region is called the *hydrodynamic entry length*,  $L_h$ , and defines the region within which the velocity profile is still developing. Usually, temperature is also considered in these discussions but since we will refrain from including this in our simulations we will also omit temperature considerations in the present part.

### Hydrodynamic entrance length

From the solution to Stokes' problems (sec. A.3.3) one can identify a *momentum diffusion front* [20, eq. 15.14]:

$$\delta_{front} = 2 \sqrt{\nu t} \quad (5.1)$$

which we can use to estimate the time it takes for the momentum to diffuse from pipe wall to center:  $t = r^2/(4\nu)$ . Multiplying this time with the average speed gives a pipe length beyond which we should have a steady state:  $L_{h,laminar} = u_{avg} \cdot t = u_{avg} \cdot (r^2/(4\nu)) = Re \cdot (1/16)D$ . Thus, to

determine the length of the hydrodynamic entrance region one can use the approximation [20, eq. 16.39]:

$$L_{h,laminar} \cong \frac{1}{16} Re D \quad (5.2)$$

The pipe has a radius of 2 m and a length of 3 m. The viscosity for the simulation was set to  $\nu = 0.2 \text{ m}^2/\text{s}$  giving the estimate:

$$L_{h,laminar} \cong (1/16) \cdot ((1 \text{ m/s} \cdot 2 \text{ m}) / (0.2 \text{ m}^2/\text{s})) \cdot 4 \text{ m} = 2.5 \text{ m}$$

This is a reasonable guess judging by the figure. One can also compute the entry length for turbulent flow which will generally result in a much shorter distance since the random fluctuations dominate over molecular diffusion. We shall however not dig further into entrance region related phenomena as it is outside the scope of this thesis but content ourselves with this brief description instead. The lesson here is simply that within a fluid one can identify different *layers*. Studying their interaction will reveal much about the fluid motion.

### 5.1.2 Steady Hagen-Poiseuille flow

The steady, pressure-driven flow through straight conduits is called Hagen-Poiseuille flow. The cross-sectional shape can take various forms, the most common being perhaps circular or rectangular.

The steady Hagen-Poiseuille flow can be derived by considering the forces related to the difference in pressure,  $\Delta p$ , and difference in *shear stress*,  $\tau = -\mu(du(r)/dr)$ , of a differential fluid element [51] (or simply by using symmetry [20, p. 268]). One can write up a force balance on the element to arrive at the momentum equation describing the flow. The resulting velocity profile is [20, eq. 16.29]:

$$u(r) = \frac{\Delta P}{4\mu L}(R^2 - r^2) \quad (5.3)$$

which clearly has a parabolic shape. The above expression was for pipes but a similar expression for other cross-sectional shapes, e.g. a square duct, can be found. Using Fourier series one can obtain the solution [36, eq. 10]:

$$u(x, y) = \frac{\Delta p}{\mu L} \frac{4h^2}{\pi^3} \sum_{n_{\text{odd}}} \frac{1}{n^3} \left( 1 - \frac{\cosh(n\pi x/h)}{\cosh(n\pi \omega/2h)} \right) \sin(n\pi y/h) \quad (5.4)$$

for ducts with width,  $w$ , and height,  $h$ , and a coordinate system defined as  $-w/2 < x < w/2$  and  $0 < y < 1$ .

Needless to say, the  $u$ -profile can be used to obtain almost anything. Perhaps two of the most obvious usages are finding the (wall) shear stress,  $\tau$ , and subsequently the pressure loss,  $\Delta P_L$ . We have chosen to focus on *stress* both for laminar and turbulent flow as it is a key concept. As mentioned, the pressure loss is described in appendix A.1.

### Wall shear stress

Evaluating the shear stress,  $\tau$ , at the wall we obtain the *wall shear stress*,  $\tau_w = -\mu(du/dr)|_R$ . When studying the entrance region as in fig. 5.1, it will be maximal right at the entrance of the pipe since the boundary layer here is very thin yielding a steep slope for the profile. Correspondingly, the drop in pressure will also be largest at the entrance. All along the entrance region  $\tau_w$  will continue to decrease until the steady state is reached.

Shear stress,  $\tau$ , and perhaps in particular wall shear stress,  $\tau_w$ , is instrumental to nearly all flow calculations. Take as an example the *Fanning friction factor*:

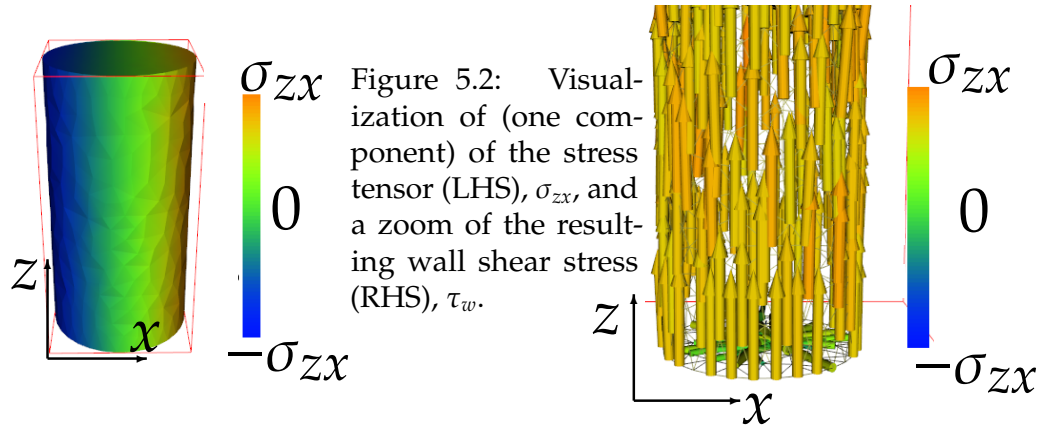
$$C_f = \frac{\tau_w}{(1/2)\rho u_{avg}^2} \quad (5.5)$$

which is valid both in the laminar and turbulent regime. Once  $\tau_w$  is known one can easily get  $C_f$  which in turn gives pressure loss or pumping requirement.

An attempt to visualize the relationship between shear stress,  $\tau$ , in a fluid and the wall shear stress,  $\tau_w$ , is seen in figure 5.2. Once we have  $u$  either from an analytical expression or a simulation we can use eq. 4.4b to get the Newtonian stress tensor. As a stress tensor field it can be difficult to visualize in one plot. In figure 5.2 LHS is seen the  $\sigma_{zx}$  component which is only a part of the entire shear stress,  $\tau$ , for the fluid. Since  $\sigma_{zx} \rightarrow dF_z/dS_x$  shows the force in the  $z$ -direction acting on the surfaces with a normal in the  $x$ -direction we combine all components of  $\sigma$  the acting force,  $F_z$ , as vectors (RHS of figure) which we can identify as the wall shear stress,  $\tau_w$ .

Hagen-Poiseuille flow for various cross-sectional shapes have been studied extensively. One example is N. A. Mortensen, F. Okkels and H. Bruus who use a dimensionless compactness number to evaluate the hydraulic resistance for numerous shapes, e.g. triangular and harmonic-perturbed circles [36] (this study also included FEM). Thus, it is not only circular and square conduits that can be studied - they are simply very convenient setups.

A straightforward extension of the above discussion of steady Hagen-



Poiseuille flow is to introduce the transient analytical solutions. This part is presented in the appendix A.2 (for verification tests see sec. 11 and 12).



## 5.2 Transitional flow regime

This regime is of particular interest for us as it is here we will conduct the comparison with the LBM. This transition is not sudden but fluctuating. Thus we find both laminar and turbulent behavior here.

Since the flow can alternate in behavior so does the friction factor,  $f$ . Data is here very unreliable - making our simulations so much more relevant.

Fluid dynamics is by far easiest to understand in the laminar regime. With transition toward a turbulent behavior comes *eddies* and chaotic behavior which seems futile to try to solve in an exact way. Thus, one makes a number of approximations. For actual, precise results one must conduct *simulations* (or experiments) which of course, is the purpose of the present work.



## 5.3 Turbulent regime

In fully developed turbulent flow there is a strong mixing of fluid particles from adjacent layers. This leads to increased friction since the momentum transfer between the fluid particles increases. The laminar flow



is characterized by highly ordered motion and the momentum transfer across streamlines is due to molecular diffusion. Now, energetic rotating regions called *eddies* dominate the flow. Thus, flow regions hitherto well-compartmentalized and distant are now mixed much more effectively.

One of the characteristics making the turbulent motions so complex to describe is its fluctuating nature. To illustrate this discrepancy compared to laminar profiles one can look at the transitional plot in figure 5.3. Here, we have performed a simulation in a deformed duct. The simulation was initiated from a laminar steady-state Stokes flow profile shown in black and was then driven by a forcing since the BCs were periodic. As we run it for over 100.000 iterations the shear forces accumulate and turbulence sets in. On a side note this corresponds to about one week of actual computation time at NBI (running in parallel on a cluster of 28 processors of type: Intel(R) Xeon(R) CPU E5-2697 v3 2.60GHz). Every thousandth iteration the profile along the line  $[x, y, z] = [x_i, 0.5, 20] \forall x_i = [0, 1]$  is plotted in shades of (increasingly deeper) blue. The most recent profile shown in red clearly has turbulent characteristics. It is flatter, fuller and asymmetric due to fluctuations. Once fully developed turbulence is obtained one can get a smooth turbulent counterpart to the smooth black laminar profile by taking the mean over many iterations. The red profile is one of these many "snapshots" needed to get the smooth (black) curve. The amount of time needed to average over can be taken as the point where the time-average-value is constant.

Had we instead chosen to plot instantaneous velocity,  $u$ , for one point,  $\mathbf{x}_0$ , over time in fully developed turbulence we would easily realize that  $u(\mathbf{x}, t)$  fluctuates around a constant average value,  $\bar{u}$ . It is therefore useful to describe  $u$  as the sum of an average and a fluctuating part [51, eq. 8.35]:

$$u = \bar{u} + u' \quad (5.6)$$

### 5.3.1 Shear stress

Just as we can find the pressure loss,  $\Delta P_L$ , of laminar flow (see app. A.1) we can use the exact same expression for turbulent flows but now the Darcy friction factor must correspond to turbulent flow. The first thing to determine is the shear stress.

We cannot just use  $\bar{u}$  as we used  $u(r)$  in the laminar case and look at the entity:  $\tau = -\mu(d\bar{u}/dr)$  because the shear stress is even larger than this. Consider the turbulent profile in figure 5.3. At the walls  $-\mu(d\bar{u}/dr)$  would yield a lot of stress but further away from the walls we would find a negligible amount of stress. This does not correspond well with the large



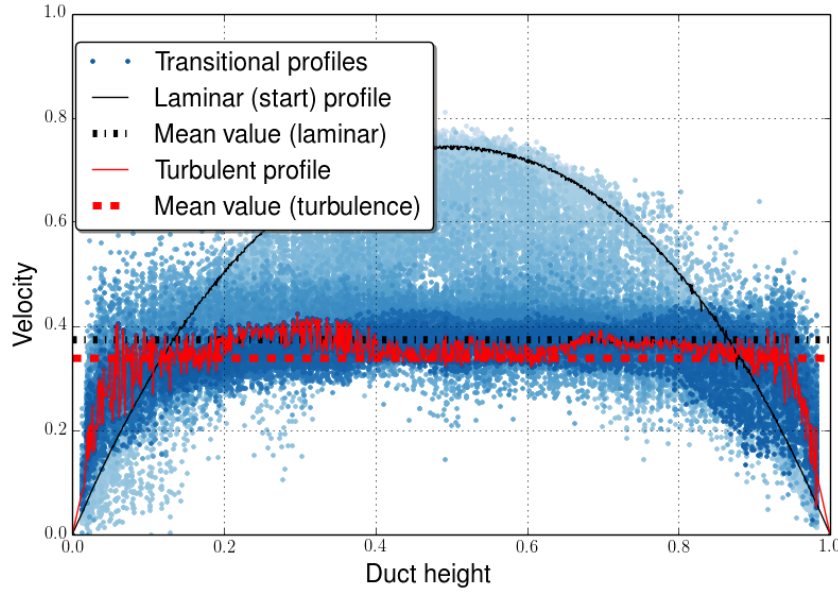


Figure 5.3: The velocity profile transition from laminar to turbulent. Profiles along the line  $[x, y, z] = [x_i, 0.5, 20] \forall x_i = [0, 1]$  of over 100.000 iterations is shown. The simulation was performed in a duct of dimensions  $[1 \times 1 \times 40]$ . The initial profile was found as a laminar Stokes flow steady-state solution (black). Every thousandth iteration has been plotted in shades of blue (increasing in saturation). The final (now turbulent) profile is shown in red. The mean velocity and thereby also the kinetic energy is still falling. Thus, the turbulence is not fully developed yet.

friction taking place due to momentum exchange of fluctuating particles. Instead, the total shear stress for turbulent flows is actually a sum of two parts [51, eq. 8.36]:

$$\tau_{total} = \tau_{laminar} + \tau_{turb} \quad (5.7)$$

There is both shear due to friction between layers,  $\tau_{laminar} = -\mu(d\bar{u}/dr)$ , and the shear due to fluctuating fluid particles,  $\tau_{turb}$ .

### Reynolds stresses

To use eq. 5.7 we need an expression for  $\tau_{turb}$  which can be found by considering a differential layer,  $dA$ , between two adjacent layers. Letting  $u'$  and  $v'$  describe the fluctuations in the average flow direction and the orthogonal direction respectively one can describe the mass flow rate of

fluid particles which due to eddy motion rises through  $dA$  as:

$$\dot{m} = \rho \bar{v}' dA \quad (5.8)$$

The rate of momentum transfer, i.e. the force, from upper to lower layer is then  $\delta F = \rho \bar{v}' dA \cdot (-u')$ . Since stress is force per area we can finally express  $\tau_{turb}$  as [51, eq. 8.37]:

$$\tau_{turb} = -\rho \bar{v}' u' \quad (5.9)$$

Terms such as  $-\rho \bar{v}' u'$  are known as *Reynold stresses* and lie at the heart of nearly all turbulent analyses.

### 5.3.2 Boussinesq hypothesis

#### Eddy viscosity

In the hope of modeling turbulence the simplest step would now be to model the Reynolds stresses as an average velocity gradient. Thus, in 1877 J. Boussinesq introduced an *eddy viscosity*,  $\mu_t$ , to describe  $\tau_{turb}$  and in turn  $\tau_{total}$  as [51, eq. 8.38-39]:

$$\tau_{turb} = \mu_t \frac{\partial \bar{u}}{\partial y} \quad \text{and} \quad \tau_{total} = \rho (\nu + \nu_t) \frac{\partial \bar{u}}{\partial y} \quad (5.10)$$

The turbulence modeling was initiated to achieve closure to the equations of motion. In turn, the modeling (eddy viscosity introduction) depends on closure to the eddy viscosity, without which we have not progressed. The problem boils down to  $\nu_t$  depending on actual flow properties instead of just being a fluid property as its laminar counterpart,  $\nu$ .

The lesson to take home from eq. 5.10 is that now we are not only dealing with simple diffusive problems. To find out which mechanism matters one can look at the viscous and advective timescales when e.g. assessing plots. Here, it will soon be clear that other forces than those due to mere viscous diffusion are at play.

### 5.3.3 Layers of turbulent flow

Just as we saw an interaction between layers in the entrance region of laminar flow there are different layers in turbulent flow. It is common to divide it into four flow regions listed with increasing distance to the boundary wall:

1. Wall layer; a thin layer right at the wall where viscous effects dominate. There are no eddies in this region and the velocity profile

is close to linear. Other names include; viscous sublayer, linear layer and laminar layer.

2. Buffer layer; flow is no longer streamlined since turbulent effects are visible. It is however, still the viscous forces that dominate.
3. Inertial sublayer; turbulence is now present though not yet dominant. The flow is clearly marred by turbulent effects.
4. Turbulent layer; viscous forces are finally surpassed by turbulence.

Considering the different regions it is no wonder that the description of  $u(r)$  is difficult. The answer is *semi-empirical*, in the sense that we have to use both analyses and measurements in our description. Using dimensional analysis one can propose forms for functionals which can then be finally determined by experimentally estimating their parameters. A good example of this procedure is given below for the *Logarithmic law*.

### Friction units

In the wall layer we can express the wall shear stress as [20, eq. 31.29]:

$$\tau_w = \mu \frac{u}{y} \quad (5.11)$$

at the distance,  $y$ , from the wall assuming the velocity profile is linear and thus its gradient is constant. Rearranging the terms and taking the square root we find the quantity  $\sqrt{\tau_w/\rho}$  which has dimensions of velocity. It is known as the *friction velocity* and is a common metric in turbulence analysis. Several symbols have been associated with it but we shall follow the paper presenting the *Oasis-solver* and use  $u_\tau$  [16, p. 186]:

$$u_\tau = \sqrt{\tau_w/\rho} = \sqrt{\nu \frac{u}{y}} \quad (5.12)$$

### Law of the wall

To obtain an expression for  $u$  (in dimensionless form) in the wall layer we simply insert eq. 5.12 in 5.11 and isolate the dimensionless velocity profile,  $u/u_\tau$  [20, eq. 31.32]:

$$\frac{u}{u_\tau} = \frac{yu_\tau}{\nu} \quad (5.13)$$

This is the *law of the wall*. It is sometimes stated in terms of the non-dimensionalized variables  $y^+ = y/\delta_0$  and  $u^+ = u/u_\tau$  (where  $\delta_0 = (\nu/u_\tau)$  is the *viscous length*) simplifying the law to:

$$u^+ = y^+ \quad (5.14)$$

The law of the wall has empirically been observed in the range  $0 \leq y/\delta_0 \leq 5$ . This limit to the wall layer sets the dimensionless *Karman number*,  $Ka$ , which also is commonly expressed as  $Re_\tau$  (friction Reynolds numbers) - for example in the article presenting the Oasis-solver [20, p. 575].

### Channel - setup

How to apply these friction units when setting up a simulation? The solver we will end up using in the comparison of turbulence statistics was presented in 2015 with focus on two applications: 2D Taylor-Green flow (for verification) and Turbulent channel flow (for turbulence comparison) [16]. The turbulence results where compared to the famous spectral DNS results for turbulent channel flow presented by Moser et al. [18]. Both papers use the conventional friction units to set the forcing. Cloning into the Oasis-git one will find the four lines (l. 97-100) in the `Channel.py` source code:

```
# Specify body force
utau = nu * Re_tau
def body_force(**NS_namespace):
    return Constant((utau**2, 0., 0.))
```

that is, the forcing used is:  $f_z = u_\tau^2$ . There is no further explanation as to how they chose this forcing - being trivial for seasoned DNS-practitioners - but for newcomers a derivation can build the physical intuition and is given below.

Using figure 5.4 we write up a force balance for the dark blue region:

$$\Delta p \cdot H^2 = \tau_w \cdot \Delta z \cdot H \cdot 2 \Leftrightarrow \frac{\Delta p}{\Delta z} \approx \nabla p = \frac{2\tau_w}{H} \quad (5.15)$$

The pressure (forcing) applied to the system (LHS) should equal the friction forces (RHS). As seen, we get a factor of 2 due to both an upper and a lower plane-wall. We are interested in the pressure gradient magnitude since we wish to apply a forcing with that exact value. To get the final  $f_x$  value we need two substitutions. First we use eq. 5.12:  $u_\tau^2 = \tau_w$  since  $\rho$  has been set to  $1 \text{ kg/m}^3$ . We then use  $Re_\tau = u_\tau H / (2\nu)$  as stated in Moser et al. [18, p. 1]. Inserting  $u_\tau$  and  $Re_\tau$  gives:

$$\nabla p \approx f_x = \frac{2u_\tau^2}{H} \Leftrightarrow f_x = \frac{2(2\nu Re_\tau / H)^2}{H} = \nu^2 Re_\tau^2 \quad (5.16)$$

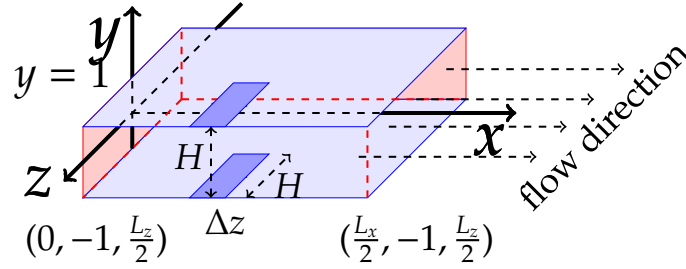


Figure 5.4: Setup for turbulence simulation in a channel using Oasis with no-slip conditions at the wall (light blue) and periodic BCs (red) to create an infinite mesh. The figure does not scale correctly with the mesh used in simulations and therefore various measures of interest are given. The dark blue area can be used to write up the force balance in eq. 5.15.

where we in the last step used that the setup has  $H = 2$  m. The final expression in eq. 5.16 is exactly the expression used for the forcing in the Oasis-source code and thus a satisfactory result.

### Duct - setup

A corresponding derivation in a duct setup is straightforward. Since we have four no-slip walls instead of just two the final expression changes to:

$$f_z = 16 \cdot \nu^2 \cdot Re_\tau^2 \quad (5.17)$$

We have  $H = 1$  m for this setup and write  $f_z$  since the length-axis in the duct setup is the z-axis. From the above we can estimate an approximate  $Re_\tau$  for our simulations once we have decided upon a forcing. Our two primary simulations for LBM comparison have forcing values;  $f_z = 1.2E - 5$  m<sup>2</sup>/s and  $f_z = 1.8E - 5$  m<sup>2</sup>/s which using eq. 5.17 results in  $Re_\tau \approx 96$  and  $Re_\tau \approx 117$  respectively. It is these friction Reynolds numbers we will use to discern our various simulations. The expected corresponding final Reynolds number for the simulations are  $Re = 2400$  and  $Re = 2600$  respectively.

### The logarithmic law

Further away from the wall we cannot hope to use the law of the wall to describe  $u$ . It turns out a straight line further out can be observed as a function of the logarithmic distance,  $\ln(y^+)$ . Thus, we get a *logarithmic law* [20, eq. 31.35]:

$$\frac{u}{u_\tau} = \frac{1}{\kappa} \ln\left(\frac{yu_\tau}{\nu}\right) + B \quad (5.18)$$

It describes the velocity in the inertial sublayer. The buffer layer therefore describes the transition from law-of-the-wall to the logarithmic law. Above,  $\kappa$  is the famous *von Kármán constant* named after the Hungarian-American scientist Theodore von Kármán as a recognition of his contribution to the field [53]. Now,  $\kappa$  and  $B$  can be determined to obtain the law. The constant values are approximately  $\kappa \approx 0.4$  and  $B \approx 5.5$ . [20, p. 576] [34, fig. 25] [33]. The interesting thing in this layer is that there definitely is turbulence in the region, but it is still the friction velocity,  $u_\tau$ , that sets the scale.

### Surface roughness

Irregularities, such as the 4 deformation bumps in the mesh we will use (see sec. 2.1.1), affect the flow in the boundary layer. This is in turn crucial to the rest of the flow. The deciding factor when determining if a surface is *smooth* or *rough* is the height of the roughness elements,  $\eta$ . Should the wall layer completely submerge the elements the surface is deemed smooth and the inner *bulk* of the flow does not feel the roughness [20, p. 275-276]. If, on the other hand, the elements protrude the layer, as the bumps do in our case ( $\eta \approx 0.1 > 0.02 \approx h_{\text{wall layer}}$ ), the flow feels the roughness of the surface. As seen in the appendix (A.8) the surface roughness is therefore also important when calculating the turbulent friction factor. Although outside of the scope of the present work it is worth mentioning that the turbulent regime is actually divided into regimes itself; e.g. *smooth* and *rough* turbulence, for the very reason described above. When  $Re$  increase above  $\approx 100.000$  the wall layer simply becomes negligible and thus all pipes will appear rough. Hence, the name ‘rough turbulence’.

## TURBULENCE THEORY

"A comprehensive theory of turbulence is still the most important outstanding problem in fluid mechanics, even if the basic equations of motion have been known for centuries." [20, p. 402]

- Benny Lautrup, 2011

Professor emeritus in theoretical physics, NBI.

### 6.1 Fully developed turbulence

**W**HAT is turbulence really? In fully developed turbulence we expect (as mentioned) to describe the velocity by a mean component,  $\bar{u}$ , and a fluctuating component,  $u'$ . To build up intuition on the matter of turbulence it can be instructive to briefly sum up some of the work done by the Russian mathematician A. N. Kolmogorov. Although his famous  $(-5/3)$ -power law only can be expected to be found at very high  $Re$  and thus is not applicable during the LBM comparison the results still render an idea of 'what turbulence really is'.

#### 6.1.1 Specific rate of dissipation

One can think of the flow as a sea of eddies; more precisely, an *eddy cascade*. The largest eddy with a size comparable to the setup and the smallest defined by a *dissipation scale*,  $\lambda_d$ . The term 'cascade' signifies that both mass and energy is conserved. The large eddies simply break

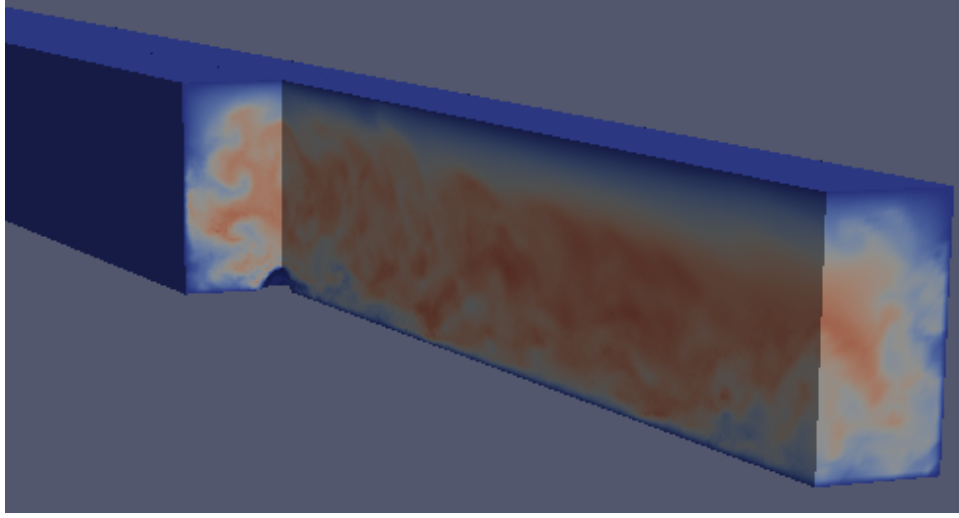


Figure 6.1: Simulation of turbulence in a duct (sliced down the middle). One can among other things notice a deformation bump mentioned previously. At the bump, half a cross section of the duct is included showing visible eddies. This is *not* fully developed turbulence. Rather, it is from the transitional regime with  $Re \approx 2600$ .

up into smaller replicas und thus the energy is transported all the way down to the smallest scale, where it is dissipated. Due to the inevitable dissipation,  $\mathcal{P}$ , we can only hope to observe (sustained) turbulence if we constantly perform work on the system by e.g. a forcing. For the very same reason, determining the average *specific rate of dissipation*,  $\epsilon = \mathcal{P}/M$ , for the pipe is of primary importance. For a cylindrical pipe the computation is straightforward: dissipation is pressure loss times volume flow rate,  $\mathcal{P} = \Delta P_L \cdot \dot{V}$ . Here, the volume flow rate is given as average velocity times cross-sectional area:  $\dot{V} = u_{avg} \cdot A_c$ . The mass is simply density times volume,  $M = \rho \cdot \pi r^2 \cdot L$ . From eq. A.3a and A.3b we get  $\Delta P_L = 2L\tau_w/r$  and we can finally compute  $\epsilon$  [20, eq. 31.1]:

$$\epsilon = \frac{\mathcal{P}}{M} = \frac{(2L\tau_w/r) \cdot (\pi r^2 \cdot u_{avg})}{\rho \pi r^2 L} = \frac{u_{avg}^3}{r} \cdot \left( \frac{\tau_w}{(1/2)\rho u_{avg}^2} \right) \Leftrightarrow \epsilon = \frac{u_{avg}^3}{r} \cdot C_f(Re) \quad (6.1)$$

where we in the last step used eq. A.6 for the dimensionless Fanning friction coefficient.



### 6.1.2 Kolmogorov's theory

The idea of an energy cascade can be traced back to L. F. Richardson (1922). His own wording was: "big whirls have little whirls that feed on their velocity, and little whirls have lesser whirls and so on to viscosity - in the molecular sense" [43, p. 66]. This picture can be used to approximate a characteristic eddy-time,  $\tau_e$ , and velocity variation,  $\delta u$ , by means of an eddy-size,  $\lambda$ . Here,  $\tau_e$  is the time it takes an eddy to break up into smaller replicas.

By considering the *inertial range*:  $\lambda_d \ll \lambda \ll L$ , where  $\lambda$  is much larger than the dissipation scale,  $\lambda_d$ , but simultaneously much smaller than the geometry characteristic length,  $L$ , we can make a certain approximation. First we assume that the velocity variation inside an eddy is,  $\delta u \approx \lambda/\tau_e$ , which means a group of eddies with mass,  $dM$ , will have kinetic energy equal to:  $d\mathcal{T} \approx dM\delta u^2$  and thereby the energy transfer rate between two eddy groups in the cascade will be:  $d\dot{\mathcal{T}} = dM\delta u^2/\tau_e$ . We can now use the specific rate of dissipation in eq. 6.1 to relate the time scale and the length scale. To do this we assume that the viscous dissipation rate equals the energy transfer rate between groups;  $d\mathcal{P} = \epsilon dM = d\dot{\mathcal{T}}$ .

#### Kolmogorov scaling law

The idea here is that if turbulence is homogenous and isotropic and mass and energy truly are conserved quantities on the way down the cascade then the rates must mirror each other. Now, by simple insertion of  $d\dot{\mathcal{T}}$  we get the *Kolmogorov's scaling law* for both  $\tau_e$  and  $\delta u$  [20, eq. 31.3]:

$$\epsilon dM = d\dot{\mathcal{T}} \Leftrightarrow \epsilon dM = dM \delta u^2/\tau_e = dM(\lambda/\tau_e)^2/\tau_e \quad (6.2a)$$

$$\tau_e \approx \epsilon^{-1/3} \lambda^{2/3} \quad \wedge \quad \delta u \approx \frac{\lambda}{\tau_e} = (\epsilon \lambda)^{1/3} \quad (6.2b)$$

#### Kolmogorov's power law

Perhaps most famous of Kolmogorov results is the  $-(5/3)$  power law of the energy spectrum [20, eq. 31.7]:

$$E(k) \approx \epsilon^{2/3} k^{-5/3} \quad (6.3)$$

which is in terms of an inverse length scale,  $k = 2\pi/\lambda$ . One can derive it by considering the specific kinetic energy according to eddy-size:  $d\mathcal{T}/(M d\lambda) = \epsilon^{2/3} \lambda^{(-1/3)}$ . Using the substitution  $d\lambda \approx dk/k$  leads to eq. 6.3.

Again, to truly get the  $-(5/3)$  slope one should be well within the turbulent region.

### Kolmogorov's dissipation scale and time-step estimates

A useful tool when simulating turbulence can be to estimate the number of vertices one would need in a mesh or the maximal  $\Delta t$  one should use if the eddies should be resolved adequately [20, eq. 31.15]. Such an expression for  $\Delta t$  be found by using Kolmogorov's scaling law 6.2b to express the *Kolmogorov dissipation scale*,  $\lambda_d$ . Subsequently one can use the diffusion approximation for time from eq. 5.1 to get the expression for  $\Delta t$ .

To get the characteristic eddy turnover time,  $\tau_e$ , (eq. 6.2b) for the dissipation scale,  $\lambda_d$ , we insert the latter in the former. Should this time equal the characteristic diffusion time:  $\tau_d \approx \lambda_d^2/\nu$  (from eq. 5.1) we get [20, eq. 31.10]:

$$\tau_e = \tau_d \Leftrightarrow \epsilon^{-1/3} \lambda_d^{2/3} = \lambda_d^2/\nu \Leftrightarrow \lambda_d = \epsilon^{-1/4} \nu^{3/4} \quad (6.4)$$

In eq. 6.4 we have the Kolmogorov dissipation scale. Using eq. 5.1 to (roughly) estimate the time it takes for diffusion to travel a system the size of the smallest eddies,  $\tau_d \approx \lambda_d^2/\nu$ , we can find an estimate of what our  $\Delta t$  should be if we hope to resolve all the scales of turbulence:

$$\Delta t \approx \lambda_d^2/\nu = \epsilon^{-1/2} \nu^{1/2} \Leftrightarrow \Delta t = (u_{avg}^3/r)^{-1/2} (\nu)^{1/2} \Leftrightarrow \Delta t = Re^{-3/2} \frac{r^2}{\nu}, \quad (6.5)$$

where we used eq. 6.1 (skipping the friction factor). Thus, one should expect to use a smaller  $\Delta t$  when  $Re$  increases - an intuitively sound conclusion. The estimated  $\Delta t$  for  $Re \approx 2600$  (corresponding to  $Re_\tau = 117$ ) with radius  $r = 0.5$  m is according to eq. 6.5  $\Delta t \approx 0.2$  s.

There have been many objections to the Kolmogorov theory over the years. To name one thing of particular relevance to this work; the theory cannot be utilized in the transitional regime. It does however have numerous experimental verifications of e.g. the energy spectrum behavior and should therefore in general point in the right direction.

## 6.2 Reynolds-averaged Navier-Stokes equations

In the preceding section we presented a decomposition of the  $\mathbf{u}$ -field into a mean and a fluctuating part (eq. 5.6);

$$\mathbf{u} = \bar{\mathbf{u}} + \mathbf{u}'$$

which we then used to approximate turbulent stress between layers (eq. 5.9).

### 6.2.1 Reynolds stress tensor

Another, perhaps more proper way is to simply insert  $\mathbf{u}$  into the NS-equations (eq. 4.4a) [20, eq. 31.19a]:

$$\frac{\partial(\bar{\mathbf{u}} + \mathbf{u}')}{\partial t} + ((\bar{\mathbf{u}} + \mathbf{u}') \cdot \nabla)(\bar{\mathbf{u}} + \mathbf{u}') = -\frac{1}{\rho} \nabla(\bar{p} + p') + \nu \nabla^2(\bar{\mathbf{u}} + \mathbf{u}') + \mathbf{f} \quad (6.6a)$$

$$\nabla \cdot (\bar{\mathbf{u}} + \mathbf{u}') = 0 \quad (6.6b)$$

By taking the mean of the above equations they can be restated as the *Reynolds-Averaged Navier-Stokes* equations (RANS) [20, eq. 31.22]:

$$\frac{\partial \bar{u}_i}{\partial t} + (\bar{u}_j \cdot \partial_j) \bar{u}_i = f_i + \frac{1}{\rho} \partial_j \left[ -\bar{p} \delta_{ij} + \mu (\partial_j \bar{u}_i + \partial_i \bar{u}_j) - \rho \overline{u'_i u'_j} \right] \quad (6.7a)$$

$$\partial_j \cdot \bar{u}_i = 0 \quad (6.7b)$$

These equations are also known as the *mean field equations* or simply *Reynolds equations*. Noticeable, is the term  $-\rho \overline{u'_i u'_j}$  which stems from the convective term and expresses a coupling between mean fields and the fluctuations. We have chosen Einstein notation below to get the exact expression in 5.9 for the Reynolds stresses. The notation has changed slightly;  $u'_x, u'_y$  and  $u'_z$  are interchangeable with  $u', v'$  and  $w'$  in this case. If the reader consults textbooks it will often be the  $u'_x, u'_y$  and  $u'_z$  notation one will find - whereas many FEM practitioners (including the Oasis-solver developers) use  $u', v'$  and  $w'$  for Reynolds stresses. Actually, Reynolds himself used  $u, v$  and  $w$  in his historical paper from 1894 where he wanted to follow up on his 1883-paper introducing what is now known as the Reynolds number. One can even date the decomposition of the velocity field in a mean and a fluctuating component back to the 1894 paper. See e.g. eq. 11 on p. 164 in Ref. [42]. Correspondingly, one will find references to *Reynolds decomposition* in present turbulence literature [6, p. 1]. Above, we also notice that the Newtonian stress tensor from eq. 4.4b has now changed. The stress field appearing in eq. 6.7a is known as the *Reynolds stress tensor*:

$$\sigma_{ij} = -\bar{p} \delta_{ij} + \mu (\partial_j \bar{u}_i + \partial_i \bar{u}_j) - \rho \overline{u'_i u'_j},$$

hence, the name Reynolds stresses. Finally, it is worth mentioning that the set of equations in eqs. 6.7a to 6.7b is only for the mean part,  $\bar{\mathbf{u}}$ . There

is another set of equations for the fluctuating part,  $\mathbf{u}'$ . We thus have *two* sets of equations coupled to each other. Now perhaps, the *Closure problem* is more clear. Since the mean field equations eq. 6.7a to 6.7b would form a closed entity if  $-\overline{\rho u'_i u'_j}$  was known, much turbulent theory goes into modeling this exact term.

## ONSET OF TURBULENCE

**T**HE overall scope of the thesis is to contribute to the rapidly developing research field of ‘Onset of Turbulence’. Within the last decade or so numerous important contributions have been published which continuously increase our insight into the onset and nature of turbulence [6, 12, 13, 14, 27, 45, 38, 5, 39]. The aim of this chapter is to give a brief overview of recent advances within the research field.

### 7.1 Historical overview

The study of the onset of turbulence can be dated back to the days of O. Reynolds [41, 42]. As stated in [5] the onset of turbulence in shear flows is not as well studied a topic as ‘fully developed turbulence’ but in recent years much has happened. Results from new experiments and DNSs are now published regularly, and numerous reviews have been compiled in order to systematize our knowledge, easily reaching in the hundreds of Refs. Some of these reviews state that their bibliography “is far from exhaustive at least by one order of magnitude” [27]. This is all to say, the following is by all means an excerpt from a much more vast literature.

Reynolds himself seemed to know that turbulence occurred beyond a critical value of a parameter ( $Re$ ). Now, it is custom to operate with at least two characteristic values: a threshold below which we have a global stability,  $Re_g$ , and a threshold above which we observe uniform, featureless turbulence,  $Re_t$  [27]. Below  $Re_g$  we will always return to the base flow, i.e. Hagen-Poiseuille flow for ducts and pipes regardless of the

perturbation. In between these two  $Re$ -thresholds we have the transitional regime where laminar and turbulent flow coexist. In this range, domains of laminar flow and domains of turbulent flow interact and propagate down the duct, one after the other.

There is yet another threshold,  $Re_c$ , which is the linear stability threshold above which there is unconditional instability [27]. This threshold belongs within the discussion of *subcritical* versus *supercritical* instability dating back to L. Landau in 1944. His theory hinged solely on temporal perturbations but in 1986 Y. Pomeau extended this to time- and space-dependent perturbations. Growth of subcritical instabilities (turbulence) occurs by contamination whereas supercritically instabilities can grow everywhere [39]. Had  $Re_g$  and  $Re_c$  coincided the transition would have been supercritical, *but they do not* for pipe flow [27]. This spatial growth by contamination is exactly what we observe in our data (see figure 7.1) and thus in agreement with Pomeau's extension of the theory. For further insight into the sub/super-critical discussion [38] can be consulted.

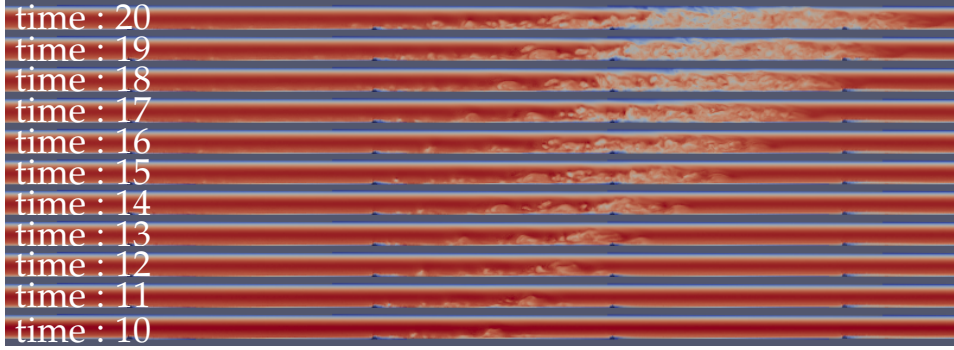


Figure 7.1: A strong slug (the term is defined in sec. 7.2) initiating at a deformation bump and propagating down the duct mesh. Time advances with increasing height of frame position. One can notice that the slug grows by contamination of the neighboring laminar flow. For an explanation of the time units go to subsection 7.1.1.

### 7.1.1 Dimensionless units

This is a small note on units used in turbulence investigations. We follow the same dimensionless units as the recent paper by Song et al. [12]. That means results will be shown in  $D$  for length and  $D/u_{avg}$  for time (*advective time*,  $\tau_a$ ). To be clear, figure 7.1 will be explained:

Every frame of the velocity field is separated by 2000 iterations. The

time-step is  $\Delta t = 0.01$  s which gives  $2000 \cdot 0.01$  s = 20 s. Now, for this simulation we have  $u_{avg} = 0.05$  m/s and the diameter is always  $D = 1$  m which gives the corresponding dimensionless time:  $\frac{20 \text{ s}}{(1 \text{ m})/(0.05 \text{ m/s})} = 1$ . This means that we advance the dimensionless time with 1 for each frame. We start at time= 10 since the first frame is 20.000 iterations after initialization.

### 7.1.2 Directed Percolation

This focus on spatial contamination between domains has led to many models based on Directed Percolation (DP) in the hope of modeling transitional turbulence. DP is often used to model Excitable Media, e.g. epidemics, fires and so on [14, 27]. The principle is simply to have a stochastic process of an active state (turbulence) that interacts with an absorbing state (laminar flow) [45].

Shih et al. used DNS of pipe flow to test an ‘ecological’ DP-model of different flow types (laminar, zonal and turbulent) taken as species that could interact. From the DNS they obtained kinetic energy curves for ‘zonal flow’ ( $\bar{u}_r = 0$ ) and for turbulence. Since the curves both showed oscillatory behavior but were separated by a phase they interpreted them as “activator-inhibitor dynamics” [14]. Based on the DNS results they then came up with a simplistic predator-prey model that described the data.

Why is this related to DP? Well, as a conclusion they examine their model in the asymptotical limit of vanishing prey (turbulence). In this limit their model equation simplifies to the exact equations of the Reaction-Diffusion (RD) model in DP and thus the connection is established. In their own words they have presented an “ecological model of transitional turbulence, which is asymptotically equivalent to DP at the transition” [14]. We will not pursue the matter of RD further in the present work. Interested readers can consult [28, sec. 4.3].

In the same edition of Nature one will find the work by Sano et al. who compared DP to actual experimental results from channel flow [45]. Their setup allowed them to measure the fraction of turbulence present in the fluid. This area fraction was treated as a transition order parameter in DP described by a particular functional [45, p. 251]. Thus, they could fit their data and finally present critical exponents [45, tab. 1] matching very well with the corresponding 2D DP universality class [28].

Be it experimental or simulative comparisons, it seems that based on their ability to predict the measured characteristic lifetime statistics, these models are very effective at capturing the proliferation or relaminarization of turbulent domains [45].

Does this mean that DP is the final answer? "No" according to Y. Pomeau. As he pointed out a few months ago in *Nature* one cannot explain non-turbulent spots within the turbulent (outer) spot [39]. Of course he is but a man like the rest of us - but given that he was awarded the **Boltzmann Medal** (most important prize in Statistical Physics) not one month ago one would probably do well to heed his advice. Why is it trivially obvious for Pomeau that a laminar spot inside an outer turbulent spot cannot be explained by DP? First of all, it is helpful to remember that Sano et al. report a *continuous* transition [45] and in 'directed' percolation (where *time* is the special direction) this would inevitably lead to one final conquering state. However, we find it more intuitive to think of other classes of DP to intuitively grasp the meaning of Pomeau's comment: As Sano et al. note, DP can be used to describe forest fires. DP is also used in the modelling of forest fires (excitable media): In this picture one would never see fresh, healthy trees occur in midst of a fire. In excitable media we operate with a slow *refractory period* (the trees must grow back up) before we can have a new fire-wave. Clearly, a laminar spot (tree) inside the turbulence (fire) would not fit in this picture.

Interestingly, Pomeau has previously lauded DP for its (qualitative) accurate description of *splitting* of turbulent domains [38]. This, perhaps is the essence of the matter: DP might not provide the complete description of the transition to turbulence but can still be a very useful tool - e.g. in the prediction of characteristic lifetime and splitting time for turbulent domains.

### Turbulence lifetime

There is an ensuing closely linked discussion regarding the *Turbulence lifetime*. One can measure the fraction of turbulence,  $f_T(t)$ , taken to be the ratio between turbulent and laminar area. From  $f_T(t)$  a characteristic time,  $\tau_{1/2}$ , can be measured describing the lifetime of a turbulent domain. At least three functionals have been proposed to describe  $\tau_{1/2}$  [5]:

- Divergent:  $\tau \propto (Re_g - Re)^{-1}$
- Exponential:  $\tau \propto \exp(a Re + b)$
- Super exponential:  $\tau \propto \exp[\exp(a Re + b)]$

According to P. Manneville [28] the consensus is now the super exponential - but since we statistically will not obtain enough data to make any qualified guess we will not pursue the topic further. Here, it suffices to mention it since it is related not only to DP but also to Front Dynamics (FD) which is the study of the upstream and downstream interfaces for a turbulent domain.



## 7.2 Front Dynamics

As noted by Barkley et al. [13] in Nature last October; “the front dynamics of turbulent regions and the transition to full turbulence have yet to be explained”. Now, almost a year removed from their article it does indeed still seem to be the case. Despite a very explanatory model from the same authors there is still need for investigation of front dynamics (FD) in order to better grasp the mechanisms dictating the onset of turbulence.

To characterize this onset it is useful to look at the various forms turbulence can take on during the transition. The current understanding of transitional turbulence is that three distinct classes of turbulent domains can be identified by the front dynamics: *puffs*, *weak slugs* and *strong slugs* [13, 12].

### Puffs

The smallest localized domains of turbulence are called puffs and occur right at the onset of (transitional) turbulence,  $Re_g \approx 2040$  for pipes [6]. Importantly, these puffs do not expand but stay in a localized structure and propagate with a speed slightly below the mean velocity. They can however split and thereby increase  $f_T(t)$  [6]. Puffs have been noted to have an upstream edge which is sharply distinguishable and a downstream edge that is ‘fuzzy’ with long streaky eddies [27, 12]. In literature, the puff has been described as “arrow head-shaped” [12] and thus for obvious reasons been compared to Emmon spots [39].

The  $Re_g$ -threshold is an approximate bound.  $Re_g$  marks the flow regime below which the *final* state is laminar - but this can be after a very long transient behavior. Thus, puffs could be seen below  $Re_g$  but will in this case ultimately decay.

One can determine  $Re_g$  by using characteristic timescales. Whichever of the three functionals above describes the characteristic time, the point is that the lifetime depends on the Reynolds number. So does another characteristic time,  $\tau_s$ , describing the splitting of turbulence. When  $Re$  increases the splitting is more likely whereas the decay is less likely. Thus, Avila et al. were able to determine  $Re_g = 2040$  (for pipes) marking the onset of (sustained) turbulence simply by taking  $Re_g$  as the point where the characteristic decay and splitting times are equal [6].

### Slugs

At higher Reynolds numbers puffs deform into entities called weak slugs. They experience a slight seed-up of the downstream front compared to

puffs and thereby become slowly expanding despite still having a velocity below the mean [13]. The edge profiles remain in essence unaltered and the downstream front is still rather fuzzy but it gradually speeds up as  $Re$  increases. Since they are now slowly expanding the asymptotic final state is fully turbulent. The weak slugs can yet again evolve into strong slugs at high enough  $Re$  which have even faster downstream edges (up to  $\approx 1.6 u_{avg}$ ) and upstream edges slowing down to  $\approx 0.6 u_{avg}$ . The downstream edge also finally takes on a clear, well-definable edge for strong slugs mirroring the shape of the upstream edge [12]. This also explains why the weak and strong slugs are called asymmetrically expanding and symmetrically expanding states in the model from Barkley et al. The above (idealized) description is based on ensemble averages and for a given slug the transition should be expected to be marred by fluctuations. As mentioned,  $Re_g$  denotes the onset of (puff) turbulence. For the upper threshold,  $Re_t$ , we will follow P. Manneville [27] who relates  $Re_t$  with "the strongly invasive character of slugs" and thus sets  $Re_t$  to the strong slug threshold. For pipes this value has loosely been set at  $Re_t \approx 2700$  [27] which is based on the two (\*)-values in table 7.1. A table-summary of all  $Re$ -threshold values from the cited literature is given in table 7.1.

Table 7.1: **Overview of results from experiments and simulations in relevant literature:** Values stemming from experiments are assigned an **E** whereas **S** stands for simulation. **O** stands for online material fit results from Ref [13]. Refs are given for each value. Notice for the sake of intuition some values have been added by eyesight from figures which are then referenced. These values are thus not fit results by authors but (very) approximate values read off of figures.

<u>Setup</u>	$Re_g$	$Re_{weak}$	$Re_t$
<u>Pipe</u>	2040 [6] <b>ES</b>	2400 [6] <b>ES</b>	$\approx 2700$ [13](fig.3a)
	1920 [13] <b>O</b>	2300 [35]	2600* [35]
		2250 [13] <b>ES</b>	2800* [1]
			2900 [12]
<u>Duct</u>	1490 [13] <b>O</b>	2030 [13] <b>E</b>	$\approx 2400$ [13] (fig.3a)

### 7.2.1 Edge profiles

Song et al. [12] also include an analysis of the edge profile in their latest paper by considering a weak slug. These slugs are as mentioned slowly expanding by virtue of an upstream edge-speed which is slower than the downstream edge-speed. The latter is in turn slower than the mean velocity. They argue that the strong upstream front is characterized by a production of turbulence that outweighs the dissipation rate (thus the slug *expands* up the stream in a comoving frame of reference). In contrast, the weak downstream front can be seen as a relaminarization. Noticeably, the upstream front is moving into a laminar profile whereas the downstream front (also moving upwards) moves into turbulence (see fig. 7.1). When the downstream front accelerates above the mean speed it starts to move into the laminar velocity profiles to the right (ahead of the slug) and the weak slug transitions to a strong slug that has two strong fronts where turbulence production dominates. How to (at least intuitively) understand the role that is played by the neighboring velocity profile?

An *energy balance equation* can be obtained from the difference between NS-equations to  $(\bar{u}_i + u'_i)$  and to  $(\bar{u}_i)$ . If one afterwards multiplies with  $(\bar{u}_i/2)$ , sums over  $i$  and integrates over the domain,  $V$ , one will get [55, eq. 4.2] [34, eq. 2.33]:

$$\frac{dE(t)}{dt} = - \int_V u'_i u'_j \frac{\partial \bar{u}_i}{\partial x_j} dV - \frac{1}{Re} \int_V \sum_{i,j} \left( \frac{\partial u'_i}{\partial x_j} \right)^2 dV \quad (7.1)$$

The equation above has been written in dimensionless form where the only parameter is  $Re$ . This equation describes the energy balance of a finite (or infinitesimal) disturbance (i.e. the slug) in a flow. The LHS,  $\frac{dE(t)}{dt}$ , tells us whether energy is transferred to the slug or not. In other words it tells us whether energy is extracted from the laminar flow and put into turbulence.

Above we cited two recent Refs but actually this equation was introduced by Reynolds himself [42]. The first term on the RHS describes the energy exchange between laminar profile and the slug. This can 'add' energy to the slug. The second term however is due to dissipation and will always be negative. As seen, for low  $Re$  the term will dominate and tame the turbulence. For higher  $Re$  the relative size between the first and second term on the RHS will determine whether the puff/slug grows or dies out.

Now perhaps it is easier to intuitively grasp why the slug expands better into laminar than turbulent neighboring flow: the first term on the RHS clearly depends on the profile slope,  $\frac{\partial \bar{u}}{\partial x_j}$ , of the mean flow. We know that the laminar slope is steep whereas the turbulent slope can become

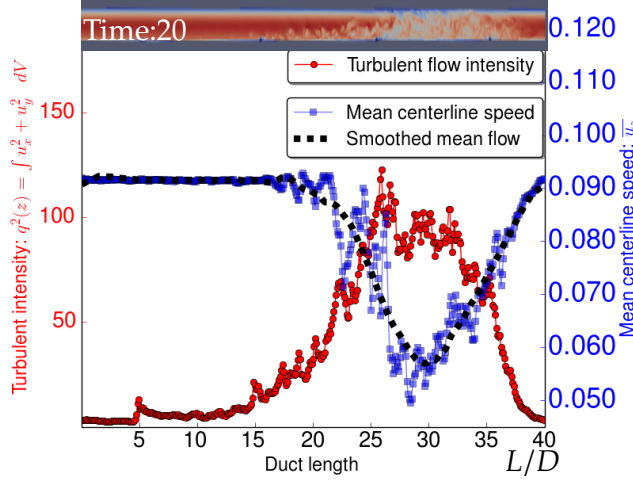


Figure 7.2: Visualization of energy transfer from base flow (laminar profile) to disturbance (turbulence). The curves are simulation results at advection time  $t = 20$  for the  $Re_\tau = 96$  simulation. The corresponding profile is the top frame in figure 7.1. The  $q^2(z)$  metric is first properly introduced in eq. 16.1 but in short it is an indicator for turbulence.

almost flat. This idea is clearly congruent with the description of weak and strong front differences.

As an attempt to visualize the energy flux from one flow phase (laminar) to the other (turbulent) one can observe figure 7.2 showing data from our final simulations. We clearly see that the two curves mirror each other: Center velocity (and thereby energy) is sucked from the region around  $L/D = 30$  and the turbulence intensifies simultaneously in that region. Checking the actual velocity profile shown in the inset above the figure we get our analysis confirmed: localized turbulence is clearly seen around  $L/D = 30$ .

Figure 7.2 is from the strong slug regime and at given time-step the slug is small enough to be confused with a puff. To better see the edge profiles we choose a later frame when the strong slug almost fills the whole duct as shown in figure 7.3. Again, there is a good agreement between curves and the corresponding velocity profile in the inset (7.3(a)) For this simulation the Reynolds number is well within the strong slug regime and we should expect to see strong slug characteristics. Thankfully, we clearly see that the slug has expanded and that both edges now have distinct peaks sharply indicating a slug-bulk from a neighboring laminar flow.

### 7.2.2 Current FD status

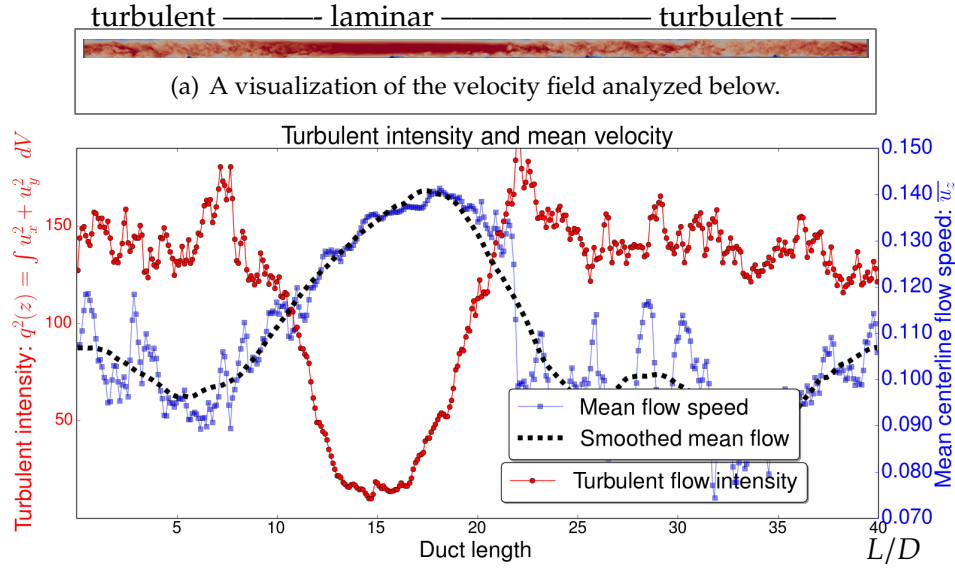


Figure 7.3: Comparison of an indicator for turbulent (red) and laminar (blue) flow (laminar profiles have a higher centerline velocity than turbulence). Above is given the velocity field for the duct at the same time-step. In the terminology of FD we observe a strong slug almost filling the entire duct. The  $q^2(z)$  metric is first properly introduced in eq. 16.1 but in short it is an indicator for turbulence.

There is as mentioned in the beginning of the FD description still need for further investigation into this topic. For instance, one thing the predictive model by Barkley et al. fails to describe is that *front-fluctuations can occur*. Thus, a system can change between weak and strong slug appearance. Furthermore, laminar intermittency is observed for  $Re_c < Re < 3000$  [13] and even up to  $Re \approx 3500$  in Ref [12]. As should be clear by now, the onset of turbulence is clearly an elusive study fraught with difficulties. In light of the above-mentioned difficulties both stemming from stochastic fluctuation it is no surprise that both Song et al. and Barkley et al. [13, 12] present an extensive amount of DNSs with around 20 runs per Reynolds number investigated. In comparison we have a single run per Reynolds number. Such a luxury of data can in part be attributed to the simple meshes used which allow them to use a spectral computational method. Keep in mind we use a *deformed* duct and employ FEM to counter the complex geometry.

As a conclusive remark on the FD discussion it is of interest that for the

FD-model by Barkley et al. an “explicit derivation of these functions from the NS-equations has yet to be achieved” [13]. Thus, we have come full circle and are back to the closure problem described in section 6.2.1. Either one will find it difficult to close the set of governing equations (RANS) or once one has finally found a descriptive model one cannot (explicitly) relate it back to the NS-equations.

### Differences in setup

A first noteworthy point to be made of the setup is that all the listed sources in this section only use pipe mesh (Ref. [13] does have duct-data from experiment but *only* simulations on a pipe mesh). The available data for ducts seem in general less exhaustive than for pipes making the present work the more relevant. Another important difference is that we have a deformed mesh as mentioned above where on the other hand the meshes used in the Refs. are smooth.

It is important to distinguish the various perturbations present in the Refs. Some interpolate a perturbed profile onto the mesh to then see it evolve. In our case we have chosen to interpolate a steady Stokes flow profile onto the mesh and then let the deformation bumps rip up the boundary layer. In the paper from Avila et al. [6], two distinct groups of perturbations are mentioned: *impulsive perturbations* and *continuous perturbations*. They use both simulations (spectral) and experiments and can apply both types of perturbations in their experiment, either by injecting a small jet of water (impulsive) into a long glass pipe or by holding a wire across the flow direction for as long as desired (continuous).

In the mesh-setup for the present work we only deal with continuous perturbations in the form of deformation bumps. Such a bump is visible in figure 7.4 as a small half-sphere.

We will in part V investigate the nature of slugs as most work seem to have been done on puffs up until now. Can we recognize the same characteristics for the slugs? We will focus on the regime  $2400 \lesssim Re \lesssim 2700$ . The referenced work above from Avila et al. focuses on the regime  $Re \lesssim 2300$  [6] whereas the more recent Song et al. cover  $1900 < Re < 5500$  for pipe mesh [12]. As such, the present work is intended as an extension of work already done within the area and results will hopefully align with and contribute to our current understanding of the transitional turbulent regime.

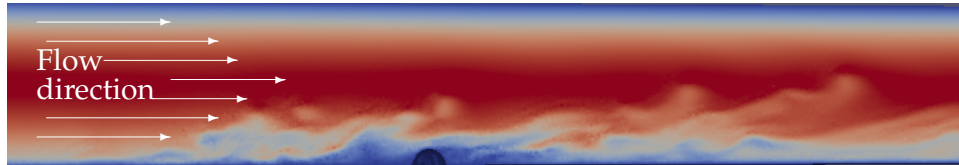



Figure 7.4: Simulation of localized turbulence in a duct (sliced down the middle). One can among other things notice a deformation bump mentioned previously. This is a snapshot taken at the onset of turbulence in our  $Re_\tau = 117$  simulation. For transitional turbulence the puff surrounding the bump may further downstream proliferate or die out. More detailed analyses will follow in part V.

## Part III

### THE FINITE ELEMENT METHOD



## COMPUTATIONAL METHOD

INCE the Finite Element Method (FEM) has been the main technique employed to investigate and approximately solve partial differential equations (PDE)s in the present work we will here give a short introduction to the FEM. First static examples will be given, after which we introduce transient analysis before finally discussing how to incorporate the pressure term to arrive at an actual NS-solver. Having read several FEM introductions recently, it is my personal experience that although easily stated, the theory of FEM is not straightforward to implement without illustrations and examples along the way. It is thus the purpose of this chapter to give even novice readers some tools to, if not completely solve, then schematically list the steps needed in order to actually solve a given problem. The following chapter (9) presents the final fluid solver built from the ground up. Readers more versed in the arts of approximation techniques can thus choose to leaf through the first chapter of this part. The following is by no means an exhaustive description and includes ideas from references [9, 54, 2, 44, 21, 15, 11, 32] which can be consulted for further reading.

### 8.1 Finite Element Method

The FEM is one of many ways to solve PDEs. Unlike e.g. the Finite Difference Method (FDM) where we approximate the equation itself, the FEM approximates the solution within the given domain. Other solution techniques include Finite Volume Method (FVM) and spectral

methods (SM). The latter is based on the Fourier Transform where one writes the solution as a sum of *basis functions* in a Fourier series and then simply determines the optimal coefficients that best approximate the solution. This is very similar to the FEM. The main difference is that FEM is based on a concept of "divide-and-conquer" where one splits the problem domain up into many small subproblems called *finite elements*. Where the SM basis functions stretch out over the entire domain the FEM basis functions are only non-zero within their own finite element. Thus SM has a *global* approach whereas FEM has a *local* one. Each approach has its own advantages and disadvantages. In general SM is very powerful for high-order derivatives as long as the problem at hand has a smooth solution. The challenge for SM arises with increasing complexity of the geometry which on the other hand is where FEM excels.

We will be using the Galerkin method of weighted residuals to show how the stiffness matrix can be assembled. We will therefore start by briefly introducing the method of weighted residuals.

### 8.1.1 Method of weighted residuals

Our starting point will be an arbitrary differential governing equation that we want to find an (approximate) solution to. The basic idea for the method of weighted residuals is to write up the weighted residual from the governing equation posed in the problem. This allows us to find the element matrix equation and subsequently the assembled system matrix equation we want to solve.

If  $f(\mathbf{x})$  is a known function and  $L$  is some linear differential operator we can state the problem of finding the unknown  $u(\mathbf{x})$  as:

$$L[u(\mathbf{x})] = f(\mathbf{x}) \quad : \quad \mathbf{x} \in \Omega \quad (8.1a)$$

$$u(\mathbf{x}) = u_0(\mathbf{x}) \quad : \quad \mathbf{x} \in \Gamma_e \quad (8.1b)$$

$$\frac{\partial u(\mathbf{x})}{\partial n} = q_0(\mathbf{x}) \quad : \quad \mathbf{x} \in \Gamma_n \quad (8.1c)$$

Along with our model equation 8.1a we also give the problem's BCs in eq. 8.1b and 8.1c where  $n$  is the outward normal unit vector to the boundary. We seek a solution that both satisfies eq. 8.1a on the domain,  $\Omega$ , as well as the BCs on the boundary,  $\Gamma$ . The border or boundary,  $\Gamma$ , is the combined entity of the *essential*,  $\Gamma_e$ , and *natural*,  $\Gamma_n$ , boundaries:  $\Gamma = \Gamma_e \cup \Gamma_n$  where we also have an empty intersection;  $\Gamma_e \cap \Gamma_n = \emptyset$ . To be specific a general

1-dimensional working example is chosen:

$$a \frac{d^2 u}{dx^2} + b \frac{du}{dx} + cu = f(x) \quad : \quad 0 < x < L \quad (8.2a)$$

$$u(x) = 0 \quad : \quad x = 0 \wedge x = L. \quad (8.2b)$$

As is evident above, the BCs are often classified as pertaining to one of two categories,  $\Gamma_e$  and  $\Gamma_n$ ; essential BCs are also called Dirichlet conditions whereas the natural conditions are called Neumann or flux conditions since integration of eq. 8.1c along an element border gives the flux of  $u$  entering or leaving the element.

Once the problem has been stated the next step is to choose a *trial function*,  $\tilde{u}$ , defined by a number of unknown coefficients that we want to determine. Here, we choose:

$$\tilde{u}(x) = a_1 x(L - x). \quad (8.3)$$

Our trial function can take any form. The only restriction is that  $\tilde{u}$  must satisfy the BCs. However, the accuracy naturally depends on the selected trial function.

Having chosen the trial function we now substitute it into the problem eq. 8.2a to get the *Residual*:

$$R = a \frac{d^2 \tilde{u}}{dx^2} + b \frac{d\tilde{u}}{dx} + c\tilde{u} - f(x). \quad (8.4)$$

Since the trial function  $\tilde{u}$  is different from the exact solution to our problem the residual will be non-vanishing within the domain. Therefore, we want to determine the coefficients for  $\tilde{u}$  such that the trial function is as close to the exact solution as possible. We introduce a weighting function called the *test function*,  $w$ , to calculate the weighted average and set it to zero:

$$I = \int_0^L w R dx \Leftrightarrow I = \int_0^L w \left[ a \frac{d^2 \tilde{u}}{dx^2} + b \frac{d\tilde{u}}{dx} + c\tilde{u} - f(x) \right] dx = 0. \quad (8.5)$$

Now it is simply a matter of choosing a test function. Once inserted into the integral the coefficient  $a_1$  for  $\tilde{u}$  can be determined.

There are various methods for selecting a test function, e.g. *Collocation* or *Least Square* method (see [54, tab. 2.1.2]). In the Galerkin method the test function is found using:

$$w_i = \frac{\partial \tilde{u}}{\partial a_i} \quad : \quad i = 1, 2, \dots, n \quad (8.6)$$

We have  $n$  test functions as we need  $n$  equations of eq. 8.5 to determine the  $n$  coefficients in the trial function,  $\tilde{u}$ . In the problem at hand the test function only has one coefficient,  $a_1$ , leaving only one equation to solve. As an example, for the settings;  $a = 1$ ,  $b = -3$ ,  $c = 2$ ,  $f(x) = 1$  and  $L = 1$  the residual takes the form:

$$R = \frac{d^2 \tilde{u}}{dx^2} - 3 \frac{d\tilde{u}}{dx} + 2\tilde{u} - 1 \quad (8.7)$$

and the coefficient can subsequently be found to be  $a_1 = -0.6250$ .

Having found the optimal coefficient we can now write our approximate solution along with the exact solution:

$$\tilde{u}(x) = -0.6250 \cdot x(1 - x) \quad (8.8)$$

$$u(x) = \left(\frac{1}{2 \cdot e}\right) \cdot e^{2x} - \left(\frac{1}{2 \cdot e}\right) \cdot e^x + \frac{1}{2}. \quad (8.9)$$

The comparison to the exact solution is seen in figure 8.1(a). By simple insertion it can easily be verified that the exact solution does indeed solve our governing equation. In the following we will see how to obtain the FEM version (fig. 8.1(b)) of an approximate solution.

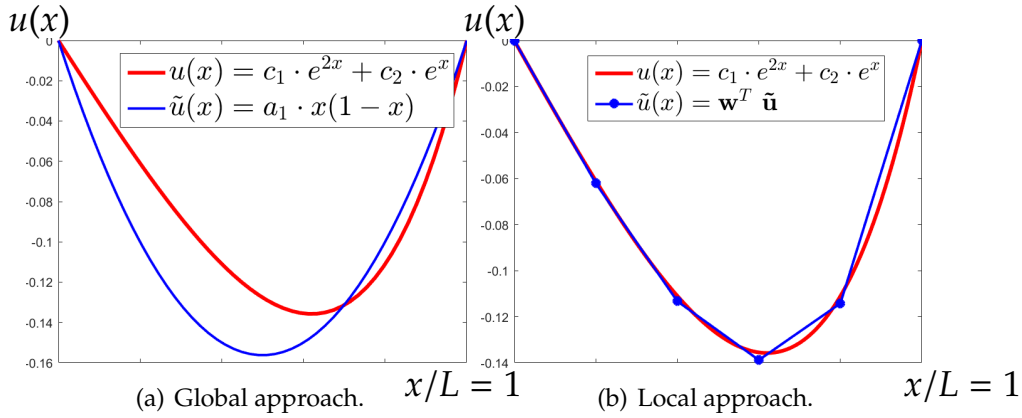


Figure 8.1: Global vs. Local approach in computational methods. LHS: Visualization of exact,  $u(x)$ , and approximate solution,  $\tilde{u}(x)$ , to the problem in eq. 8.2a. Problem settings are:  $a = 1$ ,  $b = -3$ ,  $c = 2$ ,  $f(x) = 1$  and  $L = 1$  and the functions  $u(x)$  and  $\tilde{u}(x)$  are listed in eq. 8.9 and 8.8. RHS: FEM result using 5 elements for problem eq. 8.18. The exact solution,  $u(x)$ , is seen in eq. 8.9.

As is evident from figure 8.1(a) the assumed trial function is incapable of completely emulating the exact solution to the problem. In general, one

can improve the approximate solution by adding terms, i.e. coefficients  $a_i$ , to our trial function. Choosing  $\tilde{u}(x) = a_1 \cdot x(1 - x) + a_2 \cdot x^2(1 - x)$  we obtain two test functions from eq. 8.6 which yield two weighted residual equations to be solved and so on. The principle is unaltered.

### 8.1.2 Weak formulation

The form of our working example in eq. 8.2a is called the *Strong formulation* of our problem. In this version we have a second-order derivative of  $\tilde{u}$  which should have a non-zero finite value for our approximation to work. Thus,  $\tilde{u}$  should be of at least second order. It is however possible to state the problem differently in what is known as the *Weak formulation*. To this end we employ integration by parts on the highest-order term in eq. 8.5 (leaving out the constant  $a$ ) so the second-order term can be stated as:

$$\int_0^L w \left[ \frac{d^2 \tilde{u}}{dx^2} \right] dx = - \int_0^L \frac{dw}{dx} \frac{d\tilde{u}}{dx} dx + \left[ w \frac{d\tilde{u}}{dx} \right]_0^L \quad (8.10)$$

It is clear from the RHS of eq. 8.10 that we have lowered the requirements on our trial function by restating the problem in the weak formulation which is crucial should we decide to use piecewise linear trial functions.

### 8.1.3 Piecewise Linearity

As mentioned in the introduction to the FEM section the SM has a global approach and thus global basis functions,  $\phi_i(x)$ , whereas the FEM uses local basis functions that are non-zero only in the neighboring elements to a nodal point and zero everywhere else. In 1-dimension the basis functions,  $\phi_i(x)$ , take the form:

$$\phi_i(x) = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}} & : x \in [x_{i-1}, x_i] \\ \frac{x_{i+1}-x}{x_{i+1}-x_i} & : x \in [x_i, x_{i+1}] \\ 0 & : otherwise \end{cases} \quad (8.11)$$

Thus we can create a linear piecewise trial function:

$$\tilde{u}(x) = \sum_i a_i \cdot \phi_i(x) + b \quad (8.12)$$

A minimal example is shown in figure 8.2(a) where we have solved an unspecified problem and obtained the coefficients:  $a_1 = 1.4$ ,  $a_2 = 1.3$ ,

$a_3 = 1.1$  and  $b = 0$ . From the figure it is obvious why the weak formulation is crucial. We simply would not have gotten a reasonable result with the strong formulation using a piecewise linear trial function. Also shown in the figure (RHS) is the generalization of piecewise linearity to 2D.

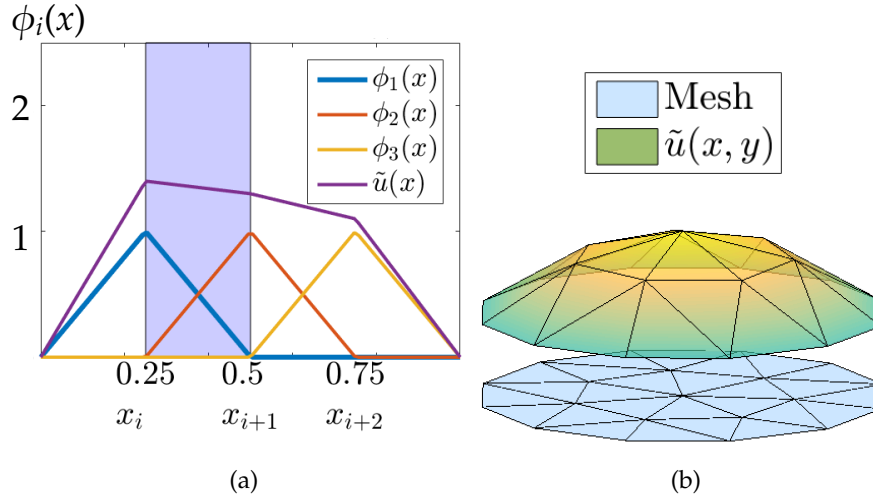


Figure 8.2: Piecewise linearity in 1D and 2D. LHS: A linear piecewise trialfunction  $\tilde{u}(x)$  and its basis functions  $\phi_i(x)$ . The functions are listed in eq. 8.11 and 8.8. RHS: A piecewise linear solution in 2D above its domain mesh.

### 8.1.4 Finite Element Formulation in 1D

To arrive at a matrix equation to systematically compute the weighted residuals it is useful to look at a single element. In 1D this is obviously a line segment. We want to describe the trial function,  $\tilde{u}$ , over the  $i$ 'th element marked as a shaded area in figure 8.2(a). The  $i$ 'th element is defined by the two *nodal variables*  $u_i$  and  $u_{i+1}$  and assuming the trial function can be put on the form:

$$\tilde{u}(x) = c_1 \cdot x + c_2 \quad \text{with} \quad c_1 = \frac{u_{i+1} - u_i}{x_{i+1} - x_i} \quad \wedge \quad c_2 = u_{i+1} + \frac{u_i - u_{i+1}}{x_{i+1} - x_i} \cdot x_{i+1} \quad (8.13)$$

we can rewrite  $\tilde{u}(x)$  in terms of its nodal variables. This is done by inserting  $x_i$  and  $x_{i+1}$  respectively [54, eq. 2.4.6]:

$$\tilde{u}(x) = \left( \frac{x_{i+1} - x}{x_{i+1} - x_i} \right) \cdot u_i + \left( \frac{x - x_i}{x_{i+1} - x_i} \right) \cdot u_{i+1} \Leftrightarrow \tilde{u}(x) = H_1(x) \cdot u_i + H_2(x) \cdot u_{i+1}. \quad (8.14)$$

The  $H(x)$  functions in the result are called *linear shape functions* and are the simplest version of FEM where we assume a linear finite solution over each element. Shape functions have the properties:

$$H_i(x_j) = \begin{cases} 1 & : i = j \\ 0 & : i \neq j \end{cases} \quad \text{and} \quad \sum_i H_i(x) = 1 \quad : x \in \Omega \quad (8.15)$$

As seen in figure 8.2(a) the shape functions simply consist of the basis functions within the given element. Put in another way; even though we only solve for the nodal values and thereby only know the approximate solution at the vertices, the shape functions allow us to interpolate the function over the elements so that we have an approximate solution on the entire domain. One interesting note about eq. 8.15: The fact that  $H_i(x)$  is zero on all other vertices than  $x_i$  which goes back to the definition of  $\phi_i(x)$  results in sparse matrices as we will shortly see. This is crucial for the computation time.

### 8.1.5 FEM example in 1D

We are now finally ready to write up the FEM formulation with the Galerkin method to obtain the (already presented) solution in figure 8.1(b). Since the trial function is expressed as in eq. 8.14 we find our test function,  $w_i$  (using eq. 8.6):  $w_i = \frac{\partial \tilde{u}}{\partial u_i}$ , only now the constant in front is called  $u_i$ . The test and trial function become:

$$w_i = \frac{\partial \tilde{u}}{\partial u_i} = H_i(x) \quad : i = 1, 2 \quad (8.16)$$

$$\tilde{u}(x) = \begin{bmatrix} H_1(x) & H_2(x) \end{bmatrix} \cdot \begin{bmatrix} u_i \\ u_{i+1} \end{bmatrix} \Leftrightarrow \tilde{u}(x) = \mathbf{w}^T \cdot \tilde{\mathbf{u}}. \quad (8.17)$$

We go back to the previous working example with settings (re)stated below:

$$\frac{d^2 u}{dx^2} - 3 \frac{du}{dx} + 2u = 1 \quad : \quad 0 < x < 1. \quad (8.18)$$

Having already dealt with the weak formulation of the second-order term the weak formulation of the entire equation becomes:

$$\int_0^1 \left( -\frac{dw}{dx} \frac{d\tilde{u}}{dx} - 3w \frac{d\tilde{u}}{dx} + 2w\tilde{u} \right) dx = \int_0^1 w dx - \left[ w \frac{d\tilde{u}}{dx} \right]_0^1. \quad (8.19)$$

Using the results from the analysis of the  $i$ 'th element we insert  $w_i$  and  $\tilde{u}$  in the LHS of eq. 8.19 to obtain the matrix equation for the  $i$ 'th element:

$$\underline{\underline{\mathbf{K}_e}} \underline{\tilde{\mathbf{u}}_e} = \int_{x_i}^{x_{i+1}} \left( -\begin{bmatrix} H'_1 \\ H'_2 \end{bmatrix} \begin{bmatrix} H'_1 & H'_2 \end{bmatrix} - 3 \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} \begin{bmatrix} H'_1 & H'_2 \end{bmatrix} + 2 \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} \begin{bmatrix} H_1 & H_2 \end{bmatrix} \right) dx \begin{bmatrix} u_i \\ u_{i+1} \end{bmatrix} \quad (8.20)$$

The RHS of eq. 8.19 can be found similarly and results in the element vector,  $\mathbf{f}_e$ . Thus we end up with the matrix equation:  $\underline{\underline{\mathbf{K}_e}} \underline{\tilde{\mathbf{u}}_e} = \mathbf{f}_e$  which after assembly results in the *system* matrix equation [54, eq. 3.4.1]:

$$\underline{\underline{\mathbf{K}}} \underline{\tilde{\mathbf{u}}} = \mathbf{f}. \quad (8.21)$$

Interestingly, it is visible from the RHS of eq. 8.19 how the Neumann conditions enter into our system equations. They are simply added to the vector for nodes on the boundary,  $\Gamma_n$ . Once the system matrix and vector have been assembled we solve the set of equations:  $\underline{\tilde{\mathbf{u}}} = \underline{\underline{\mathbf{K}}}^{-1} \mathbf{f}$ . The resulting nodal variables, called *primary variables*, can then be interpolated to make our solution span the entire problem domain,  $\Omega$ . The FEM result,  $\tilde{u}(x) = \mathbf{w}^T \underline{\tilde{\mathbf{u}}}$ , was as mentioned shown in figure 8.1(b) along with the exact solution  $u(x)$  from eq. 8.9. We have now obtained a piecewise linear approximate solution with the Galerkin Finite Element formulation.

## 8.2 FEM generalization for higher dimensions

A generalization to higher dimension is conceptually straightforward once the FEM element discretization has been chosen. The present work has dealt mainly with 2D (for self-written FEM code) which is also where we will now focus. In its simplest form using linear shape functions we seek a piecewise 2D continuous function as was visualized in figure 8.2(b).

Two common introductory examples of field governing equations are Laplace's and Poisson's equations of which we shall be using the latter. The PDE along with a complete set of BCs make up an entity called a



Boundary-value problem [19, p. 5]:

$$\frac{\partial^2 u(x, y)}{\partial x^2} + \frac{\partial^2 u(x, y)}{\partial y^2} = g(x, y) \quad : (x, y) \in \Omega \quad (8.22a)$$

$$u(x, y) = u_0 \quad : (x, y) \in \Gamma_e \quad (8.22b)$$

$$\frac{\partial u(x, y)}{\partial n} = q_0 \quad : (x, y) \in \Gamma_n. \quad (8.22c)$$

### 8.2.1 Problem formulation

To solve the Poisson equation we apply the exact same procedure as in 1D: After having introduced a test function  $w(\mathbf{x})$  where  $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$ , we integrate over the entire domain,  $\Omega$ , while demanding that it is true for any  $w(\mathbf{x})$ . Thus we arrive at the variational formulation:

$$\int_{\Omega} w(\mathbf{x}) \left( \frac{\partial^2 u(\mathbf{x})}{\partial x^2} + \frac{\partial^2 u(\mathbf{x})}{\partial y^2} \right) d\Omega = \int_{\Omega} w(\mathbf{x}) g(\mathbf{x}) d\Omega \quad (8.23)$$

As was also done in 1D we perform integration by parts to obtain the weak formulation:

$$- \int_{\Omega} \left( \frac{\partial w(\mathbf{x})}{\partial x} \frac{\partial u(\mathbf{x})}{\partial x} + \frac{\partial w(\mathbf{x})}{\partial y} \frac{\partial u(\mathbf{x})}{\partial y} \right) d\Omega + \oint_{\Gamma_n} w(\mathbf{x}) \frac{\partial u(\mathbf{x})}{\partial n} d\Gamma = \int_{\Omega} w(\mathbf{x}) g(\mathbf{x}) d\Omega \quad (8.24)$$

The weak formulation is our final problem statement. Once inserted in the weighted residual equation (e.g. eq. 8.5) we can obtain the element matrix equation and subsequently after assembly the system matrix equation we want to solve. Now comes the *discretization* of the domain and problem equation after which it is simply a matter of putting it into the computer.

### 8.2.2 Discretization

To discretize the domain one must first decide on an element type. There exists a variety of popular choices, the simplest being a line segment, triangle and a tetrahedron for respectively 1D, 2D and 3D. Examples are seen in figure 8.3(a).

To discretize our equation we again start by assuming that  $\tilde{u}(\mathbf{x})$  can be put on the simple linear form:  $\tilde{u}(\mathbf{x}) = a_1 + a_2 \cdot x + a_3 \cdot y$ . Once found, we interpolate the nodal variables with 2D basis functions so our solution spans the entire domain. An example of 2D basis functions is given in figure 8.3(b).

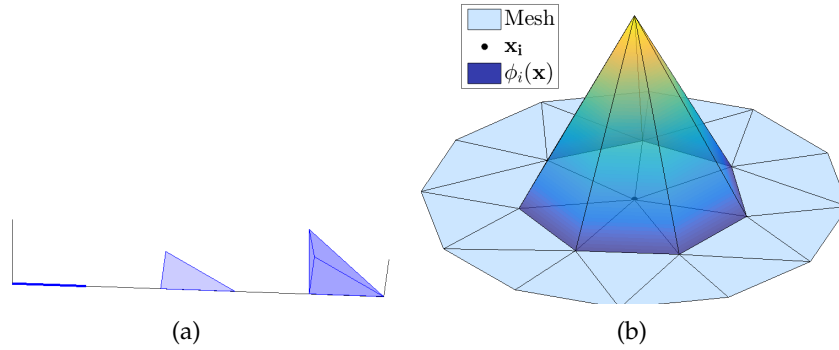


Figure 8.3: LHS: The three standard element types for 1D, 2D and 3D: Line segment, Triangle and Tetrahedron. RHS: An example of 2D basis function,  $\phi_i(\mathbf{x})$ .

Just as in 1D we want to express  $\tilde{u}$  in terms of our nodal variables. To this end, we write  $\tilde{u}(\mathbf{x})$  as:

$$\tilde{u}(\mathbf{x}) = \begin{bmatrix} 1 & x & y \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}. \quad (8.25)$$

We can then describe  $\tilde{u}(\mathbf{x})$  within a given element using three nodal variables, one for each vertex constituting the element. This can be put in a matrix equation of the form:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}. \quad (8.26)$$

Now it is simply a matter of inverting the matrix to isolate the unknown constants which can then be inserted back into eq. 8.25. The resulting expression for  $\tilde{u}(\mathbf{x})$  is:

$$\tilde{u}(\mathbf{x}) = H_1(\mathbf{x}) \cdot u_1 + H_2(\mathbf{x}) \cdot u_2 + H_3(\mathbf{x}) \cdot u_3. \quad (8.27)$$

Defining  $\tilde{u}(\mathbf{x})$  on nodal values is known as a *vertex-centered* scheme [32, p. 3]. The 2D shape functions in front of each nodal variable have properties

$H_i(x_j, y_j) = \delta_{ij}$  and  $\sum_i H_i(\mathbf{x}) = 1$  and take the form [54, eq. 5.2.7-9]:

$$\begin{aligned} H_1(\mathbf{x}) &= \frac{1}{2A} [(x_2 y_3 - x_3 y_2) + (y_2 - y_3) \cdot x + (x_3 - x_2) \cdot y] \\ H_2(\mathbf{x}) &= \frac{1}{2A} [(x_3 y_1 - x_1 y_3) + (y_3 - y_1) \cdot x + (x_1 - x_3) \cdot y] \\ H_3(\mathbf{x}) &= \frac{1}{2A} [(x_1 y_2 - x_2 y_1) + (y_1 - y_2) \cdot x + (x_2 - x_1) \cdot y] \end{aligned} \quad (8.28)$$

The choice of shape function naturally affects the error. Other common shape functions are the ones defining bilinear 4 nodal elements called *Lagrange shape functions*. Higher order shape functions naturally improve the approximation.

From eq. 8.27 it is obvious that the trial and test function can be put on the desired form:

$$\tilde{u}(\mathbf{x}) = \mathbf{w}^T \tilde{\mathbf{u}} \quad \text{and} \quad \mathbf{w} = \begin{bmatrix} H_1(\mathbf{x}) \\ H_2(\mathbf{x}) \\ H_3(\mathbf{x}) \end{bmatrix}. \quad (8.29)$$

Inserting these into the weak formulation in eq. 8.24 yields the element matrix equation and in turn the system equation:

$$\underline{\underline{\mathbf{K}_e}} \tilde{\mathbf{u}} = \mathbf{f}_e \quad \text{and} \quad \underline{\underline{\mathbf{K}}} \tilde{\mathbf{u}} = \mathbf{f} \quad (8.30)$$

$\underline{\underline{\mathbf{K}_e}}$  is called the *stiffness matrix* and arises due to the volume integral on the LHS of eq. 8.24. Its calculation is simple once we have the shape functions leading to the symmetric result [54, eq. 5.2.14]:

$$\underline{\underline{\mathbf{K}_e}} = \frac{1}{4A} \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \quad (8.31)$$

$$\begin{aligned} k_{11} &= (x_3 - x_2)^2 + (y_2 - y_3)^2 \\ k_{12} &= (x_3 - x_2)(x_1 - x_3) + (y_2 - y_3)(y_3 - y_1) \\ k_{13} &= (x_3 - x_2)(x_2 - x_1) + (y_2 - y_3)(y_3 - y_1) \\ k_{22} &= (x_1 - x_3)^2 + (y_3 - y_1)^2 \\ k_{23} &= (x_1 - x_3)(x_2 - x_1) + (y_3 - y_1)(y_1 - y_2) \\ k_{33} &= (x_2 - x_1)^2 + (y_1 - y_2)^2. \end{aligned}$$

The vector,  $\mathbf{f}$ , in eq. 8.30 is a result of the boundary integral and the RHS volume integral in eq. 8.24. The volume integral per element yields a

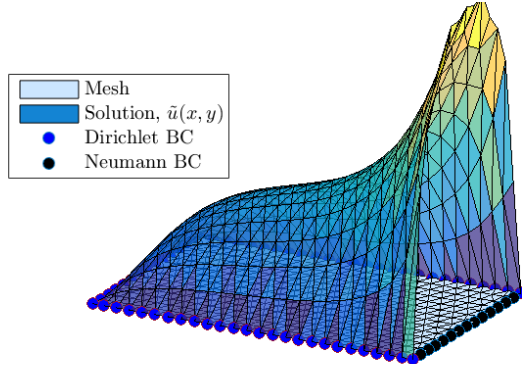


Figure 8.4: The Poisson equation solved with  $g(\mathbf{x}) = 1$ , a positive constant ( $\frac{\partial u(x,y)}{\partial n} = q_0 > 0$ ) for the Neumann boundaries (black dots) while the Dirichlet boundaries (blue dots) are zero-bound. See appendix A.3.1 for further information.

column vector:

$$\int_{\Omega_e} \begin{bmatrix} H_1(\mathbf{x}) \\ H_2(\mathbf{x}) \\ H_3(\mathbf{x}) \end{bmatrix} g(\mathbf{x}) d\Omega, \quad (8.32)$$

which for each element is then assembled into the system equation. The computation obviously depends on  $g(\mathbf{x})$  but usually integration techniques are utilized should the function be too complicated.

The aforementioned boundary integral in eq. 8.24 can also be written as a sum over elements:

$$\oint_{\Gamma_n} w(\mathbf{x}) \frac{\partial u(\mathbf{x})}{\partial n} d\Gamma = \sum_e \int_{\Gamma_n} w(\mathbf{x}) \frac{\partial u(\mathbf{x})}{\partial n} d\Gamma. \quad (8.33)$$

As seen, it is an integral over the natural boundaries summed over all elements affected by the BCs.

We have now explained all three terms in the Poisson equation's weak formulation (eq. 8.24) and are ready to solve the first static example. Interested readers can consult section A.3.1 in the appendix for a specific description. Here, we will simply show the final solution (fig. 8.4) to get a visualization of what kind of function we up until now have described and to get a feeling for the different types of boundaries. The solution is luckily just what one would expect for a positive source ( $g(\mathbf{x}) = 1$ ) and a positive constant ( $\frac{\partial u(x,y)}{\partial n} = q_0 > 0$ ) for the Neumann boundaries (black dots) while the Dirichlet boundaries (blue dots) are zero-bound.

### 8.2.3 Transient analysis

Since the NS-equation has a time derivative of the velocity, we will here briefly introduce various time integration techniques. Basically, we would

like to describe how to go from *steady-state* to *transient* problems:

$$u(x, y) \rightarrow u(x, y, t). \quad (8.34)$$

The simplest working example is the heat equation which has the boundary-value problem:

$$\alpha \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \quad : (x, y) \in \Omega \quad (8.35a)$$

$$u(x, y, t) = u_0 \quad : (x, y) \in \Gamma_e \quad (8.35b)$$

$$\frac{\partial u(x, y, t)}{\partial n} = q_0 \quad : (x, y) \in \Gamma_n, \quad (8.35c)$$

where the constant,  $\alpha$ , is determined from density,  $\rho$ , specific heat capacity,  $c_p$  and the thermal conductivity,  $k$ . From now on it is put to unity,  $\alpha = 1$ .

The weak formulation takes the same form as the Poisson equation except for a new, *transient* term:

$$\int_{\Omega} w(\mathbf{x}) \frac{\partial u(\mathbf{x}, t)}{\partial t} d\Omega + \int_{\Omega} \left( \frac{\partial w(\mathbf{x})}{\partial x} \frac{\partial u(\mathbf{x}, t)}{\partial x} + \frac{\partial w(\mathbf{x})}{\partial y} \frac{\partial u(\mathbf{x}, t)}{\partial y} \right) d\Omega - \oint_{\Gamma} w(\mathbf{x}) \frac{\partial u(\mathbf{x}, t)}{\partial n} d\Gamma = 0 \quad (8.36)$$

If we define  $u(\mathbf{x}, t)$  with shape functions:

$$u(\mathbf{x}, t) = \sum_i^n H_i(\mathbf{x}) \cdot u_i(t) \quad (8.37)$$

we can let the shape functions describe the spatial variation for a given element while the nodal variables describe the temporal variation. Thereby the time derivative can be described as:

$$\dot{u}(\mathbf{x}, t) = \sum_i^n H_i(\mathbf{x}) \cdot \dot{u}_i(t) \quad (8.38)$$

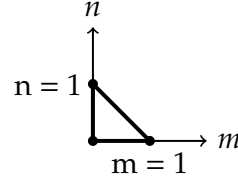
after insertion in the weak formulation we end up with a new matrix from the first term:  $\mathbf{M}_e$ .

### Isoparametric Elements

To actually calculate  $\mathbf{M}_e$  we must evaluate the integral:

$$\int_{\Omega_e} \begin{bmatrix} H_1 \\ H_2 \\ H_3 \end{bmatrix} \begin{bmatrix} H_1 & H_2 & H_3 \end{bmatrix} d\Omega \begin{bmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{u}_3 \end{bmatrix}. \quad (8.39)$$

Figure 8.5: **Isoparametric element:** A triangular element after geometric mapping to the natural coordinate system.



This can be done using a mapping from the *physical* coordinates,  $(x, y)$ , (where the problem is stated) to the *natural* coordinates,  $(m, n)$ . Thus, we express shape functions in natural coordinates [54, eq. 6.3.1]:

$$\begin{aligned} H_1(m, n) &= 1 - m - n \\ H_2(m, n) &= m \\ H_3(m, n) &= n. \end{aligned} \quad (8.40)$$

These shape functions can be used in a *geometric* mapping for  $\mathbf{x}$  as well as for the interpolation of  $u$ :

$$\mathbf{x} = \sum_i H_i(m, n) \cdot \mathbf{x}_i \quad u = \sum_i H_i(m, n) \cdot u_i. \quad (8.41)$$

In other words, we are working with *isoparametric* elements which is very convenient for computing the element matrices. The definition of isoparametric elements is that the geometric mapping and the interpolation of  $u$  must utilize the *same* shape functions. An advantage of isoparametric finite elements is that complex domains and boundaries can be easily handled. The possibly oddly shaped triangle element in question is simply mapped to the triangle seen in figure 8.5 so the integral can be carried out:

$$\begin{aligned} \int_{\Omega_e} \begin{bmatrix} H_1 \\ H_2 \\ H_3 \end{bmatrix} \begin{bmatrix} H_1 & H_2 & H_3 \end{bmatrix} d\Omega_{x,y} \begin{bmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{u}_3 \end{bmatrix} &= \int_0^1 \int_0^{1-m} \begin{bmatrix} H_1 \\ H_2 \\ H_3 \end{bmatrix} \begin{bmatrix} H_1 & H_2 & H_3 \end{bmatrix} |J| dn dm \begin{bmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{u}_3 \end{bmatrix} \\ &= 2 \cdot A_e \int_0^1 \int_0^{1-m} \begin{bmatrix} H_1 \\ H_2 \\ H_3 \end{bmatrix} \begin{bmatrix} H_1 & H_2 & H_3 \end{bmatrix} dn dm \begin{bmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{u}_3 \end{bmatrix}. \end{aligned} \quad (8.42)$$

As seen, when carrying out the substitution we use the *Jacobian* whose determinant is equal to twice the area of the triangle [54, eq. 6.2.11]:

$$J = \begin{bmatrix} \frac{\partial x}{\partial m} & \frac{\partial y}{\partial m} \\ \frac{\partial x}{\partial n} & \frac{\partial y}{\partial n} \end{bmatrix}. \quad (8.43)$$

The resulting matrix,  $\underline{\underline{\mathbf{M}_e}}$ , from the calculation becomes:

$$\underline{\underline{\mathbf{M}_e}} = \frac{A_e}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}, \quad (8.44)$$

where  $A_e$  is the element area. Clearly, isoparametric elements is a very elegant method for dealing with these integrals. Interested readers can consult Ref. [54, chapter 6] for further reading.

The weak formulation can now finally be restated as the system matrix equation:

$$\underline{\underline{\mathbf{M}}} \dot{\mathbf{u}}^t + \underline{\underline{\mathbf{K}}} \mathbf{u}^t = \mathbf{f}^t. \quad (8.45)$$

There exist a multitude of techniques to solve these equations. We will use FDM and focus on the *Forward* and *Backward* difference method as well as the Crank-Nicolson Method.

### Forward Difference Method

In the forward difference method we express the time derivative as:

$$\dot{\mathbf{u}}^t = \frac{\mathbf{u}^{t+\Delta t} - \mathbf{u}^t}{\Delta t}. \quad (8.46)$$

Once inserted into the system equation we can solve for the next time-step values,  $\mathbf{u}^{t+\Delta t}$ , and obtain:

$$\mathbf{u}^{t+\Delta t} = \underline{\underline{\mathbf{M}}}^{-1} \left[ \Delta t \left( \mathbf{f} - \underline{\underline{\mathbf{K}}} \mathbf{u}^t \right) + \underline{\underline{\mathbf{M}}} \mathbf{u}^t \right]. \quad (8.47)$$

To solve the equation we only need initial conditions (ICs),  $\mathbf{u}(\mathbf{x}, 0)$  and BCs which go into the vector,  $\mathbf{f}$  (there is no source term,  $g(\mathbf{x})$ , in our problem).

The forward difference technique is the most common explicit method. It is only conditionally stable which can be remedied by using the backward difference method.

### Backward Difference Method

Backward difference is very similar to forward difference but has a different starting point since the system equation is written at time  $t + \Delta t$ :

$$\underline{\underline{\mathbf{M}}} \dot{\mathbf{u}}^{t+\Delta t} + \underline{\underline{\mathbf{K}}} \mathbf{u}^{t+\Delta t} = \mathbf{f}^{t+\Delta t}. \quad (8.48)$$

After inserting the finite difference one obtains:

$$\left( \underline{\underline{\mathbf{M}}} + \Delta t \underline{\underline{\mathbf{K}}} \right) \dot{\mathbf{u}}^{t+\Delta t} = \Delta t \mathbf{f}^{t+\Delta t} + \underline{\underline{\mathbf{M}}} \mathbf{u}^t, \quad (8.49)$$

and can once again solve for  $\mathbf{u}^{t+\Delta t}$ . As seen, finding the solution involves not only the current state but also the following one. Thus, it is an *implicit* method. The big advantage for the backward difference technique is that it is *unconditionally stable*, though it has the same global order of error,  $O(\Delta t)$ . Stencils for both forward and backward difference can be seen in figure 8.6 to 8.7.

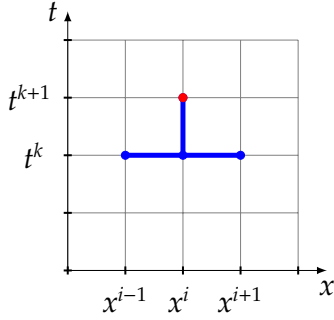


Figure 8.6: Forward difference method.

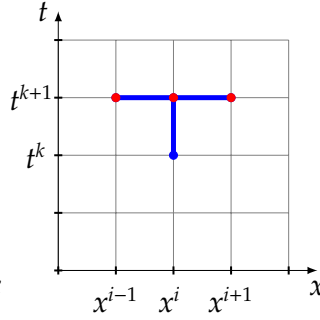


Figure 8.7: Backward difference method.

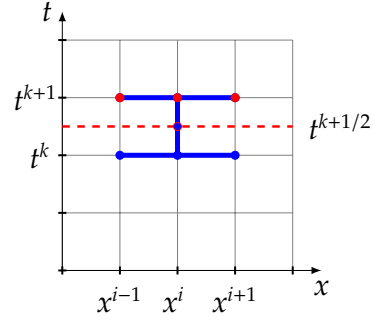


Figure 8.8: Crank-Nicolson method.

Stencils mark the grid points needed to compute the next iteration (red points).

### Crank-Nicolson Method

The third stencil shown in figure 8.8 belongs to the Crank-Nicolson method. Stating the system equation as:

$$\underline{\underline{\mathbf{M}}} \dot{\mathbf{u}}^{t+\frac{\Delta t}{2}} + \underline{\underline{\mathbf{K}}} \mathbf{u}^{t+\frac{\Delta t}{2}} = \mathbf{f}^{t+\frac{\Delta t}{2}} \quad (8.50)$$

and replacing  $\mathbf{u}^{t+\Delta t/2}$  as an average and the time derivative with a *central* difference:

$$\dot{\mathbf{u}}^{t+\frac{\Delta t}{2}} = \frac{\mathbf{u}^{t+\Delta t} - \mathbf{u}^t}{\Delta t}, \quad (8.51)$$

we get yet another system equation where we can solve for the unknown values,  $\mathbf{u}^{t+\Delta t}$ :

$$(2\underline{\underline{\mathbf{M}}} + \Delta t \underline{\underline{\mathbf{K}}}) \dot{\mathbf{u}}^{t+\Delta t} = \Delta t (\mathbf{f}^{t+\Delta t} + \mathbf{f}^t) + (2\underline{\underline{\mathbf{M}}} - \Delta t \underline{\underline{\mathbf{K}}}) \mathbf{u}^t. \quad (8.52)$$

The important thing to note is that the implicit Crank-Nicolson method not only is *unconditionally stable* but also has the global order of error



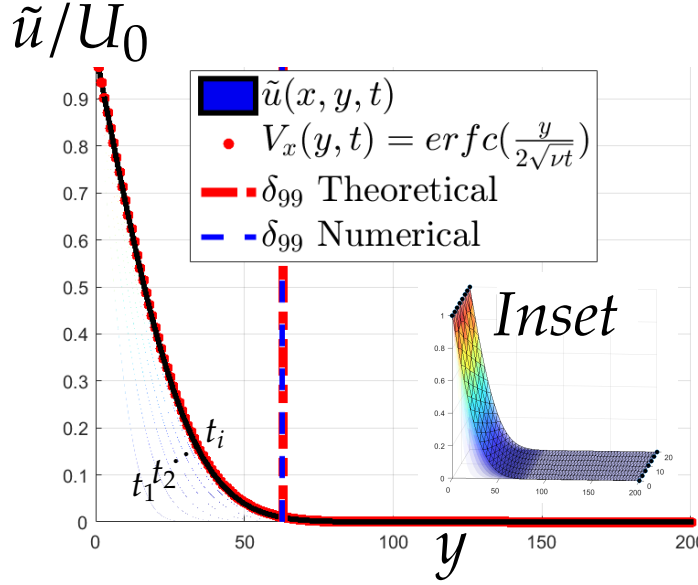


Figure 8.9: FEM solution of Stoke's first problem compared to the exact solution,  $V_x(y, t)$ , from eq. A.28. At  $y = 0$  a plane is at constant speed,  $U_0$ . The fluid is initially at rest. At  $t_i$  a clear agreement between solver and  $V_x(y, t)$  is seen. Also the boundary layer thickness,  $\delta_{99}$ , is well captured. The inset shows the setup from above.

$O(\Delta t^2)$  since the time derivative was replaced with a central difference.

Above, a considerable amount of FEM theory has been briefly listed. It is our clear experience that unless put into practice the FEM theory can remain a bit foggy. Therefore, just as we made an example for static 2D FEM we have made examples for transient simulations. In the appendix interested readers can find (A.3.2) a standard heat equation example solved with the backward difference method and (A.3.2) the classical heat convection problem solved with the Crank-Nicolson scheme where also the Neumann borders depend on  $u$ .

As already alluded to previously, the famous Stokes' problems arising from NS-equations through symmetry-simplification can also be solved with the present state of our solver. These are our first inquisitive investigations of the physics ruling the motion of fluids. Just as for the static example, we will only here show the result (of the first problem: fig. 8.9) to nurture the intuition of the physics at play. The two problems can be found in their entirety in section A.3.3.

## COUPLED SOLVER AND FLUID SIMULATIONS

**I**t is very seldom that the incompressible NS-equations reduce to a simple heat equation as in the two Stokes' problems. The pressure,  $p$ , often times enters into the problem complicating it and thereby demanding a more complete FEM solver. The end product of this chapter will be a fully *coupled* solver with pressure added in the system matrix equation. Once implemented it allows us to simulate Stokes flow in a pipe and eventually, with a few add-ons, the actual NS-equations.

### 9.1 Stokes flow solver

The following is a description of one way to implement such a solver. As for references, the two articles by Misztal et al. have been indispensable as well as CFD-project by fellow student M. Svenningsen [15, 11, 32, 44].

#### 9.1.1 Stokes flow

As a minimal working example it is useful to look at the aforementioned *Stokes flow*. The steady state equations are:

$$\nabla^2 \mathbf{u} - \nabla p = 0 \quad (9.1a)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (9.1b)$$

where we have omitted  $\rho$  and  $\nu$  due to unity rescaling and because body forces have been left out.

### 9.1.2 Problem statement

To choose an intuitive test case we look at the steady state laminar flow through a pipe. The setup can be seen in figure 9.1 but before describing the discretization we must state the problem in its weak formulation. As previously, we introduce the test function,  $w$ , write up the variational form and use integration by parts to obtain the weak formulation:

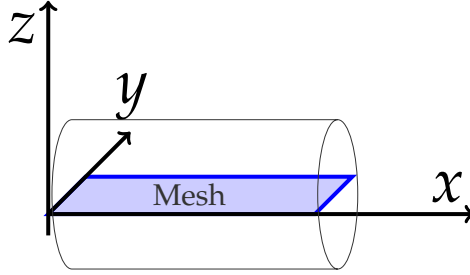


Figure 9.1: FEM setup for Stokes flow in a pipe.

$$\int_{\Omega} w [\nabla^2 \tilde{u} - \nabla \tilde{p}] d\Omega = 0 \Leftrightarrow \quad (9.2)$$

$$\oint_{\Gamma} w \nabla \tilde{u} d\Gamma - \int_{\Omega} \nabla w \nabla \tilde{u} d\Omega - \oint_{\Gamma} w \tilde{p} d\Gamma + \int_{\Omega} (\nabla w) \tilde{p} d\Omega = 0 \Leftrightarrow \quad (9.3)$$

$$\int_{\Omega} \nabla w \nabla \tilde{u} d\Omega - \int_{\Omega} (\nabla w) \tilde{p} d\Omega = 0. \quad (9.4)$$

We have left out the two boundary integrals since we have  $w|_{\Gamma} = 0$ . The first term in eq. 9.4 is simply the *Stiffness* matrix,  $\underline{\underline{K}}$ , that we encountered when solving Poisson's equation or the heat equation. The previous notation:

$$\int_{\Omega} \left( \frac{\partial w(\mathbf{x})}{\partial x} \frac{\partial \tilde{u}(\mathbf{x})}{\partial x} + \frac{\partial w(\mathbf{x})}{\partial y} \frac{\partial \tilde{u}(\mathbf{x})}{\partial y} \right) d\Omega,$$

was mainly for pedagogical reasons but from now on brevity is needed since the complexity increases and we will hence be using the notation of eq.s 9.2-9.4. There is only one minor change compared to previous examples. Up until now, we have only solved for scalar fields, e.g. temperature for the heat equation. Even for the Stokes' problems we only found one value since the velocity field only had one non-zero component due to symmetries. Now, we will solve for general 2D velocity fields,  $\mathbf{u}$ :

$$\mathbf{u} = [u_{1x} \ u_{1y} \ u_{2x} \ u_{2y} \ \dots \ u_{Nx} \ u_{Ny}]^T, \quad (9.5)$$

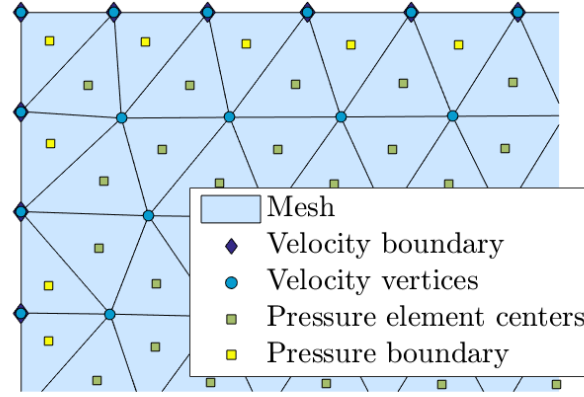


Figure 9.2: Illustration of a staggered grid. Top left corner of the mesh setup for laminar pipe flow. The setup illustration is seen in figure 9.1. The actual mesh showing the BC to be imposed in this problem will be shown later in figure 9.6.

which means that the new size of the stiffness matrix is:  $\text{size}(\underline{\mathbf{K}}) = 2N \times 2N$ , where  $N$  is number of vertices. Thus, the first volume integral can now be written as:

$$\int_{\Omega} \nabla w \nabla \tilde{u} \, d\Omega \Rightarrow \underline{\underline{\mathbf{K}}} \mathbf{u}, \quad (9.6)$$

which is exactly as in the previous examples, just in a larger version.

### Pressure implementation

To write up the system matrix equation caution is needed due to the last term in eq. 9.4. Since we are trying to find a solution for two fields,  $\mathbf{u}$  and  $p$ , at the same time we have to discretize both fields. One of the simplest approaches is discretization by *staggered grids* [20, p. 331] [32, p. 7]. This means that we store  $\mathbf{u}$ -values at the vertices but  $p$ -values at the element centers. Figure 9.2 shows the top left corner of the staggered grid for our test case.

Just as for the velocity field we can write the pressure values,  $p_k$ , for each element in vector form:

$$\mathbf{p} = [p_1 \ p_2 \ \dots \ p_T]^T \quad (9.7)$$

with the size:  $T \times 1$  where  $T$  is number of triangles (elements). The pressure (for this implementation) is assumed *constant over each element*. Thus, we can express the pressure as:  $\tilde{p}(\mathbf{x}) = \xi(\mathbf{x})^T \cdot \mathbf{p}$  where  $\xi(\mathbf{x})$  is pressure shape functions that are constant over a given element [32, p. 7]. The argument for this assumption is treated in depth in Ref. [15, p. 5] and comes down to the fact that we use linear shape functions yielding a constant deformation

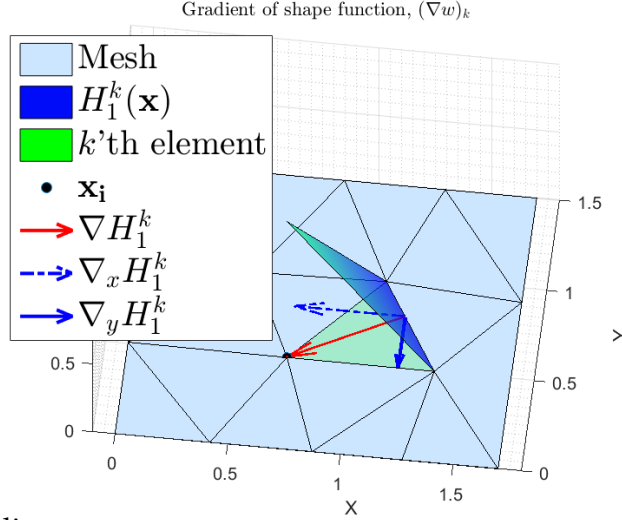


Figure 9.3: The two components from the gradient of the shape function  $H_1^k$  belonging to the  $k$ 'th element. For one element there is a total of 3 shape functions yielding up to 6 non-zero entries.

gradient.

To put the second volume integral in eq. 9.4 in matrix form we split up the integral as a sum of element integrals. It can be beneficial to look at the term  $-\int_{\Omega} (\nabla w)_k \tilde{p} d\Omega$  for just one element to begin with:

$$-\int_{\Omega_k} (\nabla w)_k \tilde{p} d\Omega = -(\nabla w)_k p_k \int_{\Omega_k} d\Omega = -(\nabla w)_k p_k A_k. \quad (9.8)$$

Here we have used the fact that the gradient of our shape functions elementwise is constant and that the pressure likewise is assumed constant over an element.

One would assume by the notation that the assembly of eq. 9.8 would yield a  $T \times T$  matrix, but this is not the case. The term:  $(\nabla w)_k$  is a result of the 3 shape functions related to element  $k$ :  $H_1^k, H_2^k$  and  $H_3^k$ . A gradient of a shape function has two components in 2D whereby we end up with a total of (maximally) 6 non-zero values. This can be expressed in a  $2N \times 1$  vector,  $\mathbf{w}_k$ , with zeros everywhere else. Likewise, we should remember that the pressure in the  $k$ 'th element,  $p_k$ , can be written:  $p_k = \tilde{p}(\mathbf{x}) = \xi(\mathbf{x})^T \cdot \mathbf{p} : \forall \mathbf{x} \in \Omega_k$ . Here,  $\xi(\mathbf{x})$  obviously has the transposed size of  $\mathbf{p}$ , namely  $[1 \times T]$ . It can be useful to visualize components from the term,  $(\nabla w)_k$ . In figure 9.3 one of three shape functions belonging to element  $k$  can be seen along with the two components of its gradient.

To sum up: the element equation 9.8 can be rephrased to:

$$-(\nabla w)_k p_k A_k = -A_k \nabla \mathbf{w}_k(\mathbf{x}) \xi(\mathbf{x})^T \cdot \mathbf{p} : \forall \mathbf{x} \in \Omega_k,$$

where  $\mathbf{w}_k(\mathbf{x})$  is the  $[2N \times 1]$  vector containing all shape functions for an element. The product  $\mathbf{w}_k(\mathbf{x}) \xi(\mathbf{x})^T$  yields a  $[2N \times T]$  size matrix called  $\underline{\mathbf{P}}$  but

since we look at the  $k'$ th element equation we only get the  $k'$ th column,  $\underline{\underline{\mathbf{P}}}_k$  [32, p. 8]. Eq. 9.8 tells us that such a column will at most have 6 non-zero values since all other shape functions,  $H_i$ , than the ones pertaining to the  $k'$ th element will be zero within  $\Omega_k$ . To get the full matrix,  $\underline{\underline{\mathbf{P}}}$ , we simply take the sum of the equation over each of the elements:

$$\sum_{k=1}^T - \int_{\Omega_k} (\nabla w)_k \tilde{p} d\Omega = - \sum_{k=1}^T \underline{\underline{\mathbf{P}}}^k \mathbf{p} = - \underline{\underline{\mathbf{P}}} \mathbf{p} \quad (9.9)$$

The weak formulation in eq. 9.4 can now be expressed in matrix form using eq. 9.6 and 9.9 resulting in:

$$\underline{\underline{\mathbf{K}}} \mathbf{u} - \underline{\underline{\mathbf{P}}} \mathbf{p} = 0 \quad (9.10)$$

### Continuity equation

To obtain the complete system matrix equation we also need to write the continuity equation 9.1b in matrix form. By inserting  $\tilde{u}(\mathbf{x}) = \mathbf{w}^T(\mathbf{x}) \cdot \mathbf{u}$  in the variational form and splitting the integral to a sum over elements we get the matrix equation:

$$\begin{aligned} \int_{\Omega} \nabla \cdot \tilde{u}(\mathbf{x}) d\Omega = 0 &\Leftrightarrow \int_{\Omega} \nabla \cdot \tilde{u}(\mathbf{x}) d\Omega = \int_{\Omega} \nabla \cdot \mathbf{w}^T(\mathbf{x}) \mathbf{u} d\Omega \\ &= \sum_{k=1}^T \int_{\Omega_k} (\nabla w)_k^T \mathbf{u} d\Omega = \sum_{k=1}^T \underline{\underline{\mathbf{P}}}^k \mathbf{u} = \underline{\underline{\mathbf{P}}}^T \mathbf{u} \Leftrightarrow \underline{\underline{\mathbf{P}}}^T \mathbf{u} = 0 \end{aligned} \quad (9.11)$$

Here again, the factor  $(\nabla w)_k^T$  is the gradient of the 3 shape functions for the  $k'$ th element but since the factor is transposed it yields a transposed column (i.e. a row) and in turn a transposed matrix,  $\underline{\underline{\mathbf{P}}}^T$ .

### 9.1.3 Steady Stokes flow system equation

We now have the necessary information to establish the matrix system equation to solve Stokes flow. Using eq. 9.10 and 9.11 we define the system matrix equation:

$$\begin{bmatrix} \underline{\underline{\mathbf{K}}} & \underline{\underline{\mathbf{P}}} \\ \underline{\underline{\mathbf{P}}}^T & \underline{\underline{\mathbf{0}}} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ -\mathbf{p} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (9.12)$$

The system above is the steady state version of the Karush-Kuhn-tucker (KKT) conditions and the LHS matrix will be denoted the KKT matrix from now on [11, p. 8]. Since  $\text{size}(\underline{\mathbf{K}}) = 2N \times 2N$  and  $\text{size}(\underline{\mathbf{P}}) = 2N \times T$  we end up with a system of size:  $\text{size}(\underline{\underline{\mathbf{KKT}}}) = (2N + T) \times (2N + T)$ .

### 9.1.4 KKT Boundary conditions

Even though we have a solvable system of linear equations we still need to implement BCs to solve our problem. Without BCs we end up with the trivial zero-solution. For this problem we will specify the velocity on the mesh-border (Dirichlet BC). We also have to fix one pressure point to give the system a pressure reference. The BCs summary is:

$$\mathbf{u}(\mathbf{x}) = \mathbf{u}(\mathbf{x})_{exact} : \mathbf{x} \in \Gamma \quad \wedge \quad p(\mathbf{x}_0) = p(\mathbf{x}_0)_{exact} : \mathbf{x}_0 \in \Gamma \quad (9.13a)$$

$$\mathbf{u}_{exact} = \frac{\Delta p}{2 \cdot L} \cdot y \cdot (D - y) \hat{\mathbf{x}} \quad \wedge \quad p_{exact} = p_{inlet} - \frac{\Delta p}{L} \cdot x. \quad (9.13b)$$

Here,  $\mathbf{u}_{exact}$  and  $p_{exact}$  are the analytical solutions to the problem seen in eq. 9.13b and  $D$  and  $L$  are the diameter and length of the pipe respectively.

We cannot as previously impose our BCs by inserting identity-rows in the stiffness matrix (this BC-implementation is found in the static 2D FEM example for Poisson's equation in sec. A.3.1). The problem is that the KKT matrix also entails the pressure matrix,  $\underline{\mathbf{P}}$ . We therefore use *Lagrange multipliers* by stating the BCs in matrix form:  $\underline{\underline{\mathbf{C}}} \mathbf{u} = \mathbf{u}_0$  where the  $\mathbf{u}_0$ -values are  $\mathbf{u}_{exact}$  evaluated on the constrained vertices [32, p. 6] [11, p. 10]. The revised system equation becomes:

$$\begin{bmatrix} \underline{\underline{\mathbf{K}}} & \underline{\mathbf{P}} & \underline{\mathbf{C}}^T & \underline{\mathbf{0}} \\ \underline{\mathbf{P}}^T & \underline{\mathbf{0}} & \underline{\mathbf{0}} & \underline{\mathbf{C}}^T \\ \underline{\mathbf{C}} & \underline{\mathbf{0}} & \underline{\mathbf{0}} & \underline{\mathbf{0}} \\ \underline{\mathbf{0}} & \underline{\mathbf{C}} & \underline{\mathbf{0}} & \underline{\mathbf{0}} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ -\mathbf{p} \\ \lambda_u \\ \lambda_p \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{u}_0 \\ \mathbf{p}_0 \end{bmatrix} \quad (9.14)$$

Here, the size of  $\underline{\underline{\mathbf{C}}}$  (defined below) is  $N_c \times (2N + T)$  where  $N_c$  is number of BCs. It contains rows of zeros except for a 1 at the place corresponding to the BC-node. The vectors,  $\lambda$ , are the *Lagrange multipliers* giving name to the method. Obviously, for the present example the vector,  $\mathbf{p}_0$ , contains only one element, namely the pressure reference point.

$$\begin{bmatrix} \underline{\mathbf{C}} & \underline{\mathbf{0}} \\ \underline{\mathbf{0}} & \underline{\mathbf{C}}_p \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{p}_0 \end{bmatrix} \Rightarrow \begin{bmatrix} \underline{\underline{\mathbf{C}}} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{p}_0 \end{bmatrix} \quad (9.15)$$

### 9.1.5 Pseudo-compressibility

As explained in the FEM literature, imposing BCs can lead to *locking* for the chosen element type [11, p. 9] [17, p. 94]. This means that the continuity equation stating zero divergence of  $\mathbf{u}$  is in conflict with the imposed conditions. We therefore introduce *pseudo-compressibility* as suggested by Miztal et al. [11]. Adding the stabilization term,  $-\epsilon \nabla^2 p$ , we get the altered continuity equation:

$$\nabla \cdot \mathbf{u} - \epsilon \nabla^2 p = 0. \quad (9.16)$$

The altered continuity equation will eventually result in the equation:  $\underline{\underline{\mathbf{P}}}^T \mathbf{u} - \epsilon \underline{\underline{\mathbf{S}}} \mathbf{p} = \mathbf{0}$ . The matrix elements in  $\underline{\underline{\mathbf{S}}}$  have slightly different formulas depending on whether we are working in 2D [44, p. 12-13] or 3D [11, p. 10] but the derivation is conceptually the same: First we split the volume integral into a sum of element-integrals and apply the Gauss theorem to reduce the order of derivative where  $\mathbf{n}_{kl}$  is the normal to the  $k$ - $l$ -element-border,  $\Gamma_{kl}$ , and  $\mathbf{n}_{kl}$  is pointing from the  $k$ 'th to the  $l$ 'th element:

$$\int_{\Omega} \nabla^2 p \, d\Omega = \sum_{k=1}^T \sum_{l=1}^T \int_{\Gamma_{kl}} \nabla p \cdot \mathbf{n}_{kl} \, d\Gamma. \quad (9.17)$$

To approximate the gradient we take the difference in pressure and divide it with the distance between the neighboring element-centers:  $\nabla p \cdot \mathbf{n}_{kl} \approx \frac{p_l - p_k}{d_{kl}}$  [11, eq. 20]. Using that the distance,  $d_k$ , from an element center to the interfacing border,  $\Gamma_{kl}$ , is:  $d_k = \frac{2A_k}{3\Gamma_{kl}}$ , we can express  $d_{kl}$  as:

$$d_{kl} = d_k + d_l = \frac{2}{3\Gamma_{kl}} \cdot (A_k + A_l), \quad (9.18)$$

where  $\Gamma_{kl}$  is the border between neighboring triangles  $k$  and  $l$ .

Finally, we can compute the integral, yielding the matrix,  $\underline{\underline{\mathbf{S}}}$ :

$$\begin{aligned} \sum_{k=1}^T \sum_{l=1}^T \int_{\Gamma} \nabla p \cdot \mathbf{n}_{kl} \, d\Gamma &\approx \sum_{k=1}^T \sum_{l=1}^T \frac{p_l - p_k}{d_{kl}} \cdot \Gamma_{kl} = \\ \sum_{k=1}^T \sum_{l=1}^T \Gamma_{kl}^2 \frac{3}{2 \cdot (A_k + A_l)} \cdot (p_l - p_k) &= \sum_{k=1}^T \sum_{l=1}^T \delta_{kl} \cdot (p_l - p_k) \end{aligned} \quad (9.19)$$

$$\underline{\underline{\mathbf{S}}}_{kl} = \begin{cases} \delta_{kl} & : k \neq l \\ -\sum_{m \neq k} \delta_{km} & : k = l, \end{cases} \quad (9.20)$$

with the  $kl$ 'th element of  $\underline{\underline{\mathbf{S}}}$  defined in eq. 9.20. Here, the  $\delta_{kl}$  is  $\delta_{kl} = \Gamma_{kl}^2 \frac{3}{2 \cdot (A_k + A_l)}$  if element  $k$  and  $l$  are neighboring elements. Otherwise:  $\delta_{kl} = 0$ .



What the  $\underline{\underline{\mathbf{S}}}$ -matrix really means is that we allow for a certain degree of area exchange between neighboring elements while we ensure that the total area is constant [11, p. 10]. Notice the  $m$  index in eq. 9.20 for the case:  $k = l$ . Basically, we state in the equation that flux entering neighboring elements of the  $k$ 'th element must equal flux exiting the  $k$ 'th element and vice versa. A visualization of the  $\underline{\underline{\mathbf{S}}}$ -entries is given in figure 9.4. Finally, a curiosity (also noted by the authors [11]): It is interesting that the above actually is a FVM approach where the equation itself is emulated instead of the solution as is typical for FEM.

With the recently defined pseudo-compressibility matrix,  $\underline{\underline{\mathbf{S}}}$ , we can write up the final steady state KKT conditions:

$$\begin{bmatrix} \underline{\underline{\mathbf{K}}} & \underline{\underline{\mathbf{P}}} & \underline{\underline{\mathbf{C}}}^T & \underline{\underline{\mathbf{0}}} \\ \underline{\underline{\mathbf{P}}}^T & \epsilon \underline{\underline{\mathbf{S}}} & \underline{\underline{\mathbf{0}}} & \underline{\underline{\mathbf{C}}}^T \\ \underline{\underline{\mathbf{C}}} & \underline{\underline{\mathbf{0}}} & \underline{\underline{\mathbf{0}}} & \underline{\underline{\mathbf{0}}} \\ \underline{\underline{\mathbf{0}}} & \underline{\underline{\mathbf{C}}} & \underline{\underline{\mathbf{0}}} & \underline{\underline{\mathbf{0}}} \end{bmatrix} \begin{bmatrix} \underline{\mathbf{u}} \\ -\underline{\mathbf{p}} \\ \lambda_{\mathbf{u}} \\ \lambda_{\mathbf{p}} \end{bmatrix} = \begin{bmatrix} \underline{\mathbf{0}} \\ \underline{\mathbf{0}} \\ \underline{\mathbf{u}}_0 \\ \underline{\mathbf{p}}_0 \end{bmatrix}. \quad (9.21)$$

The stabilization parameter,  $\epsilon$ , has been suggested to be of the order of  $\Delta t$  for transient problems [11, p. 9]. As a rule of thumb  $\epsilon$  should be as small as possible but still leave the system solvable. For the present test case  $\epsilon \approx 10^{-9}$ . Written as in eq. 9.21 the BCs seem to take half the system size. Although crucially important for one particular solution, it is not generally the case. As seen in figure 9.5 the size of the BC matrix,  $\underline{\underline{\mathbf{C}}}$ , needed to solve this Stokes pipe flow is minute compared to e.g. the pressure matrix,  $\underline{\underline{\mathbf{P}}}$ .

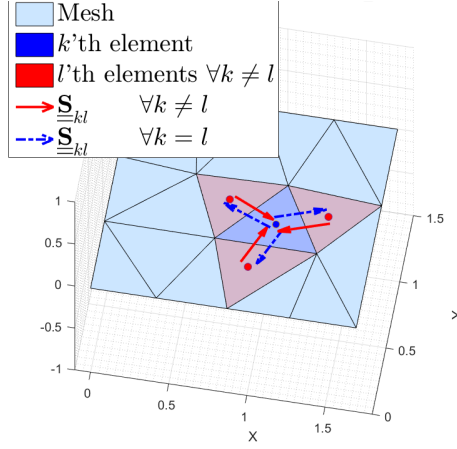


Figure 9.4: Visualization of eq. 9.20 expressing the compressibility between neighboring elements.

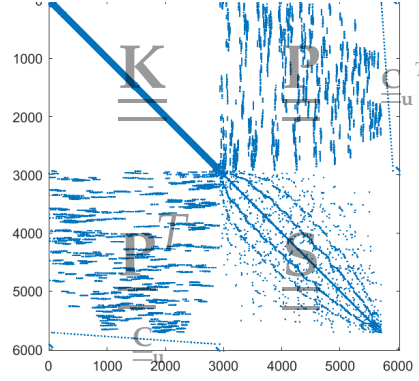


Figure 9.5: The resulting KKT matrix when solving Stokes pipe flow on a mesh with  $\approx 3000$  elements. The KKT matrix is seen in eq. 9.21. The plot should give the reader an idea of the KKT-components relative sizes.

## 9.2 Stokes flow solver verification

The 2D laminar steady state pipe flow is easily verified once the KKT matrix has been implemented. The setup, which was illustrated in figure 9.1, can be implemented with the mesh shown in figure 9.6. The nodes where BCs (from 9.13a) are imposed are shown both for  $\mathbf{u}(\mathbf{x})$  and the pressure point  $p(\mathbf{x}_0)$  (yellow square).

### 9.2.1 Solution assessment

Solving the KKT-system gives  $\tilde{\mathbf{u}}$  and  $\tilde{p}$ . The solutions are visualized in figure 9.7 ( $\tilde{\mathbf{u}}$ ) and 9.8(a) ( $\tilde{p}$ ) for a coarse mesh of  $\approx 500$  elements. The actual  $\tilde{\mathbf{u}}$ -result is the blue arrows in the XY-plane. As expected, the flow is in the  $\hat{x}$ -direction with  $\tilde{\mathbf{u}} = 0$  close to the pipe-boundary and  $\max\{\tilde{\mathbf{u}}\}$  in the center of the pipe. To (visually) compare  $\tilde{\mathbf{u}}$  to the exact solution we compute the speed,  $|\tilde{\mathbf{u}}|$ , (blue circles) and plot them along with the analytical exact values (red stars) from eq. 9.13b. As seen, all the blue circles have a red star inside them showing complete agreement between the exact and approximate solution. Black circles show imposed values from our BCs.

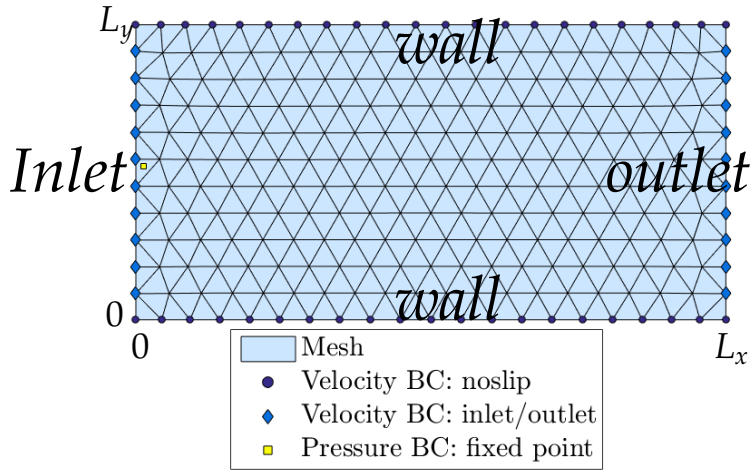


Figure 9.6: The discretized mesh showing the BCs to be imposed in order to solve for laminar steady state Stokes flow. A setup illustration is seen in figure 9.1.

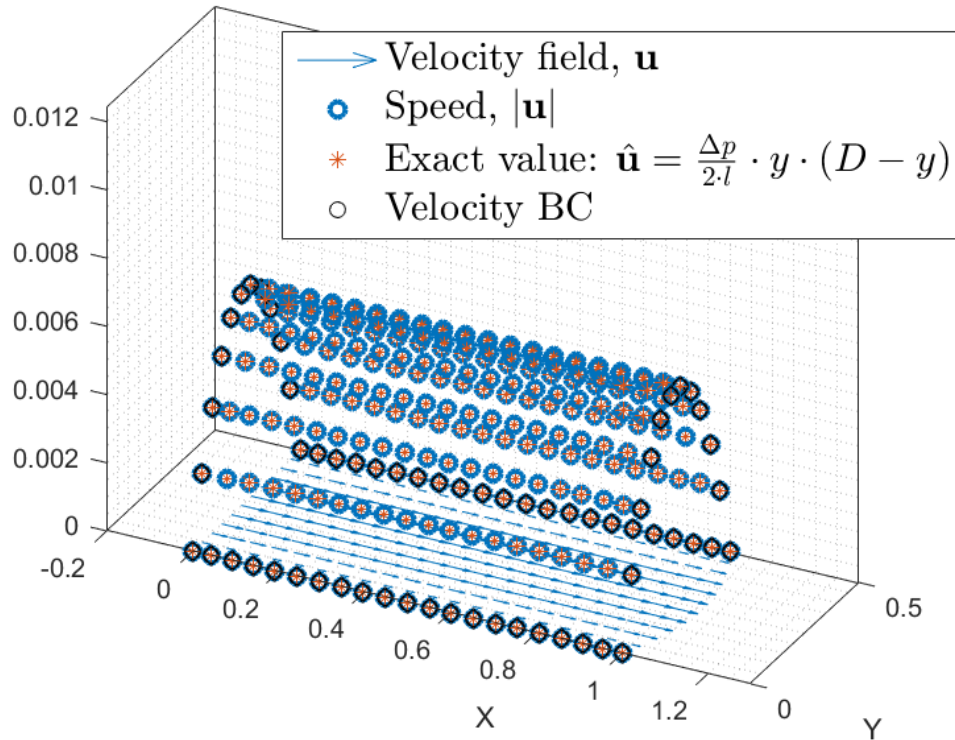


Figure 9.7: Verification of coupled solver. FEM  $\tilde{\mathbf{u}}$  solution to Stokes pipe flow is seen as blue arrows in the  $xy$ -plane. The exact, analytical solution (red stars) are right at the blue circle centers which mark the speed,  $|\tilde{\mathbf{u}}|$ . Thus, we have verified the solver.

The pressure solution is visualized as blue circles above the mesh in figure 9.8(a). As expected,  $\tilde{p}$  is a linearly decreasing plane along the  $\hat{x}$ -direction. The plane descends from  $\tilde{p} \approx p(\mathbf{x}_0)$  at the fixed pressure point (black square) located on the  $y$ -axis to  $\tilde{p} \approx 0$  at  $x = L_x$ . Except for a few outliers due to the crudity of the mesh we also find the pressure solution as we expected. From the figure it is clear that  $p_0 = 0.2$ ,  $L_x = 1$  and  $L_y = 0.5$ . It is now trivial to compute  $u_{max}$  using eq. 9.13b:  $u_{max} = \frac{0.2}{2 \cdot 1} \cdot 0.25(0.5 - 0.25) \Leftrightarrow u_{max} = 0.00625$ . Reassuringly it seems in agreement with figure 9.7. The verification-section above is a qualitative assessment. Obviously the solver is correct as is evident from the figures but this would not suffice as a proper solver verification. To describe how correct a solver is one usually also calculates *error convergence rates*. To intuitively introduce the topic one could compare the pressure solution from figure 9.8(a) with the solution in figure 9.8(b) which is obtained using a refined mesh of  $\approx 3000$  elements. As seen from the plots the solution improves as the mesh is refined. One can calculate the actual rate with which the integral of the *error field* diminishes as we refine the mesh. Depending on which element type we use we should obtain a certain error convergence rate. Should a solver not be put together optimally it would therefore become apparent here. Seeing that this part of the thesis is meant as a more intuitive, pedagogical introduction to the FEM we will wait with the actual error convergence calculations until part IV where we will pick up where we left off: simulating pipe flow.

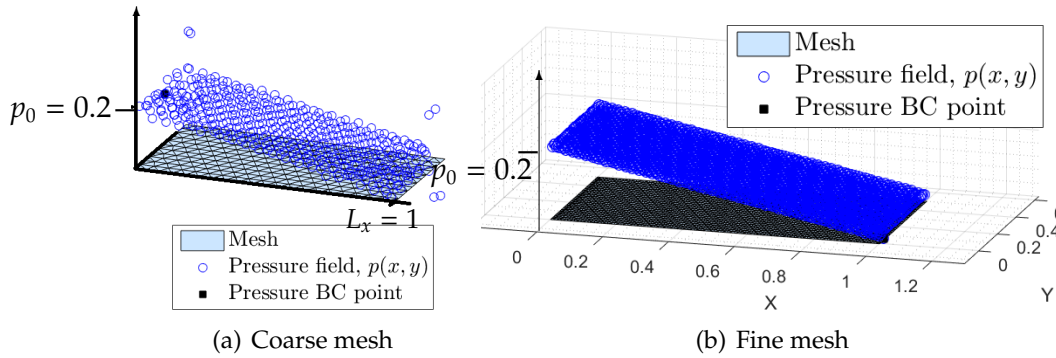


Figure 9.8: Visualization of *error convergence*. As the mesh is refined we increasingly get a better solution. LHS: FEM  $\tilde{p}(x, y)$  solution to Stokes pipe flow obtained on a coarse mesh. RHS: FEM  $\tilde{p}(x, y)$  solution to Stokes pipe flow obtained on a refined mesh.

### 9.3 Navier-Stokes solver

Having put together a functioning Stokes flow solver we are now not far from having an actual Navier-Stokes solver *written from scratch*. The two steps left to be taken are:

1. Making the solver transient,  $\frac{\partial \mathbf{u}}{\partial t}$ , and
2. Adding non-linearity,  $(\mathbf{u} \cdot \nabla) \mathbf{u}$ .

Where the coupling of velocity and pressure truly is the decisive step for building intuition on a self-written solver, these remaining two additions are straightforward and offer little new insight. Therefore, a brief mentioning will suffice.

#### 9.3.1 Stokes flow transient analysis

To have a time-dependent problem we add the term  $\frac{\partial \mathbf{u}}{\partial t}$  to our problem equation. We have seen that the integral of such a term with a test function results in the matrix,  $\underline{\underline{\mathbf{M}}}$ , as in e.g. eq. 8.44. The previous problem equations 9.1a and 9.16 as well as the matrix equations 9.10 and 9.11 then change to eq.s 9.22a-9.22b and 9.22c-9.22d respectively seen below:

$$\frac{\partial \mathbf{u}}{\partial t} - \nabla^2 \mathbf{u} + \nabla p = \mathbf{0} \quad (9.22a)$$

$$\nabla \cdot \mathbf{u} - \epsilon \nabla^2 p = 0 \quad (9.22b)$$

$$\underline{\underline{\mathbf{M}}} \dot{\mathbf{u}} + \underline{\underline{\mathbf{K}}} \mathbf{u} - \underline{\underline{\mathbf{P}}} \mathbf{p} = \mathbf{0} \quad (9.22c)$$

$$\underline{\underline{\mathbf{P}}}^T \mathbf{u} - \epsilon \underline{\underline{\mathbf{S}}} \mathbf{p} = \mathbf{0} \quad (9.22d)$$

In turn, we use a simple backward difference to approximate  $\dot{\mathbf{u}} \approx \frac{1}{\Delta t}(\mathbf{u}^{t+\Delta t} - \mathbf{u}^t)$ . This alters eq. 9.22c to:

$$\left( \underline{\underline{\mathbf{M}}} + \Delta t \underline{\underline{\mathbf{K}}} \right) \mathbf{u}^{t+\Delta t} - \underline{\underline{\mathbf{M}}} \mathbf{u}^t - \Delta t \underline{\underline{\mathbf{P}}} \mathbf{p} = \mathbf{0}. \quad (9.23)$$

The final KKT-matrix can be seen below. As seen, the  $\underline{\underline{\mathbf{A}}}$  matrix contains both  $\underline{\underline{\mathbf{M}}}$  and  $\underline{\underline{\mathbf{K}}}$ . We have set  $\mathbf{b} = -\underline{\underline{\mathbf{K}}} \mathbf{u}^t$  and the solution vector for pressure has been renamed entirely to  $\mathbf{p}' = -\Delta t \mathbf{p}$ . This can be done since we now have a transient case and therefore have set the stabilization parameter to

$\epsilon \approx \Delta t$  as suggested in the literature [11, p. 9]:

$$\begin{bmatrix} \underline{\underline{\mathbf{A}}} & \underline{\underline{\mathbf{P}}} & \underline{\underline{\mathbf{C}}}^T & \underline{\underline{\mathbf{0}}} \\ \underline{\underline{\mathbf{P}}}^T & \underline{\underline{\mathbf{S}}} & \underline{\underline{\mathbf{0}}} & \underline{\underline{\mathbf{C}}}^T \\ \underline{\underline{\mathbf{C}}} & \underline{\underline{\mathbf{0}}} & \underline{\underline{\mathbf{0}}} & \underline{\underline{\mathbf{0}}} \\ \underline{\underline{\mathbf{0}}} & \underline{\underline{\mathbf{C}}} & \underline{\underline{\mathbf{0}}} & \underline{\underline{\mathbf{0}}} \end{bmatrix} \begin{bmatrix} \underline{\underline{\mathbf{u}}} \\ \underline{\underline{\mathbf{p}'}} \\ \underline{\underline{\lambda}}_{\mathbf{u}} \\ \underline{\underline{\lambda}}_p \end{bmatrix} = \begin{bmatrix} -\underline{\underline{\mathbf{b}}} \\ 0 \\ \underline{\underline{\mathbf{u}}}_0 \\ \underline{\underline{\mathbf{p}}}_0 \end{bmatrix} \quad (9.24)$$

### 9.3.2 Convection implementation

With the above listed KKT-system in eq. 9.24 we only need to add the convection term,  $(\mathbf{u} \cdot \nabla)\mathbf{u}$ , to have a true Navier-Stokes solver. There are many ways to do this of which we will only touch upon one of the very simplest; *Picard Linearization*. Basically this comes down to linearizing the convection term. Another method could be a nonlinear Newton method which is described in sec. A.5.

#### Picard Linearization

Two ways of linearizing the term are:

1.  $(\mathbf{u} \cdot \nabla)\mathbf{u} \rightarrow (\mathbf{u}_t^{N-1} \cdot \nabla)\mathbf{u}_t^N$ , and
2.  $(\mathbf{u} \cdot \nabla)\mathbf{u} \rightarrow (\mathbf{u}_t^N \cdot \nabla)\mathbf{u}_t^{N-1}$ ,

where  $t$  refers to a time-step and  $N$  are Picard iterations. Here, naturally the choice between the two matters. According to B. M. DeBlois the first form,  $(\mathbf{u}_t^{N-1} \cdot \nabla)\mathbf{u}_t^N$ , is the correct approach to emulate the convection  $(\mathbf{u} \cdot \nabla)$  of our flow  $\mathbf{u}$ . The idea is to add a loop where Picard iteration can be done on the nonlinear term for each time-step (example given for  $t_i$ ):

$$\begin{array}{cc} \underline{\underline{\mathbf{u}}}_{t_i}^{N-1} & \underline{\underline{\mathbf{u}}}_{t_i}^N \\ \hline \underline{\underline{\mathbf{u}}}_{t_i}^0 = 0 & \underline{\underline{\mathbf{u}}}_{t_i}^1 \\ \underline{\underline{\mathbf{u}}}_{t_i}^1 & \underline{\underline{\mathbf{u}}}_{t_i}^2 \\ \underline{\underline{\mathbf{u}}}_{t_i}^2 & \underline{\underline{\mathbf{u}}}_{t_i}^3 \\ \vdots & \vdots \end{array}$$

which is continued to the desired degree of convergence. As explained by B. M. DeBlois only the first form produces the physics true to the NS-equations. His example is a nonlinear flow dominated by vorticity waves when the fluid is simulated to pass over a backward facing step.

The other form simply produced a flow similar to Stokes flow. His FEM analysis was carried out with *Taylor-Hood* elements (which we will present shortly) and interested readers are encouraged to read his entire analysis for more insight [7].

We shall not present further results from this self-written fluid solver but in principle, it is now straightforward to set up classical test problems like Lid-Driven Cavity. In the following section the efficient, automated PDE solver: FEniCS, will be presented along with test examples to show how easy and powerful the Finite Element Method is and how unproblematic it is to calculate relevant functionals of our solution.

## Part IV

### COMPUTATIONAL FLUID DYNAMICS




## FLOW SIMULATIONS AND ANALYTICAL COMPARISONS

“The world is an exciting place when you know CFD!”

- John N. Shadid

Professor, Computational Mathematics

As is evident from the above description of a Navier-Stokes-solver it can be strenuous work to write your own PDE-solver, not to mention the ensuing optimization process without which one will find the road to scientific results rather cumbersome. Luckily, automated scientific computing platforms, e.g. FEniCS, can be utilized and lead to very efficient solutions.

### 10.1 Mathematical framework of FEM

#### Stating Model Problems

To keep mathematics simple and focus on the FEniCS features we will use a known model problem to briefly run through some points of emphasis. An obvious choice here is the Poisson equation for static problems as it is rather simple and still easily related to more difficult equations. Furthermore, we often times use a uniform, structured mesh (for model problems) which simplifies the verification process immensely.

To solve any PDE the problem must be put in terms of a variational problem. We have done so numerous times in the more pedagogical

introduction (part III) when writing FEM solutions from scratch. We will therefore only briefly touch upon the subject and mention a few important considerations previously left out.

In part III when solving the Poisson equation we stated the weak formulation and then went on to derive each term in the element system equation to assemble it. With FEniCS things change dramatically. We need only feed FEniCS the variational formulation along with the BCs before calling a solver. Thus, what amounted to over a hundred lines of code before can now be done better and more efficient in less than 20 lines!

One important notion previously left out is *function spaces*. These must be declared along with the weak formulation. For the Poisson equation,  $-\nabla^2 u = f$ , we get the resulting weak form:

$$-\int_{\Omega} (\nabla^2 u) v \, dx = \int_{\Omega} f v \, dx \Leftrightarrow \int_{\Omega} \nabla u \cdot \nabla v \, dx = \int_{\Omega} f v \, dx. \quad (10.1)$$

Please notice that from now on the test function is simply denoted  $v$  and we drop  $\tilde{\cdot}$  on trial functions,  $\tilde{u} \rightarrow u$ . We also used that  $v$  is required to vanish on Dirichlet boundaries [19, p. 6].

Now the test function comes into play: Eq. 10.1 is supposed to hold for all  $v$  in the function space,  $\hat{V}$ . The trial function on the other hand pertains to some other function space,  $V$ , and the two spaces are *not* necessarily identical. We can now formally state the problem (more correctly):

Seek  $u \in V$  such that:

$$\int_{\Omega} \nabla u \cdot \nabla v \, dx = \int_{\Omega} f v \, dx \quad \forall v \in \hat{V}. \quad (10.2)$$

Notice how eq. 10.2 follows the typical form for a variational equation:

$$a(u, v) = L(v) \quad : u \in V \wedge v \in \hat{V}.$$

The function spaces for test and trial function can be defined as in eq. 10.3 and 10.4 [19, p. 7]:

$$\hat{V} = \{v \in H^1(\Omega) : v = 0 \text{ on } \Gamma\} \quad (10.3)$$

$$V = \{v \in H^1(\Omega) : v = u_0 \text{ on } \Gamma\}. \quad (10.4)$$

Here,  $H^1(\Omega)$  is a *Sobolev space* where  $\int_{\Omega} v^2 d\Omega$  and  $\int_{\Omega} \|\nabla v\|^2 d\Omega$  must be finite. We have to introduce yet another pair of function spaces since we wish to transform the *continuous* variational problem in eq. 10.2 to a *discrete* variational problem:

Seek  $u_h \in V_h \subset V$  such that:

$$\int_{\Omega} \nabla u_h \cdot \nabla v \, dx = \int_{\Omega} f v \, dx \quad \forall v \in \hat{V}_h \subset \hat{V}. \quad (10.5)$$

The new pair of function spaces are finite dimensional: one test space,  $\hat{V}_h \subset \hat{V}$  and one trial space,  $V_h \subset V$ . The nature of the finite function spaces are determined when we choose which finite element type we use to discretize our domain. Up until now, we have only used the standard linear triangular elements in which case  $\hat{V}_h$  would be a space of all piecewise linear functions vanishing on the boundary whereas functions of  $V_h$  would equal the BC-values,  $u_0$ , on ditto. As seen, both spaces are subsets of previously introduced spaces.

Oftentimes in FEniCS-tutorials only one space type is stated:  $u \in V$  or  $p \in P$  and then it is simply stated that the finite dimensional spaces,  $V_h$  and  $\hat{V}_h$  will be chosen by restricting the spaces to discrete versions depending on mesh and element type. Regardless of how strict one's notation is, the notion of function spaces is very important to keep track of.

As mentioned, the weak formulation can be put on the form:  $a(u, v) = L(v)$ . Here,  $a(u, v)$  is called the *bilinear form* and  $L(v)$  is the *linear form*. This may seem pedantic or trivial depending on the reader's background but it proves very useful to have a somewhat strict notation since the FEniCS programs are written in (almost) complete correspondence to the mathematical conventions. Thus, we define  $a = \dots$  and  $L = \dots$  before calling the solver.

## 10.2 Verification of FEniCS implementation

Implementing and using FEniCS can be done by C++ or Python. We have chosen the Python implementation as it was introduced in the course Continuum Mechanics, spring 2016 at NBI. Once FEniCS has been installed it can be utilized through the userinterface package `dolfin` that gives access to the DOLFIN software library written in C++ [19, p. 11].

To ascertain ourselves of a correct implementation of the FEniCS PDE solver - and to introduce key-aspects of CFD - we have conducted three short classic test cases, namely:

1. Laminar pipe flow  
CFD-aspect: Solver assessment

2. Flow around an object (see appendix A.4)  
CFD-aspect: Functional computation
3. Lid-driven cavity flow (see appendix A.5)  
CFD-aspect: Iterative solvers

All three test cases were integral parts of the project in the Continuum Mechanics course and the overall design should be contributed to Linga, G. [21]. Both for implementation and test cases more thorough information can be found consulting Refs [21, 19, 52]. As seen, we have placed two test cases to the appendix in an attempt to keep a somewhat compact storyline for the general reader. However, for those who aspire to actually practice FEM analysis the ability to compute functionals or to handle nonlinear problems is paramount. Both of these abilities are discussed at length in the two test cases which are hereby recommended.

## 10.3 Laminar pipe flow

In the first test case we will compute laminar pipe flow and will as promised subsequently test our solver performance and assess our solution quantitatively. Thus, we pick up where we left off in part III.

### 10.3.1 Weak Formulation

Since Stokes flow has already been presented we can simply use the above stated notion of function spaces to formally state the variational problem as:

Seek  $(\mathbf{u}, p) \in W$  such that:

$$a((\mathbf{u}, p), (\mathbf{v}, q)) = L((\mathbf{v}, q)) \quad \forall (\mathbf{v}, q) \in W \quad (10.6a)$$

$$\text{with } \mathbf{u} = u_{\text{exact}}(\mathbf{x}) \quad \text{on } \Gamma_{\mathbf{u}} \quad (\text{see eq. 9.13b}) \quad (10.6b)$$

$$\text{and } p = p_{\text{exact}}(\mathbf{x}) \quad \text{on } \Gamma_p \quad (\text{see eq. 9.13b}) \quad (10.6c)$$

where  $W = V \times P$  should be a mixed function space such that  $\mathbf{u} \in V$  and  $p \in P$ , and:

$$a((\mathbf{u}, p), (\mathbf{v}, q)) = \int_{\Omega} \nabla \mathbf{u} : \nabla \mathbf{v} - p \nabla \cdot \mathbf{v} + q \nabla \cdot \mathbf{u} \, dx \quad (10.7a)$$

$$L((\mathbf{v}, q)) = - \int_{\Gamma_p} p_0 \mathbf{n} \cdot \mathbf{v} \, dS. \quad (10.7b)$$

To actually define the function spaces and test/trial functions in FEniCS we only need 5 lines:

```
V = VectorFunctionSpace(mesh, 'CG', 2)
P = FunctionSpace(mesh, 'CG', 1)
W = V * P
(u, p) = TrialFunctions(W)
(v, q) = TestFunctions(W)
```

The astute reader will notice that even though we distinguish mathematically between test space,  $\hat{V}$  and trial space,  $V$ , the FEniCS program shows no such distinction. The reason is that BCs are specified independently of function spaces. Therefore, it is sufficient to use one common space in our program - hence, the tendency to only list one function space type in the problem statement [19, p. 12].

### 10.3.2 Setup

The setup can be seen in figure 10.1. Since we are in 2D we see a plane parallel with the cylindrical axis of the pipe similar to the setup in part III. Here the distance between the walls is set to  $D = 4$  and the length is chosen to  $L = 10$ . The BCs have changed slightly from the previous example with our self-written solver. At the pipe walls,  $\Gamma_u$ , we still have a no-slip condition, thus:  $\mathbf{u}(x, 0) = \mathbf{u}(x, D) = 0$ , but to get a flow we now impose a pressure gradient over the tube with a fixed pressure,  $p_0$ , at the inlet and zero pressure at the outlet:  $p(0, y) = p_0 = 1$  and  $p(L, y) = 0$ . As seen in the figure, it is necessary in FEniCS to define the different subdomains in order to impose our BCs. Once this is done we impose BCs by typing:

```
no_slip = Constant((0.0, 0.0))
bc_wall = DirichletBC(W.sub(0), no_slip, subdomains, mark["wall"])
assuming the subdomains have been marked as in figure 10.1. In this case
we get 4 subdomains: (inner) domain, wall, inlet and outlet where BCs
are imposed - all visualized in different colors. Finally, we remember to
store our BCs in a vector to be parsed to the solver:
bcs = [bc_wall, bc_inlet, bc_outlet]
```

### 10.3.3 Solution to laminar pipe flow

To acquire the solution to  $\mathbf{u}$  and  $p$  we define function spaces, impose BCs on our subdomains and finally feed FEniCS the variational problem by defining the bilinear and linear form as:

```
a = inner(grad(u), grad(v))*dx - p*div(v)*dx + q * div(u)*dx
L = inner(force, v)*dx - p_inlet*inner(n, v)*ds(mark['inlet'])\
```

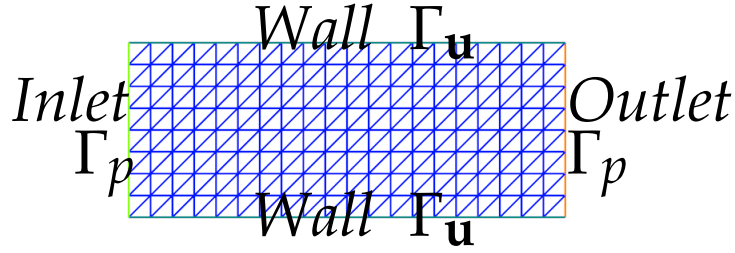


Figure 10.1: Setup for pipe flow analysis showing the four different subdomains.

```
-p.outlet*inner(n,v)*ds(mark['outlet'])
```

Then, one simply types: `solve(a == L, w, bcs)` to call the solver. Here, `bcs` is a BC-vector containing all BCs. Stated in this form the problem is assumed linear in FEniCS and the solution vector, `w`, can be split into its velocity and pressure part. Solutions for  $\mathbf{u}$  and  $p$  can be seen in figure 10.2(a) and 10.2(b) respectively. As expected, the velocity field has a parabolic profile with  $u_{max} = 0.2$  at the pipe center and  $u_{min} = 0$  at the walls. As for the pressure; since  $p_0$  was set to 1 we get a linear decreasing pressure field from inlet to outlet also as expected and in complete agreement with the exact solution (eq. 9.13b) and with the results from part III - only now we have used less than 2 dozen lines!

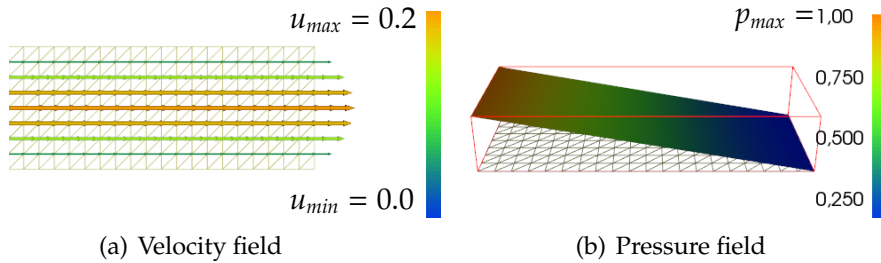


Figure 10.2: Results of pipe flow simulation using FEniCS. a) The resulting velocity field,  $\mathbf{u}$  for pipe flow analysis. b) The resulting pressure field for pipe flow analysis. Both results are seen to be in agreement with the results from the self-written coupled solver (see fig. 9.7-9.8(b)).

### Solver settings

The default solver uses "Sparse LU decomposition" (i.e. *Gaussian elimination*) for linear systems. This is a very robust approach for small 2D and 3D problems. For larger systems the method becomes too slow and memory demanding, which is why iterative methods are recommended here. A couple of popular solver methods are listed below. A complete

table of both solvers and preconditioners is given in the documentation [19, section 7.4]. There are naturally several parameters that can be tuned, e.g. number of iterations or maximum tolerance. These can all be accessed through the global parameter dictionary if need be by typing:

```
parameters['krylov_solver']['maximum_iterations'] = 500
```

- Conjugate Gradient method (CG): Requires symmetric BC implementation. Optimal Krylov solver for symmetric, positive definite coefficient matrices ( $a(u, v)$ )
- Generalized Minimal RESidual method (GMRES): Krylov solver for non-symmetric systems. Can be preconditioned with Incomplete LU factorization (ILU)

Solver implementations depend on the *linear algebra backend* to FEniCS. We use the default, PETSc, but several other backends are supported. An alternative to setting the above mentioned global dictionary of parameters is to set the parameters for each object. Instead of the compact solver call: `solve(a == L, w, bcs)`, one writes:

```
w = Function(W)
problem = LinearVariationalProblem(a, L, w, bcs)
solver = LinearVariationalSolver(problem)
solver.solve()
```

Now, to set an object parameter we type:

```
solver.parameters['linear_solver'] = 'cg'.
```

### 10.3.4 Solution assessment

As promised in part III we will now (once and for all) give a proper example of how to quantify the performance of a FEM solver. Obviously, the solver we shall end up using (Oasis [16]) has already been analyzed in this regard. Hence, we shall not pedantically do this all over. Nonetheless, it can be an instructive example to see and it certainly is a crucial ability for all FEM users to master.

The best way to verify a PDE code is not surprisingly "exact recovery of a solution" [19, p. 35]. Hereafter, the next best thing is to study the *convergence rates*. We will in the following give examples of both assessment techniques and mention some pitfalls along the way.

As stated in the literature the FEM will "exactly reproduce a second-order polynomial at the vertices of the cells" as long as we have a rectangular uniformly partitioned domain with linear triangles (i.e. P1

elements). [19, p. 9]. The analytical solutions to pipe flow (already given in eq. 9.13b) are here restated for convenience:

$$\mathbf{u}_{exact} = \frac{\Delta p}{2 \cdot L} \cdot y \cdot (D - y) \hat{\mathbf{u}} \quad (10.8a)$$

$$p_{exact} = p_{inlet} - \frac{\Delta p}{L} \cdot x. \quad (10.8b)$$

Luckily, the analytical velocity is exactly a second-order polynomial. Thus we should expect an exact recovery of the solution at the nodes (to machine precision). The maximal nodal error can be found in just 3 lines as seen below by making an expression before interpolating it to the proper space and extracting maximal error of the difference-numpy-array:

```
u0 = Expression(...)
u_e = interpolate(u0, V)
error = np.abs(u_e.vector().array() - u.vector().array()).max()
```

The result of running the above three lines (with P1 elements: 'CG'=1) will be a maximal nodal error of  $\approx 2.79 \cdot 10^{-15}$ , not far from the machine precision ( $10^{-16}$ ). Thus, we have examined our solution and verified our PDE code in the most proper way (the procedure for pressure is naturally equivalent).

### Error convergence

As mentioned, the next best way to assess the found solution is to compare  $\mathbf{u}$  and  $p$  to the analytical solutions by taking the difference in order to obtain the *Error Field*. One can as before construct an *Expression* and then project the analytic  $\mathbf{u}$  and  $p$  onto the domain,  $\Omega$ . Once this is done, we can find the local error, where error is taken to be the absolute difference between our solution and the analytical one. In turn, we can get the global error by integrating the error over the entire domain (using the function *assemble()*). We now go on to study how the found solution *converges* to the true analytical one as we refine the mesh. Hence, the name *error convergence rate*. The complete operation of integrating the error field can be stated as [19, eq. 28, p. 33]:

$$\text{error}_{global} : \|\mathbf{u}_e - \mathbf{u}\|_h = \sqrt{\int_{\Omega} (\mathbf{u}_e - \mathbf{u})^2 d\Omega}. \quad (10.9)$$

There is one crucial caveat when assessing the solution like this. Simply performing the above operations in FEniCS is straightforward and will result in a global error of  $\approx 5 \cdot 10^{-7}$  both for  $\mathbf{u}$  and  $p$ . This is however *not*



the correct (global) error. The square function in eq. 10.9 will in FEniCS be expanded leading to many round-off errors [19, p. 34-35]. Therefore, one must utilize the `errornorm()` function to circumvent this problem. Here, a more accurate error is obtained by interpolating the expression onto a space with higher-order elements. The usage is quite simple:

```
E=errornorm(u_e,u,norm_type='L2',degree_rise=3).
```

We only have to specify which norm type and how many degrees in element-order we want to use. Setting the velocity element type back to second order, 'CG'=2, we are now ready to study convergence rates.

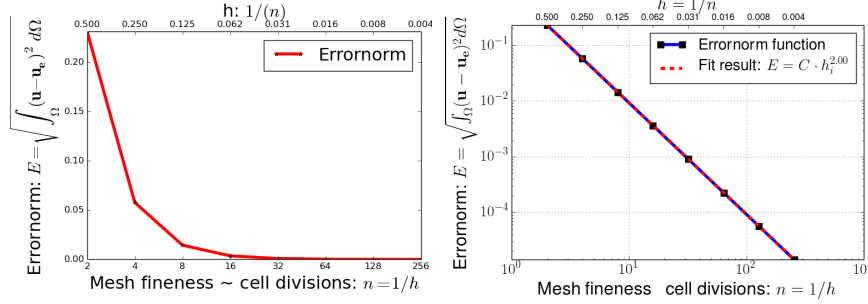
### Taylor-Hood elements

Usually, one would expect the global error to increase as the mesh becomes more coarse-grained. In this case however, we will get a small increase as we refine the mesh. Furthermore, the entire integrated error field is very small and varies from  $10^{-15} \rightarrow 10^{-13}$ . This is due to our choice of element-order. The solution has been obtained with second-order continuous-Galerkin basis functions for  $\mathbf{u}$  ('CG'=2) and first-order basis functions for  $p$  ('CG' is a synonym for 'Lagrange') [19, p. 11]. In other words we have used P2P1-elements known as the Taylor-Hood element which should be a stable element pair for Stokes equations. This is crucial since stability can be an issue for mixed function spaces. In the end this comes down to the Babuška-Brezzi stability condition [17, p. 382]. The stability theory for mixed FEM discretization is in large part owed to the work of Babuška (1973) and Brezzi (1974). The theory is treated more carefully in the FEniCS book [17, chapter 36] but will not be pursued further in the present work. The important thing to notice is that the polynomial order of the function spaces  $V$  and  $P$  (set in sec. 10.3.1) matches the order of the analytical solutions. Thus, the error norm function for the entire domain gives a *global* error around  $\approx 10^{-15}$  which (like the maximal nodal error) is also not far from machine precision ( $10^{-16}$ ) [19, p. 21].

However, if we change the velocity basis functions to first order we can study the convergence rate of our global error as we refine the mesh. Thus we carry out a number of simulations where we vary  $n$  which is the number of divisions in the  $\hat{x}$  (and  $\hat{y}$ ) direction. In turn we get a series of global errors,  $E_0, E_1, \dots, E_n$ , and element sizes:  $h_0, h_1, \dots, h_n$ . Here,  $h_0 > h_1 > \dots > h_n$  since we define  $h = \frac{1}{n}$ . The resulting error convergence plot is seen in figure 10.3(a). As expected, our solution improves as we refine the mesh.

If we assume the error dependence:

$$E_i = C(h_i)^r, \quad (10.10)$$



(a) The convergence of the Errornorm()

(b) Fit result of global error.

Figure 10.3: Error convergence visualization. a) The convergence of the Errornorm() FEniCS function as we refine the mesh in the laminar pipe flow simulation. b) Fit result of global error as the mesh is gradually refined. As seen, the expected convergence rate of,  $r = \text{degree} + 1 = 2$  for CG=1 is in complete agreement with the fit result.

where  $r$  is the error convergence rate one can obtain  $r$  by comparing two experiments:

$$r = \frac{\ln(E_i/E_{i-1})}{\ln(h_i/h_{i-1})}. \quad (10.11)$$

Should the assumed dependence hold true we would obviously expect a straight line in a log-log-plot. Thus, we plot the curve from figure 10.3(a) in a new figure with logarithmic scales on both axis as seen in figure 10.3(b).

As stated in the literature,  $r$  should approach the convergence rate  $r = \text{element\_degree} + 1$  as we increase  $n$  [19, p. 35]. This can readily be checked by consecutive simulations. In this case however, there is no sign of transient behavior as we clearly see from the figure that we obtain  $r = 2$  for 'CG' = 1 basis functions. Please note that the graph is decreasing both in figure 10.3(a) and 10.3(b) but we fit to the monotonically increasing expression in eq. 10.10 since the variable is the characteristic mesh size,  $h$  and not the number of divisions,  $n$ . Hence, a second x-axis has been added to both figures to include both the variable we fit to as well as the actual variable we change.

It is however not custom to visualize the error convergence. Usually, a simple table is given listing the convergence rate,  $r$ , for consecutive runs for a given element type. An obvious example is the paper by Mortensen

et al. where they present the Navier-Stokes solver Oasis [16, p. 186]. Here, the performance is described in a table similar to tab. 10.1. Naturally, the information in figure 10.3(b) and tab 10.1 is equivalent.

Table 10.1: **Error convergence for 2D pipe flow:** The error convergence,  $O(h^r)$ , is found with the  $L2$  norm stated in eq. 10.9 represented by the notation,  $\|\cdot\|$ . Characteristic mesh size and convergence rate is respectively:  $h$  and  $r$ .

Element: P1		
$h$	$\ \mathbf{u} - \mathbf{u}_e\ _h$	$r$
$2.50E - 01$	$5.77E - 02$	2.00
$1.25E - 01$	$1.44E - 02$	2.00
$6.25E - 02$	$3.61E - 03$	2.00
$3.12E - 02$	$9.02E - 04$	2.00
$1.56E - 02$	$2.26E - 04$	2.00
$7.81E - 03$	$5.64E - 05$	2.00
$3.91E - 03$	$1.41E - 05$	2.00

### 10.3.5 Complex meshes

If we are dealing with complex meshes (for which FEM excels) there are most likely no analytical expression to compare with. There are however other functionals one can compute to check whether we have a meaningful result. This topic will be more thoroughly introduced in the test case "Flow around objects" (A.4) but one example given here is the flux at inlet compared to flux at outlet. For a well-functioning solver the difference in flux should diminish as we refine the grid even in complex geometries. Obviously, the present geometry is hardly complex but the approach is the same. Assuming the different subdomains have been labeled as in figure 10.1 the following lines will do:

```
n = FacetNormal(mesh)
ds = Measure("ds")[subdomains]
# Calculate fluxes:
flux = -dot(u, n)
flux_inlet = assemble(flux*ds(mark["inlet"]))
flux_outlet = assemble(flux*ds(mark["outlet"]))
flux_wall = assemble(flux*ds(mark["wall"]))
print flux_inlet, flux_outlet, flux_wall
= 0.53, -0.53, -3.03E-17
```

Not surprisingly, the influx mirrors the outflux with a sign-change whereas the flux at the pipe walls is zero to machine precision.

Yet another sanity check for the solution which is easy to obtain is the divergence of the velocity field,  $\nabla \cdot \mathbf{u} = 0$ . Using the lines:

```
div_u = div(u)
```

```
div_u = project(div_u, P)
```

```
div_u_error_local = magnitude(div_u)
```

```
div_u_error = assemble(div_u_error_local*dx)
```

for dimensions:  $L = 10$ ,  $D = 4$  and element length of 0.5 the global error on the divergence is of the order  $10^{-15}$  which means the fluid practically is incompressible. Thus, our numerous tests all point to a correct implementation of FEniCS and our solver.

## TIME DEPENDENT SOLVERS

“The results of carefully executed DNS have in the fluid mechanics community the same status as carefully executed experiments.”[17, p. 420]

- M. Mortensen, K. A. Mardal and H. P. Lantangen

**T**HE main scope of the present work is as mentioned to contribute to the research field called ‘Onset of Turbulence’ by conducting DNS in the transitional turbulent regime. What exactly is meant by DNS? The FEniCS-literature states that: “Direct Numerical Simulations (DNS) is understood as the three-dimensional and time dependent numerical simulations of the NS-equations that resolve all turbulence scales and that have negligible numerical dissipation (artificial viscosity) and dispersion.” However, to actually carry out these DNS, we certainly need a well-functioning, transient solver. Even though we will end up using a solver developed by seasoned FEM practitioners it is important to be able to put together a fluid solver - and this is exactly the purpose of the present chapter.

To verify that we end up with a correctly implemented solver we will subsequently compare the results of transient 3D pipe flow to the corresponding analytical solution called *unsteady Hagen-Poiseuille flow*.

### 11.1 Chorin’s Method

The chosen solver is based on Chorin’s method but there are many viable options. In fact, the FEniCS-literature presents a thorough comparison

of no less than 6 schemes which is hereby recommended as well as the online tutorial on Chorin's method. Thus the current chapter is merely a summary and a utilization (now in 3D) of information presented elsewhere [17, chapter 21][52, tutorial 10]. Our solver will use the projection scheme based on the work of A. J. Chorin (1968) and R. Temam (1969) [4, 49]. It has been chosen since it gives the reader a good understanding of segregated solvers *and* because of its historical prominence - being the first proposed projection method of its kind [4, p. 1].

### 11.1.1 Coupled and segregated solvers

There are many options when designing a solver. Most importantly, one must choose if the solver should be *coupled* or *segregated*. A coupled solver solves simultaneously for  $u$  and  $p$  whereas a segregated solver has a fractional step method to solve for  $u$  and  $p$  one after the other (such as Chorin's method). Up until this point we have only dealt with coupled solvers. However, since these solvers demand more memory and we eventually will end up using a segregated solver-scheme we have chosen to implement a solver of the latter type.

### 11.1.2 FEM scheme

The discretization scheme in Chorin's method has three steps [17, eq. 21.7-9]:

1. Compute the *tentative* velocity,  $u_h^\star$

$$\langle D_t u_h^\star, v \rangle + \langle u_h^{n-1} \cdot \nabla u_h^{n-1}, v \rangle + \langle \nu \nabla u_h^\star, \nabla v \rangle = \langle f^n, v \rangle \quad \forall v \in V_h. \quad (11.1)$$

2. Compute the corrected pressure,  $p_h^n$

$$\langle \nabla p_h^n, \nabla q \rangle = -\langle \nabla \cdot u_h^\star, q \rangle / k_n \quad \forall q \in P_h. \quad (11.2)$$

3. Compute the corrected velocity,  $u_h^n$

$$\langle u_h^n, v \rangle = \langle u_h^\star, v \rangle - k_n \langle \nabla p_h^n, v \rangle \quad \forall v \in V_h. \quad (11.3)$$

In all three steps one includes the related BCs. Furthermore,  $h$  is mesh size,  $k_n = t_n - t_{n-1}$  is time-step and  $D_t u_h^n = (u_h^n - u_h^{n-1})/k_n$ . For this setup we choose  $V_h$  and  $P_h$  as the discrete function spaces for respectively  $v$  and  $q$  corresponding to Taylor-Hood elements (P2-P1).

In the first step we compute a  $u$ -field,  $u_h^\star$ , by leaving out the pressure

field. Notice here that the convective term is explicit. Then the velocity is projected onto a space which is divergence free. This projection step can be written like [17, eq. 21.5-6]:

$$\frac{u_h^n - u_h^\star}{k_n} + \nabla p_h^n = 0 \quad (11.4a)$$

$$\nabla \cdot u_h^n = 0. \quad (11.4b)$$

Applying eq. 11.4b to 11.4a we get  $-\nabla^2 p_h^n = -\nabla \cdot u_h^\star / k_n$  and it is this expression for which we write the corresponding variational form in eq. 11.2. This second step allows us to compute  $p_h^n$ . In the final step we simply use the results from step 1 ( $u_h^\star$ ) and step 2 ( $p_h^n$ ) to solve for  $u_h^n$  in eq. 11.4a.

### 11.1.3 Solver implementation

The FEniCS implementation closely resembles the mathematical notation and is straightforward. After defining spaces and thereby also Taylor-Hood elements like:

```
V = VectorFunctionSpace(mesh, "CG", 2)
```

```
P = FunctionSpace(mesh, "CG", 1),
```

we can define the variational form in the following way:

#### Variational form

```
k = Constant(dt)
```

```
f = Constant((0,0,0)) # dummy force
```

```
# 1) Tentative velocity step:
```

```
F1 = (1/k)*inner(u-u0,v)*dx+inner(nabla_grad(u0)*u0,v)*dx+\
      nu*inner(nabla_grad(u),nabla_grad(v))*dx-inner(f,v)*dx
```

```
a1 = lhs(F1)
```

```
L1 = rhs(F1)
```

```
# 2) Pressure update step:
```

```
a2 = inner(nabla_grad(p),nabla_grad(q)) * dx
```

```
L2 = -(1/k) * div(u1) * q * dx
```

```
# 3) Velocity update step:
```

```
a3 = inner(u,v) * dx
```

```
L3 = inner(u1,v)*dx -k*inner(nabla_grad(p1),v)*dx
```

Notice that we for each step define a bilinear form (e.g. a2) containing the

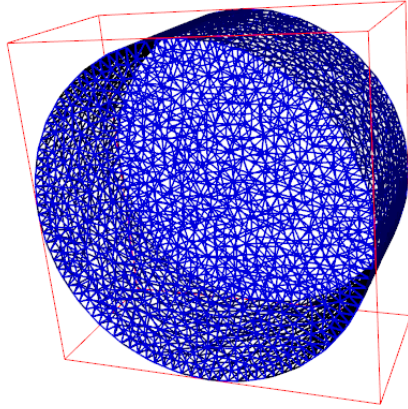


Figure 11.1: A cylindrical mesh generated with the FEniCS component 'mshr' containing 50027 elements corresponding to 9124 vertices. Radius and length are both 1.

unknown and a linear form (e.g.  $L_2$ ). Finally, we assemble the system and allocate memory for the time-iteration:

```
# matrix assembly:
A1 = assemble(a1)
# space allocation:
b1 = None
```

### Solver parameters

We now make a simple time-stepping loop where for each iteration we compute the three steps and update the time. An example is given for step 1:

```
b1 = assemble(L1, tensor=b1)
[bc.apply(A1, b1) for bc in bcu]
solve(A1, u1.vector(), b1, "gmres", "ilu")
```

The other steps are similar except for BCs and a change of preconditioner to 'algebraic multigrid' for step 2.

#### 11.1.4 Verification of fractional step solver

We are now ready to test the solver. As an analytical reference we will use the unsteady Hagen-Poiseuille flow in a cylindrical pipe. Figure 11.1 shows the setup that we will later use for the verification.

### Transient Hagen-Poiseuille

One can obtain an analytical transient solution to the cylindrical pipe for an incompressible Newtonian fluid using the Hankel transform. Further information can be found in section A.2 of the appendix. The final transient



Hagen-Poiseuille flow is (re)stated in eq. 11.5.

$$u(r, t) = (1 - r^2) - 8 \sum_{n=1}^{\infty} \lambda_n^{-3} \frac{J_0(\lambda_n r)}{J_1(\lambda_n)} e^{-\lambda_n^2 \nu t} \quad : 0 \leq u(r, t) \leq 1, \quad (11.5)$$

where  $J_n$  is the  $n$ 'th order Bessel function of first kind,  $\lambda_n$  the  $n$ 'th root of  $J_0$  and the equation is scaled so that  $u_{max} = 1$  and  $r_{max} = R = 1$ . Thus, we should expect a radial velocity profile which should converge towards the steady parabolic Hagen-Poiseuille flow for a pipe.

### Courant-Friedrichs-Lewy condition

Chorin's method is called a non-incremental pressure correction scheme and the convective term is as mentioned solved *explicitly*. Therefore, we have implemented an explicit solver and must now consider the following. *How large time-steps can be used?* To answer this question we turn to the very important aspect of transient solvers called the Courant-Friedrichs-Lewy (CFL) condition:

$$C = \frac{u_{max} \cdot \Delta t}{h_{min}} < C_{max} \quad (11.6)$$

Above,  $C$  is the dimensionless *Courant number* and  $h_{min}$  is the characteristic mesh size taken as the minimum cell diameter which is computed as two times the circumradius found in FEniCS as:

`h = mesh.hmin()`

CFL is crucial and states an upper bound of the time-step,  $\Delta t$ , if convergence is to be obtained. In an attempt to give a physical understanding of the CFL one can think of the numerator as a measure of how much distance the information in our system travels in one time-step. The denominator is obviously the element length and the fraction,  $C$ , then tells us whether we can resolve the movement of information with our chosen mesh resolution. As stated in the FEniCS-literature  $C_{max} = 1$  [17, p. 423,461]. Using unstructured grids one should be careful with the CFL. In general however, using minimum  $\Delta x$  and maximum  $\Delta t$  should give a reasonable hint.

### Adaptive time-stepping

An elegant way of dealing with the CFL criteria is to apply it in reverse: First we choose a Courant number,  $C$ , and subsequently we find the corresponding time-step,  $\Delta t$  [17, eq. 31.34]:

$$\Delta t = \frac{h_{min} \cdot C}{u_{max}} \quad (11.7)$$

This is called *adaptive* time-stepping and means that we for each iteration find a new, optimal  $\Delta t$  for the given iterative step. This way we ensure that the whole simulation is executed in the fewest possible time-steps while maintaining a constant  $C$ .

### 11.1.5 Solution to transient pipe flow

To compute the solution and compare it to the exact expression in eq. 11.5 we must either rescale it to  $u_{max}$  and  $R = 1$  which is a trivial operation *or* we can simply set up the simulation so it will yield exactly  $u_{max} = 1$ . To do the latter we generated the mesh, presented in figure 11.1 with  $R = 1$  and  $L = 1$ . It had 50027 elements, 9124 vertices and  $h_{min} = 0.076$ . We apply a pressure difference over the pipe by setting a non-zero pressure value at the inlet. Looking at the analytical expression for steady Hagen-Poiseuille cylindrical pipe flow:

$$u_z = \frac{\Delta p}{4L\mu}(R^2 - r^2), \quad (11.8)$$

we remember the pressure-scaling,  $p \rightarrow \rho \cdot p$  and use  $R = 1$  to rephrase the expression as [20, eq. 16.29]:

$$u_z = \frac{\Delta p}{4Lv}(1^2 - r^2). \quad (11.9)$$

Clearly, all parameter settings yielding  $\frac{\Delta p}{4Lv} = 1$  will allow for a direct comparison with eq. 11.5. Choosing  $\nu = 1$  we are certainly in the laminar regime and for  $L = 1$  we must choose a pressure at inlet equal to 4 since pressure at outlet is kept to zero.

Figure 11.2 and 11.3 show the pressure and velocity field respectively during the simulation at iteration time  $t = 0.21$ . Adaptive time-stepping is trivially implemented but with  $\Delta t = 0.001$  we have a Courant number of  $C = u_{max} \cdot \Delta t / h_{min} = (1) \cdot (0.001) / (0.076) = 0.013$  and need not worry because it is sufficiently below  $C_{max}$ . As expected, the pressure field is linearly decreasing from inlet to outlet. The velocity field rises from 0 at the pipe walls due to the no-slip BC and reaches maximum at the center of the pipe - also as expected.

### Error convergence

Usually, one would present a convergence table similar to tab. 10.1 presented in the test case for laminar pipe flow - only, now with  $\Delta t$  as the variable. Thus, one could describe  $O(\Delta t^k)$  where  $k$  is the temporal order of

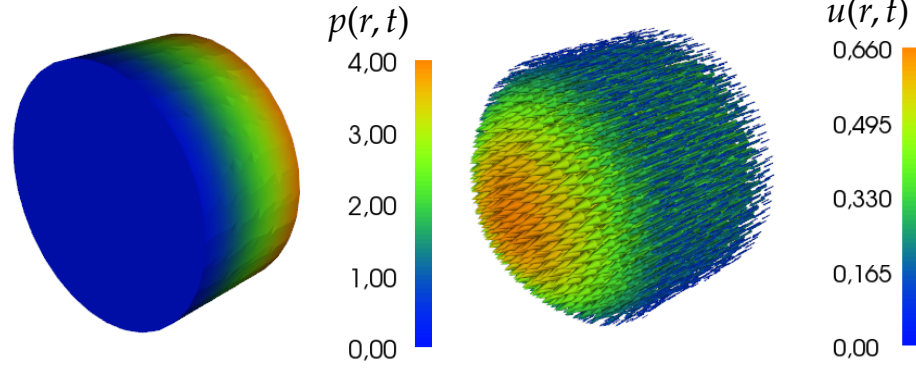


Figure 11.2: Resulting pressure field from the transient pipe flow simulation using Chorin's projection method. The simulation time for the current  $p$ -field is  $t = 0.21$ . Figure 11.3: Resulting velocity field from the transient pipe flow simulation using Chorin's projection method. The simulation time for the current  $u$ -field is  $t = 0.21$ .

convergence just as  $r$  was the spatial convergence rate.

Unfortunately, the solution we will compare with cannot be put on a closed form (it is an infinite sum). Furthermore, it is defined by Bessel functions of first kind as well as their roots. This makes it *very* problematic to define it in an `Expression()` object based on plain C++ syntax. Conveniently, in the case of the Oasis-paper; the Taylor-Green solution used in convergence estimation only entails `cos()`, `sin()` and `exp()` functions and thus it is unproblematic to present tables for  $O(\Delta t^k)$  as well as  $O(\Delta x^r)$  [16, p. 186]. The procedure is almost the same as explained for table 10.1 and only two small changes are needed. First of all one must choose an integration time (e.g.  $0 \leq t \leq 0.5$ ) [17, p. 433]. Second of all one should use higher order elements (e.g. P3 or P4) to eliminate spatial discretization error. Once the two minor changes to the procedure have been made it is no challenge to produce the convergence table.

### Comparison to transient Hagen-Poiseuille flow

Having already showed how to obtain a convergence table we now choose to verify the solver by recovery of nodal values instead. The analytic solution (eq. 11.5) yields an axial plot from  $[-R; R]$  on the x-axis and the velocity,  $u(r, t)$ , along the y-axis. Thus, we need to `probe()` the velocity

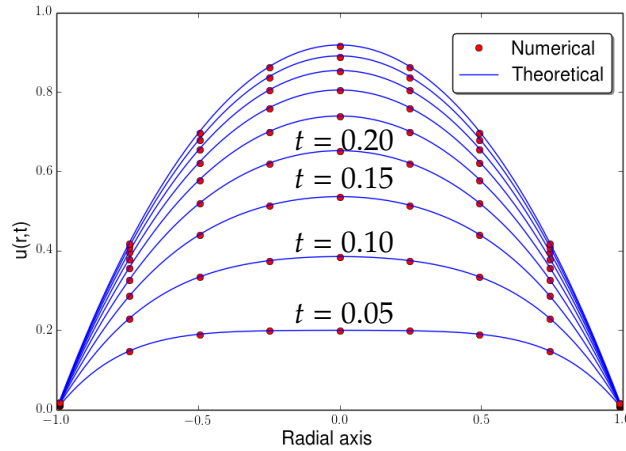


Figure 11.4: Comparison between NS simulation using Chorin's projection method ('Numerical') and the analytical solution in eq. 11.5 ('Theoretical'). There is a clear agreement between simulation and analytic velocity profile.

values,  $u$ , at various distances,  $r$ , e.g:

```
pt = Point(np.array([0.0,r,0.5])) # on y-axis
pt_val = u(pt)[2]                 # u's z-component
```

Finally, we can plot the resulting comparison in figure 11.4 which shows clear congruence between simulation ('Numerical') and analytic profile ('Theoretical'). Thus, we have put together a fully functional NS solver in FEniCS *and* we have verified it (in the laminar regime).

### High performance

As is clearly seen in the figure, the solver implemented above is fully capable of producing physically sound results and thus allows us to emulate nature itself. In principle we should not need high-performance open-source NS-solvers such as Oasis to produce interesting results but in reality the speed-up and performance increase is paramount should one wish to enter e.g. biomedical research where Oasis has been applied.

The computation time depends immensely on the solver design. In general, the FEniCS documentation points to two time-costly operations, which are assembling the RHS (e.g. b3) and solving the system [19, p. 56]. There are also minor tricks like defining the time-step as a `constant()` (i.e. `k`) instead of an `Expression()` whenever possible or writing memory-friendly by not allocating more space than what is actually needed [19, p. 13,55]. The latter is obtained by:

```
b1 = assemble(L1, tensor=b1)
```

where `b1` has been defined as `None` before the time-stepping loop. Leaving out the `tensor=b1` option (as they do in the online tutorial) will allocate new space in every iteration - obviously not a wise choice since the mesh

we shall work with has over  $5 \cdot 10^6$  vertices. Often times, assembling the bilinear form can be done before time-stepping once and for all. As for the two main time-consuming steps mentioned above one should consider avoiding the assembly of the linear form altogether as explained in section 2.3 [19]. The `solve()`-step is more tricky and has no obvious solution. As stated in the Oasis-paper, when the computational speed is dominated by the iterative solvers provided by the algebra backend it is arguably the best performance an implicit solver using PETSc may hope for [16, abstract].

## USING THE OASIS SOLVER

**T**HE structure of Oasis is a main executable module, `NSfracStep.py`, and three sub-modules, `common`, `solvers` and `problems`. To use Oasis, the minimum requirement is to specify a mesh, ICs and BCs as well as control parameters in the `problems` sub-module. Then one simply types:

```
python NSfracStep.py problem=pipe_flow
```

in the terminal to run Oasis assuming the created `problems` sub-module is called `pipe_flow.py`. Naturally, Oasis has nearly endless features and a user can specify or alter every detail one could think of - but very few settings are demanded to make Oasis run.

### 12.1 Solver and iteration scheme

Oasis is not only faster than the previously presented Chorin-solver but also more accurate and it is very stable. It has numerous options for choice of solver but the implemented default solver uses an optimized Incremental Pressure Correction scheme with an implicit Adams-Bashforth convection (IPCS\_ABCN).

#### 12.1.1 Incremental Pressure Correction Scheme

IPCS is an improvement of the non-incremental pressure correction scheme (i.e. Chorin's method). In IPCS, one does not neglect  $p$  altogether in the first

step but *uses* the pressure from the previous time-step when computing the tentative velocity,  $u_h^\star$  [17, p. 397].

### Splitting error

The segregated solvers introduce a splitting error when using the fractional step method. This error is of first order,  $O(\Delta t^1)$ , when neglecting pressure in step 1 but could in principle be completely avoided by using a coupled solver. As mentioned, this is unfortunately not feasible for large turbulence simulations. The splitting error can however fall to second order,  $O(\Delta t^2)$ , simply by using the pressure in step 1, and thus IPCS certainly is merited [17, p. 426].

The IPCS has been described in detail in the FEniCS documentation which interested readers can consult should this brief description not suffice. Similar to the Chorin's method we can write up the three steps of the IPCS [17, eq. 21.10-12]:

1. Compute the *tentative* velocity,  $u_h^\star$

$$\begin{aligned} \langle D_t u_h^\star, v \rangle + \langle u_h^{n-1} \cdot \nabla u_h^{n-1}, v \rangle + \langle \sigma(u_h^{n-\frac{1}{2}}, p_h^{n-1}), \epsilon(v) \rangle \\ + \langle p_h^{n-1} n, v \rangle_\Gamma = \langle f^n, v \rangle \quad \forall v \in V_h \end{aligned} \quad (12.1)$$

2. Compute the corrected pressure,  $p_h^n$

$$\langle \nabla p_h^n, \nabla q \rangle = \langle \nabla p_h^{n-1}, \nabla q \rangle - \langle \nabla \cdot u_h^\star, q \rangle / k_n \quad \forall q \in P_h \quad (12.2)$$

3. Compute the corrected velocity,  $u_h^n$

$$\langle u_h^n, v \rangle = \langle u_h^\star, v \rangle - k_n \langle \nabla(p_h^n - p_h^{n-1}), v \rangle \quad \forall v \in V_h, \quad (12.3)$$

where  $u_h^{n-\frac{1}{2}} = (u_h^\star + u_h^{n-1})/2$ . We now see that the pressure enters in the first step through  $\sigma$  which is the stress tensor restated below for convenience with  $\epsilon$  being the symmetric gradient [17, eq. 21.3-4]:

$$\sigma(u, p) = 2\nu\epsilon(u) - pI \quad \wedge \quad \epsilon = \frac{1}{2}(\nabla u + \nabla u^T). \quad (12.4)$$

Notice also that step 1 is missing the term:  $-\langle \nu n \cdot (\nabla u_h^{n-\frac{1}{2}})^T, v \rangle_\Gamma$  which we have left out since we will be working with *periodic* BCs [17, p. 400].

### Temporal discretization

On this topic the aforementioned FEniCS documentation [17] aligns with the work by J. C. Simo and F. Armero [46] who analyzed transient algorithms for the NS-equations in detail. The temporal discretization in our Chorin-solver above was a first-order backwards differencing scheme:  $D_t u_h^n = D_t^{q=1} u_h^n = (u_h^n - u_h^{n-1})$ . Thus fittingly, it matched the splitting error for Chorin's method. The backward differencing scheme (BDM) can be stated in the general form:

$$D_t^q u_h^n = \beta_q u_h^n + \sum_{j=0}^{q-1} \beta_j u_h^{n-1-j} \quad (12.5)$$

where  $\beta_j$  are constant coefficients and it is of interest to note that already at second-order,  $D_t^2 u_h^n = \frac{1}{2}(3u_h^n - 4u_h^{n-1} + u_h^{n-2})$ , the scheme has a second-order accuracy with respect to time,  $O(\Delta t^2)$ . To truly have a better solver than the Chorin-solver we must therefore choose  $D_t$  in eq. 12.1 wisely to maintain the second-order accuracy [22, eq. 5].

### Crank-Nicolson scheme

It certainly is possible to write a second-order accurate time discretization of the NS-equations using the BDM from eq. 12.5 as explained by M. Mortensen [22]. However, central schemes are very popular when solving NS-equations since they conserve the discrete energy in time [17, p. 423-424]. Thus, we write up the NS-equations at time  $k + \frac{1}{2}$  [22, eq. 14-15]:

$$\left( \frac{\partial \mathbf{u}}{\partial t} \right)^{k+\frac{1}{2}} + ((\mathbf{u} \cdot \nabla) \mathbf{u})^{k+\frac{1}{2}} = -\nabla p^{k+\frac{1}{2}} + \nu \nabla^2 \mathbf{u}^{k+\frac{1}{2}} \quad (12.6a)$$

$$\nabla \cdot \mathbf{u}^{k+1} = 0. \quad (12.6b)$$

To get to the expression in Oasis for step 1 we need to insert the midway approximations  $\frac{\mathbf{u}^{k+1} - \mathbf{u}^k}{\Delta t} + O(\Delta t^2)$  and  $\frac{\mathbf{u}^{k+1} + \mathbf{u}^k}{2} + O(\Delta t^2)$  for time and space and we need to deal with the convection term. One can write the term as:  $(\mathbf{u}^{k+\frac{1}{2}} \cdot \nabla) \mathbf{u}^{k+\frac{1}{2}}$  and then explicitly approximate  $\mathbf{u}^{k+\frac{1}{2}}$  by the Adams-Bashforth projection  $\mathbf{u}^{k+\frac{1}{2}} = \frac{3}{2} \mathbf{u}^k - \frac{1}{2} \mathbf{u}^{k-1} + O(\Delta t^2)$ . The final scheme is:

$$\frac{\mathbf{u}^{k+1} - \mathbf{u}^k}{\Delta t} + \left( \left( \frac{3}{2} \mathbf{u}^k - \frac{1}{2} \mathbf{u}^{k-1} \right) \cdot \nabla \right) \mathbf{u}^{k+\frac{1}{2}} = -\nabla p^{k+\frac{1}{2}} + \nu \nabla^2 \frac{\mathbf{u}^{k+1} + \mathbf{u}^k}{2} \quad (12.7a)$$

$$\nabla \cdot \mathbf{u}^{k+1} = 0. \quad (12.7b)$$



Exchanging  $\mathbf{u}^{k+1}$  with  $\mathbf{u}^\star$  we get the expression for the first IPCS step in Oasis as seen in [eq. 1 at the Oasis-wiki](#) for the fractional step solver: `NSfracStep.py` [26]. The developer of Oasis also points to Simo and Armero [46] for further insight into the IPCS-solver. Actually, the IPCS can be used iteratively as explained by M. Mortensen [22]. Thus it is in fact  $p^\star$  that enters into the first step - likewise step 2 and 3 changes a bit. The point however is, that by choosing the central Crank-Nicolson scheme we now have an IPCS-method which is accurate in time to second-order .

## 12.2 Setup and verification

Setting up your own simulations in Oasis is fairly straightforward - at least for not too complex meshes. Increasing the complexity will similarly increase the difficulties of e.g. imposing BCs the right way. Below we will give an example of how simple it can be.

### 12.2.1 Setup

We will give an example using a square pipe, i.e. a *duct*, in 3D. Why 'always' pipes or ducts? Apart from the numerous channel simulations ([33, 18, 40, 16]) both pipes and ducts are indeed very popular environments for studying turbulence. A duct has been chosen here, since it is the mesh shape in which we will perform the comparison to LBM. Also, a cylindrical pipe, as the one generated for the Chorin solver, will not have a one-to-one correspondence in nodes between inlet and outlet - a condition which must be fulfilled for periodic BCs. The study of turbulence is fittingly put into words by R. Feynman:

"The simplest form of the problem [turbulence] is to take a pipe that is very long and push water through it at high speed. We ask: to push a given amount of water through that pipe, how much pressure is needed? No one can analyze it from first principles and the properties of water. If the water flows very slowly, or if we use a thick goo like honey, then we can do it nicely. You will find that in your textbook. What we really cannot do is deal with actual, wet water running through a pipe. That is the central problem which we ought to solve some day, and we have not." [37, chapter 3, p. 16], (my bracket)

- **Richard P. Feynman**, 1964  
American Nobel Prize Laureate for Physics

According to Richard Feynman we approach the topic of turbulence in the very simplest way by investigating the motion of fluids in a pipe. As such, it is an obvious choice of environment for us.

### 12.2.2 Verification

As has been previously alluded to we will not calculate convergence tables for Oasis since it has already been done by the developers. Thus, we refer to table 1 and 2 in Ref. [16] where one can find the order of convergence  $O(h^2)$  and  $O(\Delta t^2)$  for spatial and temporal convergence when using P1-P1 elements. The convergence rates are obtained using the 2D-Taylor-Green flow but also a turbulent channel flow from Oasis has been (favorably) compared to the classical, spectral simulations Moser et al [18]. Oasis is in short very thoroughly tested.

#### Periodic boundary conditions

As stated by Feynman the duct should be *very* long. This is easily emulated by using periodic boundaries. The mesh and periodic BCs can in Oasis be implemented as:

```
# define mesh
def mesh(refine=5, **params):
    mesh = BoxMesh(Point(0,0,0),Point(1,1,2),1*refine,1*refine,2*refine)
    return mesh

# define pbc
class PeriodicDomain(SubDomain):
    # target domain, G:
    def inside(self, x, on_bnd):
        return bool(near(x[2],0) and on_bnd)
    def map(self, x, y):
        y[0] = x[0]
        y[1] = x[1]
        y[2] = x[2] - 2.
constrained_domain = PeriodicDomain()
```

The generated mesh is a  $1 \times 1 \times 2$  duct and the `refine` parameter can be overloaded to adjust mesh resolution when calling Oasis. As seen, the periodic boundaries in FEniCS are obtained by defining a target sub-domain,  $G$ , and a map;  $F : H \rightarrow G$ , from the sub-domain  $H$  to the target

sub-domain  $G$ . In our case we send the fluid from outlet back to the inlet of the duct. Thus we have an infinite duct mesh.

We still need to define BCs and ICs before calling the solver. This is however easily done by defining the duct walls with a no-slip condition and initializing from zero:

```
# define Walls
def walls(x, on_bnd):
    return on_bnd and (near(x[1],0.0) or \
                       near(x[1],1.0) or near(x[0],0.0) or near(x[0],1.0))

# define bcs
def create_bcs(V, sys_comp, **NS_namespace):
    bcs = dict((ui, []) for ui in sys_comp)
    bc0 = DirichletBC(V, Constant(0), walls)
    bcs['u0'] = [bc0]
    bcs['u1'] = [bc0]
    bcs['u2'] = [bc0]
    return bcs

# define initial conditions
def initialize(x_1, x_2, bcs, **NS_namespace):
    for ui in x_2:
        [bc.apply(x_1[ui]) for bc in bcs[ui]]
        [bc.apply(x_2[ui]) for bc in bcs[ui]]
```

The above functions are explained in detail in the Oasis manual in its introduction to Lid Driven Cavity [23]. Obviously, since we are working with a different setup the functions are different in some aspects. One should however, by knowledge of general FEniCS syntax and inspiration from the many examples in the Oasis git, be able to change the functions to fit a given setup. It is noteworthy here that we initialize at two previous time-steps:  $x_1$  and  $x_2$ . This is in complete agreement with the IPCS in eq. 12.7a where we need  $\mathbf{u}^k$  and  $\mathbf{u}^{k-1}$  to find  $\mathbf{u}^{k+\frac{1}{2}}$  which was the tentative velocity,  $\mathbf{u}^\star$ .

### Solution assessment

We are finally ready to use Oasis. Assuming the above functions have been put in the problems sub-module `duct.py` and that we have added some proper parameters, we open a terminal and type:

```
python NSfracStep.py problem=duct refine=10.
```

If we have added a plot command in `duct.py` the solution and mesh can

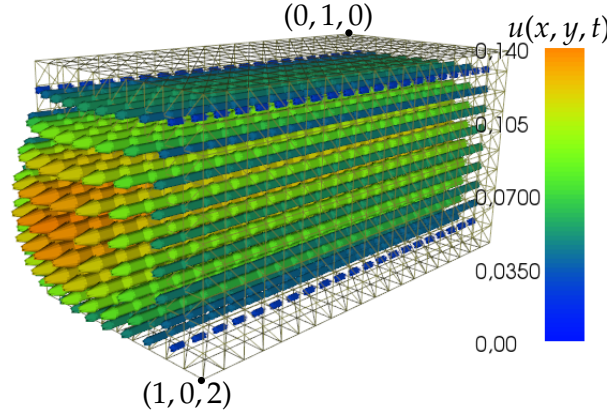


Figure 12.1: The  $u$ -field solution to NS-equations in a duct. The frame is from iteration time  $T = 82$ . The convergence of the transient solution towards the steady-state profile stated in eq. 12.8 can be seen in figure 12.3.

be shown as in figure 12.1 where we see the velocity field rising from zero at the walls to maximum in the center of the duct. More complex domains also increase the probability of imposing BCs in an erroneous way. For example the duct we will work with in the LBM comparison has deformations along one side of the duct. Therefore it is useful to be able to check that the BCs have been correctly imposed on the system.

To this end we introduce the steady-state Hagen-Poiseuille flow for a duct:

$$u(x, y) = \frac{\Delta p}{\mu L} \frac{4h^2}{\pi^3} \sum_{n_{\text{odd}}} \frac{1}{n^3} \left( 1 - \frac{\cosh(n\pi x/h)}{\cosh(n\pi\omega/2h)} \right) \sin(n\pi y/h) \quad (12.8)$$

The Hagen-Poiseuille solution to (square) duct pipe flow is different from the cylindrical solution and can be obtained using Fourier series as explained in Ref. [36, eq. 10]. Here,  $-\frac{w}{2} < x < \frac{w}{2}$  and  $0 < y < h$  span the cross sectional area of the duct as shown in figure 12.2.

A quick and dirty way of assuring ourselves that the BCs are imposed correctly is to check if we converge towards the steady-state flow from eq. 12.8. Thus we make a figure comparable to the Chorin-solver verification figure 11.4 but now only the steady-state analytical solution (blue line) is plotted as seen in figure 12.3. We then *probe* the velocity at nine points along the  $y$ -axis of the duct at different times and observe a convergence towards the steady state. The line is positioned at:  $[x, y, z] = [0.5, y_i, L/2]$  where  $y_i$  are equidistantly interspersed (red, dashed line in figure 12.2). Here, the parameters were  $\Delta t = 0.1$  and  $\nu = 0.01$ . For periodic boundaries we must use a forcing of magnitude  $F_0$  instead of imposing a pressure gradient. Here, we set  $\Delta p = L \cdot F_0$  to ensure that the pressure gradient in the analytic Hagen-Poiseuille flow corresponds to the forcing on the fluid

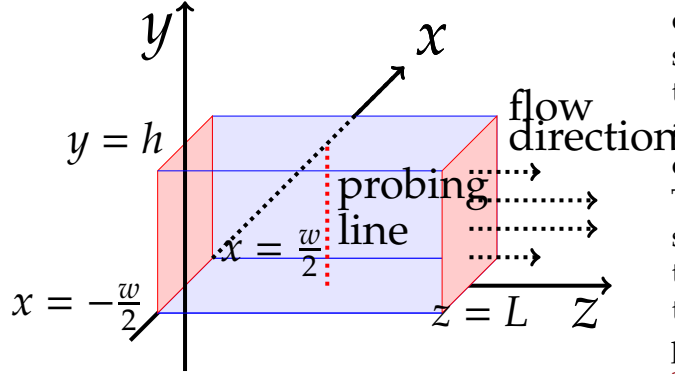


Figure 12.2: Setup for duct simulation in Oasis with no-slip conditions at the wall (blue) and periodic BCs from outlet to inlet (red). The red, dashed line shows the positions of the probes resulting in the time-dependent  $u$ -profile plot in figure 12.3.

regardless of which length we choose for the duct [17, p. 432].

How do we estimate the iteration time,  $T$ , on which we will obtain convergence? Since there is no turbulence in this simulation the viscous forces must be the dominating mechanism. We therefore use the characteristic viscous timescale introduced in sec. 4.4.1:  $\tau_v = L_0^2/\nu$ . Once the shear forces have propagated from the walls to the center of the duct the driving force on the fluid should balance the shear forces and we should be close to the steady-state profile. For our parameters we get:  $\tau_v = \frac{1}{0.001} = 100$ . The amount of iterations should therefore be around:  $N \cdot \Delta t = \tau_v \Leftrightarrow N = \frac{100}{0.1} = 1000$  iterations. Reassuringly, it is evident from the figure that the solution converges nicely towards the steady-state solution and there is hardly any discrepancy between the velocity profiles at  $T = 100$ . It should be stressed that the above is an *estimate* and should be used to build up intuition. Should figure 12.3 give any reason for concern one can perform a thorough verification as we did for the Chorin-solver. We have naturally done so and the result can be found in figure A.20 of the appendix where additional Oasis information is found.

Thus, we should now be ready to simulate actual turbulence with a very accurate and efficient solver and correctly imposed BCs - at least for a smooth, non-deformed duct. We simply need to choose parameters for the fluid and then run the simulation for a considerable amount of time.

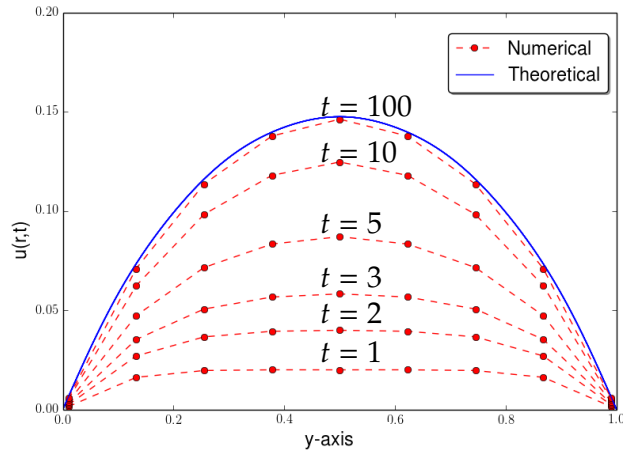


Figure 12.3: The transient solution from the duct simulation slowly converges to the analytical steady-state profile in eq. 12.8. The probing line's position in the duct can be seen in figure 12.2.

## INITIALIZATION OF SIMULATIONS

**W**E will in the following simulate in the turbulent regime on large meshes. This can be achieved by initiating the velocity at zero and then running the simulation with a constant forcing until fully developed turbulence occur. This should in theory work and we could just use the same procedure as in the laminar Oasis-example above. In fact, we would only need to load a different mesh and add a few lines when imposing BCs. Unfortunately, initiating from zero leads to a considerable amount of waiting time. An alternative is to initiate the simulation from a pre-computed velocity profile. An approximation based on having tested the two approaches is that *ten times as much* iteration time is actually needed compared to starting from a suitable profile. Put bluntly, since we only had a couple of months available we would not have obtained any results at all(!) had we not pursued this different initiation step. In the following we will describe how to obtain such a profile and most importantly how to feed it to Oasis.

### 13.1 Setup

The duct-setup shown in figure 13.1 resembles the setup for the final part (V). The dimensions of the duct are now a  $1 \times 1 \times 40$  square pipe with four small bumps in the  $xz$ -plane positioned at:

$$(x, y, z) = (0.5, 0, z_i) \quad \forall z_i = [5, 15, 25, 35].$$

Thus, all bumps are all equidistantly interspersed due to periodic boundaries. Each bump is a half-sphere with centers placed at the coordinates

stated above and with a radius of  $r = 0.1$ .

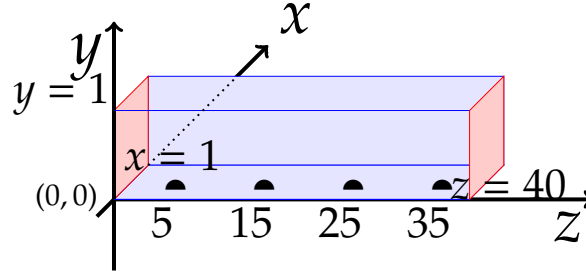


Figure 13.1: Setup for turbulence simulation in a duct using Oasis with no-slip conditions at the wall (blue) and periodic BCs from outlet to inlet (red). Deformation bumps are marked as black, filled half spheres. The figure does not scale correctly with the mesh used in simulations and therefore various measures of interest are given in the figure.

## 13.2 Steady-state solver

We now solve for the steady Stokes flow in the duct and subsequently interpolate it to the Oasis when starting the turbulence simulation. We could also have started from laminar Hagen-Poiseuille flow interpolated to the duct (which is a considerably easier feat). This would however violate the incompressibility condition at the bumps. Thus, we try to solve for the  $u$ -profile instead.

Earlier we presented a 2D pipe flow steady state solver using Taylor-Hood elements which had the default sparse LU-decomposition settings. This is simply not possible in a 3D mesh with close to 27 *M* elements. Thus, we use an iterative solver even though the system is linear as we have no convective term. For large problems such as this it can also be valuable to keep the numbers of parameters to determine at a minimum, e.g. by choosing simple elements such as P1-P1. This element is not in the Taylor-Hood family and we must therefore use a *pressure stabilized method*.

### 13.2.1 Stabilization

In this technique (introduced back in part III) we add some (pseudo-)compressibility in order to obtain convergence by adding a term involving the pressure to the continuity equation [17, p. 382]. Thus, the original



weak form of the Stokes equations:

$$a((u, p), (v, q)) = \int_{\Omega} \nabla u \cdot \nabla v - (\nabla \cdot v)p + (\nabla \cdot u)q \, dx \quad (13.1a)$$

$$L((v, q)) = \int_{\Omega} f \cdot v \, dx + \int_{\Gamma} g \cdot v d\Gamma \quad (13.1b)$$

changes to:

$$a((u, p), (v, q)) = \int_{\Omega} \nabla u \cdot \nabla v - (\nabla \cdot v)p + (\nabla \cdot u)q + \delta \nabla q \cdot \nabla p \, dx \quad (13.2a)$$

$$L((v, q)) = \int_{\Omega} f \cdot v + \delta \nabla q \cdot f \, dx. \quad (13.2b)$$

As always, the bilinear and linear form given above express the weak form as  $a((u, p), (v, q)) = L((v, q))$ . The extra term in eq. 13.2a has a plus in front since the continuity equation is modified from  $\nabla \cdot u = 0$  to  $\nabla \cdot u - \delta \nabla^2 p = 0$ . When integrating the last term by parts we flip sign and we therefore have a plus in the last term of eq. 13.2a. This stabilization method and the change in weak form is described thoroughly in Ref. [17, Chapter 20] ([17, eq. 20.4-5 & eq. 20.9-10]).

The perceptive reader will notice a discrepancy to the Stokes flow formulation in part IV (eq. 10.7b). This is due to the change in BCs. Usually, we list the BCs as:

$$u = u_0 \quad \text{on } \Gamma_D \quad \wedge \quad \nabla u \cdot n - p n = g \quad \text{on } \Gamma_N. \quad (13.3)$$

In the example in part IV we had (Neumann)  $p$ -boundaries for inlet and outlet and thus it figured in the linear form,  $L((v, q))$ . Here, we have periodic inlet/outlet BCs instead and  $p$  therefore does not enter the linear form here. However, as one uses *forcing* to emulate the pressure gradient the linear form,  $L((v, q))$  in eq. 13.2a *does* have a force term instead.

### Stabilization parameter

The parameter  $\delta$  determines the degree to which extent we alter the continuity equation. For this simulation we follow FEniCS-literature and tutorials and thus set  $\delta = \beta h_{min}^2$  where  $h_{min}$  is the minimum cell diameter which is computed as two times the circumradius and  $\beta$  is some small number.

### 13.2.2 Solver settings

The actual code-implementation, as is always the case for FEniCS code, closely matches the mathematical definitions. The first part of the script is simply to load in a mesh and define the periodic BCs. The latter part is similar to the Oasis code-snippet on periodic boundaries save for the fact that we must tell FEniCS explicitly of the periodicity when subsequently creating function spaces:

```
# define mesh
mesh = Mesh()
hdf = HDF5File(mesh.mpi_comm()/mesh/finest_duct.h5", "r")
hdf.read(mesh, "/mesh", False)

# define pbc
class PeriodicDomain(SubDomain):
    # target domain, G:
    def inside(self, x, on_bnd):
        return bool(near(x[2], 0) and on_bnd)
    def map(self, x, y):
        y[0] = x[0]
        y[1] = x[1]
        y[2] = x[2] - 40.
pbc = PeriodicDomain()

# define test, trial and mixed spaces
V = VectorFunctionSpace(mesh, "CG", 1, constrained_domain=pbc)
P = FunctionSpace(mesh, "CG", 1, constrained_domain=pbc)
W = V * P # mixed space
Defining subdomains and imposing BCs have already been exemplified
in sec. 12.2.2 and as before the no-slip BCs are stored in the vector bcs.
Since we have no explicit pressure boundaries we must remember to
constrain one pressure point, here taken to be  $(x, y, z) = (0, 0, 0)$ . This is in
turn added to the same BC vector:
# define pressure reference point
p_0 = Constant((0.0))
bc_p_point = DirichletBC(W.sub(1), p_0, "x[0] < DOLFIN_EPS \
        x[1] < DOLFIN_EPS && x[2] < DOLFIN_EPS", "pointwise")
bcs = [bc.wall, bc_p_point] # BCs vector
As for solver settings, we have here followed (and tested) the several
Stokes flow-examples available on fenicsproject.org and of interest is
the 16'th tutorial where it is described how to define a preconditioner
```

explicitly. These settings have proven exceedingly efficient which is why we have preferred them. Preconditioning has already been introduced but only with FEniCS keywords. An important extension is to learn how to explicitly define a form to use. For the above equations it is stated in the online tutorial that a suitable preconditioner is of the form:

$$\text{prec} = \int_{\Omega} \nabla u \cdot \nabla v + p \, q \, dx \quad (13.4)$$

Please note: In many FEniCS tutorials the sign of the pressure is flipped compared to the classical definition for symmetry reasons as we will do now when defining the weak form. Also, the mentioned tutorial has no stability implementation so it is mainly the lines defining the preconditioner and solver settings one should use. The stabilized weak form we will use defined in eq. 13.2a and 13.2b (with  $p$ -sign change) have instead the resulting lines seen below:

```
a = nu*inner(grad(u),grad(v))*dx +p*div(v)*dx\
    +q*div(u)*dx+delta*inner(grad(q),grad(p))*dx
L = inner(force,v+delta*grad(q))*dx
```

Similarly, the preconditioner in eq. 13.4 can be defined as:

```
b = inner(grad(u), grad(v))*dx + p*q*dx
```

The lines to (symmetrically) assemble the system and subsequently solve it are as in the tutorial:

```
A, bb = assemble_system(a, L, bcs)
P, btmp = assemble_system(b, L, bcs)
solver = KrylovSolver(krylov_method, "amg")
solver.set_operators(A, P)
```

Before calling the solver with:

```
solver.solve(w.vector(), bb)
```

we add the lines:

```
parameters["std.out_all_processes"] = False;
parameters['krylov_solver']['relative_tolerance'] = 10e-03;
```

The first line mutes all but the root processor. It should be noted that we will need 28 processors to solve the steady-state flow so that line is very convenient. The second line stops the convergence at a relatively early time. This is okay in our situation as we simply want some (not too precise) starting point for our simulation and we would expect a transient period anyway before we reach fully developed turbulent flow.

### Scaling from steady-state

There are several simulations we would like to use in our comparison with the LBM. Parameters for some of them can be found in table 13.1.

It is *very* important to optimally adjust the stability parameter.

Table 13.1: **Parameters for simulations in the transitional regime:**  $F_0$  is the non-zero force component along the duct axis.  $Re$ -numbers are approximate.

$\nu$	$F_0$	$Re$	$Re_\tau$
$0.9E - 05$	$1.2E - 05$	2400	$\approx 96$
$0.9E - 05$	$1.8E - 05$	2600	$\approx 117$

Otherwise the solver will simply not converge! To avoid having to tune  $\beta$  in the stabilization parameter for every steady-state profile we only solve the Stokes equations *once* by solving:

$$-\nabla^2 \mathbf{u}' = -\nabla p' + \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \quad (13.5)$$

and then use the scaling:

$$\mathbf{u}' = \nu/F_0 \cdot \mathbf{u} \quad \wedge \quad p' = 1/F_0 \cdot p \quad (13.6)$$

to get the  $u$ -profile we really wanted which is a solution to the equation:

$$-\nu \nabla^2 \mathbf{u} = -\nabla p + \begin{bmatrix} 0 & 0 & F_0 \end{bmatrix}^T. \quad (13.7)$$

For the unity equation in eq. 13.5 the stabilization method will converge with  $\beta = 0.2$  in the stabilization parameter  $\delta = \beta h_{min}^2$ . The last thing needed is to save the profile from eq. 13.5 in file:

```
ufile = File('v.xml', 'compressed')
pfile = File('p.xml', 'compressed')
ufile = < < u
ufile = < < p
```

In turn, once we wish to feed the profile to Oasis there are (at least) two options. One is independent of the fenicstools library but also very difficult to use. The other is based on fenicstools functions and unproblematic to use.

The reason we cannot just ask Oasis to read in the files saved above is the periodic BCs. Just as we in the steady-state solver before told FEniCS that the BCs are periodic when creating function space - Oasis will create space based on periodicity. The actual computational mesh doesn't include the

end slice at  $z = 40$  as it is identical to the first slice at  $z = 0$ . A simple load command in Oasis will therefore fail, as it will tell the operator that the profile is too big for the mesh. One solution is to make a sub-mesh where we have cut off the end slice:

```
# remove end slice
Class EndSliceOff(SubDomain):
    def inside(self,x,on_bnd):
        return x[2] < 40-DOLFIN_EPS
# define sub mesh:
sub_mesh = SubMesh(mesh,EndSliceOff())
V_sub = VectorFunctionSpace(sub_mesh,'CG',1)
P_sub = FunctionSpace(sub_mesh,'CG',1)
# make profiles for Oasis
u_sub = interpolate(u,V_sub)
p_sub = interpolate(p,P_sub)
```

These `_sub` profiles can then be given to Oasis. Unfortunately, the code above only works for the coarse mesh ( $\approx 11$  M cells). The threshold `x[2]<40-DOLFIN_EPS` does not remove all unwanted nodes on the end slice for the fine mesh ( $\approx 27$  M cells) which results in non-matching profiles when loaded into Oasis.

Luckily, there is a very convenient function in `fenicstools` called `interpolate_nonmatching_mesh()`. The name should be self-explanatory and is just what we need in this situation. Now, the scaling can be carried out in the Oasis problem module for a given simulation. First we must import the function at the very top of the script:

```
from fenicstools import interpolate_nonmatching_mesh
```

Further down in the problem module one can define an `initialize()` function which Oasis will call before starting the generic fractional step solver:

```
def initialize(q,q_1,q_2,VV,nu,**NS_namespace):
    if restart_folder is None:
        m2 = Mesh()
        hdf = HDF5File(m2.mpi_comm(),finest_duct.h5,'r')
        hdf.read(m2,'/mesh',False)
        VP_old = FunctionSpace(m2,'CG',1)
        p = Function(VP_old,'p.xml')
        u0z = Function(VP_old,'v_z.xml')
        # scale to turbulent (pseudo)steady state:
        F_0 = 1.8e-5 # approx Re=2600
        uz_array = u0z.vector().array()
        p_array = p.vector().array()
```

```

uz_array *= F_0/nu
p_array *= F_0
u0z.vector()[:] = uz_array
p.vector()[:] = p_array

```

Above we have omitted several lines, e.g. that all  $x$  and  $y$  components of the velocity profile are similar to the  $z$  component. Now that the `initialize()` function has the profile in `u0z` and `p` - we only need to interpolate them. Thus, the end of the function has a loop for `p`, `ux`, `uy` and `uz` similar to:

```

for ui in q_:
    if ui=='p':
        vv = interpolate_nonmatching_mesh(p,VV[ui])
        q[ui].vector()[:] = vv.vector()[:]
        q_1['p'].vector()[:] = vv.vector()[:]
        :

```

The above loop is shown for pressure but it is similar for velocity components except for the fact that we need to store the two previous time-steps, `q_1` and `q_2`, instead of just one in the pressure case. Now, we are finally ready to initiate any desired simulation from a laminar profile simply by scaling it.

### 13.3 Assessment of steady-state

We can now visualize the steady-state solution to make sure we got the expected results before we start the turbulence simulation. In figure 13.2 and 13.3 the  $u$ -profile for the entire duct and a zoom of the result from one of the 28 processors are shown. The zoom is a slice through the profile. The chosen processor has a mesh-partition which includes the first deformation bump at  $z_i = 5$ .

Is the steady-state solution as we expected? Judging from figures 13.2 and 13.3 it does indeed look like a standard laminar profile. Two quick ways of making sure the result is meaningful is to either interpolate the Hagen-Poiseuille flow onto a very coarse (similar but regular)  $1 \times 1 \times 40$  duct with same forcing or to simply solve the Stokes flow on such a duct. Both is fairly straightforward and one will find that the resulting laminar profiles have a maximum around  $\approx 7.5E - 02$ . This is certainly in the same neighborhood as the (hardly visible) maximum of  $7.36E - 02$  in figure 13.2 and there is no cause for alarm.

The presented initialization procedure is one of many possible options. In

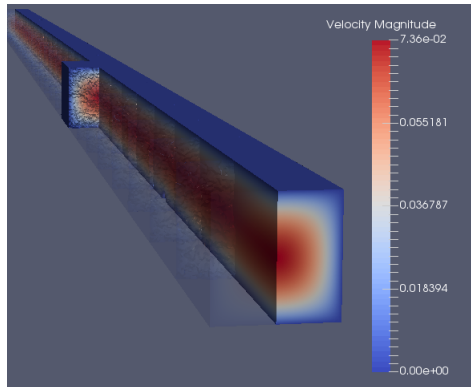


Figure 13.2: An overview of the entire duct being solved by 28 processors. The duct have been sliced down the middle. One half is transparent. One of the processors have been shown fully colored. A zoom of this processor can be seen in figure 13.3.

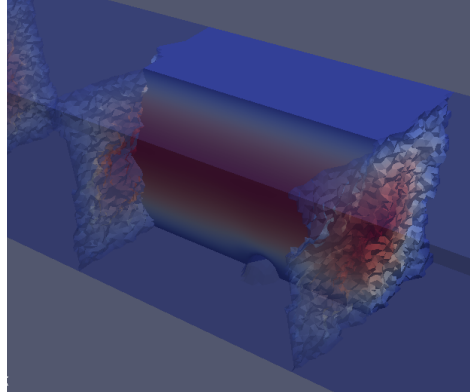


Figure 13.3: A close-up of one of the 28 processors solving the steady-state solution in the duct from figure 13.2. For the sake of interest the duct for the current processor have been cut in two so the bump shows clearly. Bump-locations are given in figure 13.1

the appendix (A.6) two other approaches are described as well as a further (functional) based analysis of the found steady state profile.

Obviously, there will still be a transient period where the  $u$ -profile changes to the flatter and fuller turbulent profile before we will be in a turbulent steady state but a considerable amount of simulation time has been avoided. In the end, our choice of initialization was made out of necessity. It could have been very interesting to see the entire transition starting from zero velocity, going up through the laminar range to finally transition to turbulence - but as seen in figure 13.4 this is very time-consuming. The figure shows the kinetic energy development for two simulations with identical parameters *but* with disparate initial conditions. Both simulations are still in transition towards a steady state that presumable lies midway between them. One is started from zero velocity. The other is started from the above introduced steady-state Stokes flow profile. Since the latter simulation quickly reaches turbulence (giving a higher friction coefficient) it takes much less time to dissipate the excess in energy. It should be mentioned that the simulation for the lower curve initially can benefit from a much higher  $\Delta t$  which can mitigate - but not

completely remedy - the extra cost in computation time. As mentioned in the very beginning of the "Initialization" section (13) it does indeed seem to be the case that around ten times as much iteration time needed judging by the figure.

A small remark on the computation of kinetic energy: Since we have an unstructured grid we must weight all nodal velocity values with the element volume. The correct approach in Oasis syntax is (see l. 93 in `TaylorGreen3D.py` in the Oasis source code):

```
kinetic = assemble(0.5*dot(u_, u_)*dx).
```

However, to save time we simply compute the kinetic energy with no concern of the element volume. This is of course an approximation and when we finally reach steady state we should use the above to improve accuracy.



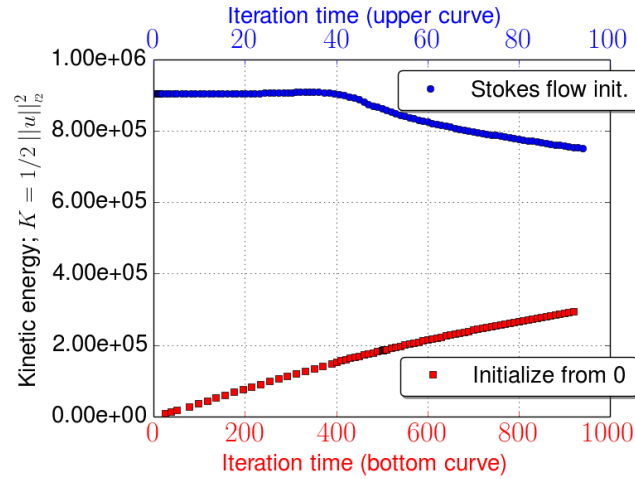


Figure 13.4: The kinetic energy development for two simulations: One has been initiated from zero-velocity, the other has been initiated from a steady-state Stokes flow profile. The simulations have the same forcing and should therefore end up at the same final value (both are still developing). Each curve has its own corresponding x-axis since the simulation time needed to reach a constant kinetic energy value when starting from zero velocity vastly dwarfs the time needed to relax a simulation from a steady-state profile. As a result, the lower x-axis is larger than the upper by a factor close to one order of magnitude. This is to be expected since the upper simulation quickly transitions to turbulence and thus have a higher friction coefficient. Thereby, the kinetic energy is much more rapidly dissipated. The metric indicating the kinetic energy,  $K$ , is first introduced properly in eq. 14.1.

## 13.4 Fenicstools

The FEniCS library, `fenicstools`, has already been mentioned several times and (invaluable) functions like `interpolate_nonmatching_mesh()` has been utilized. To conduct statistical measurements the library is indeed very laudable. It is not that the statistics cannot be obtained without it - it is simply a very lenient and elegant tool to use in this context. In the following we describe how we have made good use of it. Apart from the wikipedia-page the author does not know of written accessible information on the library and therefore a good part of knowledge has been gained simply by looking into the source code on the git which is also the reference the developer gives in the article presenting Oasis. This

is all to say: There might be more proper uses of the library than presented below. The given example is simply one way of obtaining the statistics.

To use `fenicstools` one must define a new function called `pre_solve_hook()` in the `problems` sub-module. Up until this point we have described the functions; `mesh()`, `create_bcs()`, `initialize()`, `body_force()` and finally `temporal_hook()`.

In the following section we will present velocity profiles which are obtained by probing along a line. In order to do this we define a `stats()` object in the `pre_solve_hook()` function:

```
def pre_solve_hook(V,u_,mesh,AssignedVectorFunction,newfolder,MPI,\
                    mpi_comm_world, **NS.namespace):
    if MPI.rank(mpi_comm_world())==0:makedirs(path.join(newfolder,"Stats"))
    from fenicstools import StatisticsProbes
    stats = StatisticsProbes(x.flatten(), V, True)
    return dict(stats=stats,x=x,plt=plt, np=np,h5l=h5l)
```

As seen, we load in the probe ability from the library and give it a coordinate container, `x`, for all points on the line. Since it is now in the dictionary, `dict()`, we can call it in the following `hook()` function called `temporal_hook()`:

```
if tstep % update_statistics == 0:
    stats(q-['u0'], q-['u1'], q-['u2'])
```

Now `stats.array()` will contain the data from probing along the line.

## 13.5 Mesh considerations

Before finally presenting the main results for the thesis we will make some observations related to the mesh we are using. Obviously, the finer the mesh, the better the results. How is the mesh designed? What regime can we actually hope to obtain meaningful results in, and what limitations will the mesh have? These are the questions we will answer below.

### 13.5.1 Mesh partitioning

The mesh structure has already been visualized in figure 13.2-13.3 where it is clearly seen how the 27 *M* cells in the mesh get distributed to the 28 processors. The entire duct is shown sliced down the middle except for one processor that is fully drawn. Assuming the steady-state script is

called `steady_unity.py` one simply opens a terminal and types:

```
mpiexec -n 28 python steady_unity.py
```

to solve for the steady-state profile. Through “`mpiexec -n 28`” we make sure the program is run in parallel on 28 processors. FEniCS subsequently partitions the entire mesh into 28 chunks which are then distributed to the 28 processors. In other words: the mesh partitioning is automatic. The only thing we have to do is to provide the number of processors. From the zoom in figure 13.3 we learn that the mesh chunks have a coarse grained wall separating one chunk from the next. This is a result of the actual elements in the mesh.

There are two meshes prepared for the comparison. One with  $\approx 11$  M elements and one with  $\approx 27$  M. Both are square pipes with dimension  $1 \times 1 \times 40$ . Further information is listed in table 13.2.

Table 13.2: **Information on the two duct meshes:**  $h_{min}$  is the minimum cell diameter which is computed as two times the circumradius. Similar definition for  $h_{max}$ .

<i>cells</i>	<i>vertices</i>	$h_{min}$	$h_{max}$
$26.59E + 06$	$5.11E + 06$	$1.13E - 02$	$5.68E - 02$
$10.88E + 06$	$2.35E + 06$	$1.03E - 02$	$1.53E - 01$

### 13.5.2 Mesh design

The mesh is designed after the expected friction Reynolds number,  $Re_\tau$ , of the simulation. The two primary simulations have already been described in tab. 13.1. As seen, the highest  $Re_\tau$  is close to 120. We can now estimate the average number of nodes the grid should have along the spanwise direction by choosing the number of points we would like to lie within the wall layer. The layer is approximately five times the viscous length,  $5\delta_0$ , and if we would like 2-3 points within this region we must choose our grid spacing,  $\Delta$ , accordingly:

$$\delta_0 \approx \frac{\Delta}{3} \quad (13.8)$$

We can now insert  $\delta_0$  into the definition of  $Re_\tau$  which for  $Re_\tau = 120$  gives:

$$Re_\tau = \frac{u_\tau H}{2\nu} = \frac{H}{2\delta_0} \Leftrightarrow \frac{\Delta}{H} = \frac{1}{80} \quad (13.9)$$

where  $H$  is the height of the duct as seen in figure 13.5. In other words, the (finest) mesh we have used has an average of 80 grid points in the spanwise

direction. For comparison; the most recent paper [12] referenced in section 7 on 'Onset of Turbulence' reports a domain resolution of  $48 < N < 96$  where  $N$  is number of grid points in the radial direction (pipe mesh) for a corresponding flow regime of:  $1900 < Re < 5500$ . In short, a comparable resolution. Perhaps more impressive is their mesh length  $L = 180D$  which allows them to quantify the turbulence much better than we can hope for with length  $L = 40D$ . A similar, impressive resolution is reported by Barkley et al [13] who presented a model on front dynamics we will return to when assessing our results. Even though their improved resolution can seem impressive it is worth remembering that they use spectral methods (and thereby have a global approach) whereas we utilize the FEM with a local approach. A classical spectral method would simply not work on our deformed mesh.

### 13.5.3 Grid limitations

"Direct numerical simulation of realistic turbulent flows belongs among the hardest computational problems." [20, p. 570]

- Benny Lautrup, 2011

Professor emeritus in theoretical physics, NBI.

To exemplify how artifacts can occur when the mesh gets too rough we now present some data from a simulation with  $Re_\tau \approx 180$  where the limitations of the grid can be recognized in figure 13.5. As stated in the literature, "the results depend inherently on the grid" [17, p. 420]. We clearly recognize the spikes/coarse grained nature of the flow particularly around the bump. Had we tried to do a classical Kolmogorov power spectrum - we would probably have ended up with a spectrum that would not be correctly resolved for the higher frequencies due to the mesh. Thus, knowing the limitations in the mesh is important when performing DNS; at least for the transitional regime  $2400 < Re < 2600$  (corresponding to  $96 < Re_\tau < 120$ ) where we will end up simulating we should have an adequate resolution.

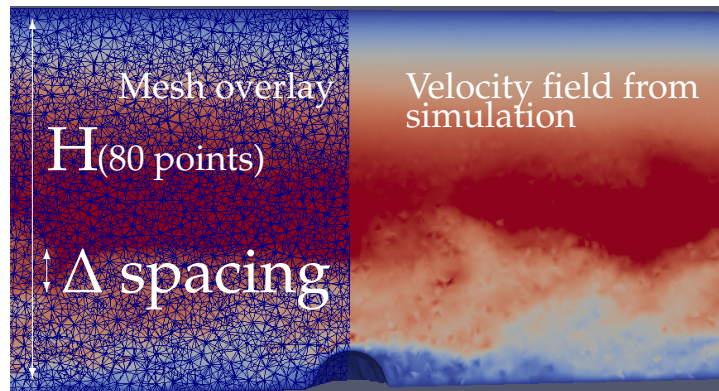


Figure 13.5: The figure shows the fineness of the unstructured mesh (blue, LHS). Also visible are spots in the velocity field near the bump which show the limitations of the grid. We cannot compute turbulence on scales below the mesh-scale limitations.

## Part V

### ONSET OF TURBULENCE

## TURBULENCE SIMULATION RESULTS

"I am an old man now, and when I die and go to Heaven there are two matters on which I hope for enlightenment. One is quantum electrodynamics and the other is the turbulent motion of fluids. And about the former I am really rather optimistic..."

- **Horace Lamb**

British fluid dynamicist



THE following sections: 14 Turbulence simulation results, 15 Comparison to the LBM and 16 Onset of turbulence are the main results of the thesis.

Before making any comparisons to the results from the LBM we will now present results from turbulence simulation results of the simulations on the duct-meshes. This is to ensure that we recognize expected turbulent characteristics and thus can produce meaningful, physically intelligible results.

Finally, we kindly remind the reader that the mesh and our parameters are dimensionless (tab. 13.1 and 13.2). We have already touched upon hydrodynamical similarity in section 4.4.1. The most obvious example of a solver usage with dimensionless parameters is of course the Oasis paper [16].

## 14.1 Profile assessment

To assess the simulated fluid motion we will now present the velocity profiles and scrutinize them for turbulent characteristics. Since both simulations started from a steady Stokes flow profile we expect the very first profile to be laminar. Then after a transient period where turbulence sets in we should end up in a (quasi) steady state with a profile mixed of laminar and turbulent flow if we get far enough down in the transitional region. We write quasi steady state because the kinetic energy might fluctuate a bit since turbulence suddenly arises in localized puffs or slugs from time to time - but *on average* the energy should be constant, i.e. it should fluctuate around some mean value since the forcing is constant. This system state is also known as a *Statistical equilibrium* [20, p. 571].

### 14.1.1 Profiles

To nurture the reader's intuition of this profile-transition we plot data from the two simulations in figure 14.1(a) and 14.1(b) for  $Re_\tau = 96$  and  $Re_\tau = 117$  respectively ( $Re_\tau$  refers to the *final* state). The profiles have been probed along the line:  $[x, y, z] = [0.5, y_i, 20] \forall y_i = [0, 1]$ .

In both profile figures the initial (gray) profile is seen to be clearly laminar in shape. All subsequent profiles as time progressed have been plotted in increasingly darker shades of blue. It is here worth noting that the turbulence sets in on the "left" side, i.e. the  $xz$ -plane where the deformation bumps are located. This is to be expected. The final profile from the most recent iteration is shown in red. The profile shows clear turbulent characteristics and is much flatter and fuller in shape compared to the laminar initial state. To estimate the mean value profile for the turbulent fluid we take the mean of all profiles *after* a given time. The resulting profile is shown in black. Also this profile has the shape one would expect. There is one noticeable difference particularly evident in figure 14.1(a): The profile is a bit skewed towards the plane with deformation bumps. This could be due to the fact that the turbulence for figure 14.1(b) is more fully developed and therefore better smears out the effect of the bump. We will later on see that the bumps are particularly easy to detect for the LBM profiles which are just on the cusp of transitioning to turbulence from below. In our case, the bump is not as easy to distinguish and for the simulation with highest  $Re$  (fig. 14.1(b)) it is indistinguishable. All the above would agree with a  $Re$ -dependence. Thus, again we find the results to be in agreement with what one could expect. It is these black



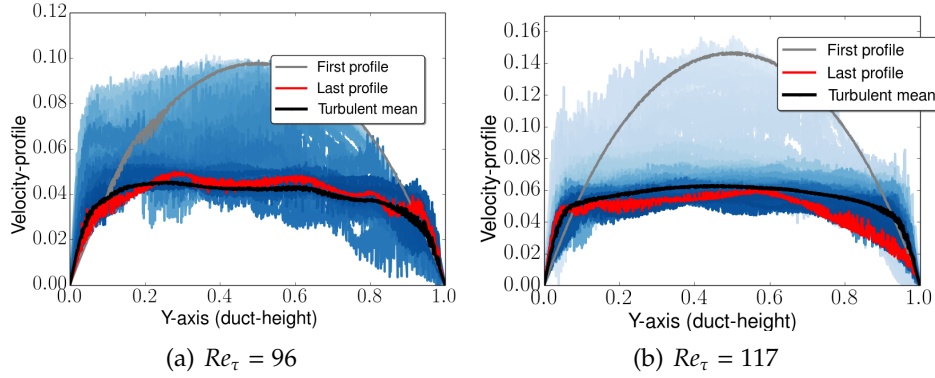


Figure 14.1: Velocity profiles along the line  $[x, y, z] = [0.5, y_i, 20] \forall y_i = [0, 1]$ . Initial Stokes flow profile is seen in gray. All subsequent profiles are shown in increasingly darker shades of blue. The profile from the most recent iteration is shown in red and a mean turbulent profile in black. The averaging of turbulent profiles is after a certain iteration - all used data is indicated as colorless markers in figure 14.2. None of the simulations are at steady state yet: a)  $Re = 2400$  and b)  $Re = 2600$ . For the shown data: a) The  $Re_\tau = 96$  simulation starts at  $Re = 5555$  and falls to  $Re = 4400$ . b) The  $Re_\tau = 117$  simulation starts at  $Re = 8333$  and falls to  $Re = 5500$ .

mean-profiles we later on will compare to their equivalents from the LBM simulations.

### 14.1.2 Kinetic energy

To further examine the transition from the initial laminar profile to a turbulent profile we compute the kinetic energy of the fluid (with *unit fluid density*,  $\rho = 1$ ). As a metric of interest we follow FEniCS-literature [17, p. 409] as well as Simo and Armero [46, eq. 2.11] and use the functional,  $K(u)$ , to describe the kinetic energy:

$$K(u) = \frac{1}{2} \int_{\Omega} \|u\|^2 d\Omega. \quad (14.1)$$

The resulting energy curves for both simulations are seen in figure 14.2. The figure clearly shows that both simulations start in a very high kinetic energy state and as soon as turbulence kicks in the friction increases and energy is dissipated from the system. Although pipes have been shown to remain in a laminar state up to around  $Re \approx 10^5$  the emergence of turbulence is certainly as expected since the duct in our case also is

deformed. The onset of turbulence is most apparent in the (blue)  $Re_\tau = 117$  curve. This can be attributed to the choice of  $\Delta t$  which was kept at  $\Delta t = 0.01$  whereas the  $Re_\tau = 96$  simulation had  $\Delta t = 0.1$  the first 10.000 iterations. This resulted in the first 10 data points (we sample every thousandth iteration) being differently dispersed than the remaining data. These first 10 data points have been left out of the figure but a red diamond marks the original starting level for the red curve. A plot showing all of the data can be found in the appendix A.26.

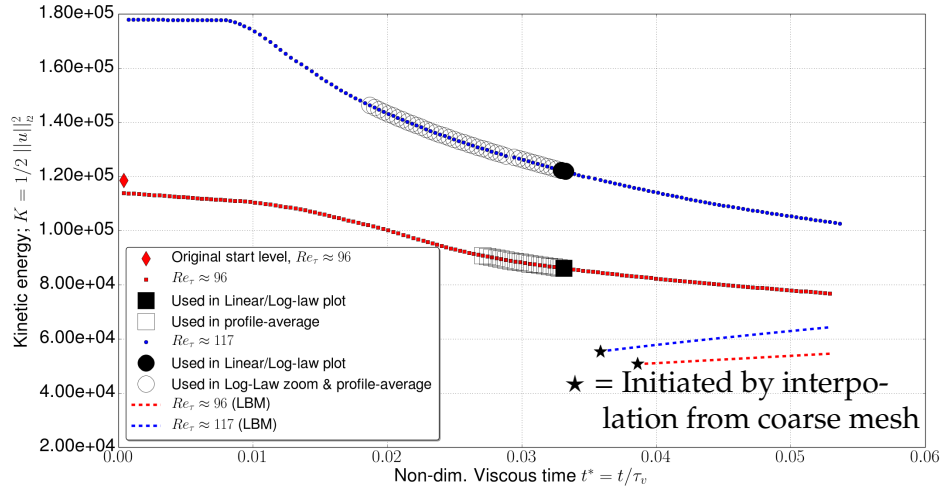


Figure 14.2: Kinetic energy for  $Re_\tau = 96$  (red) and  $Re_\tau = 117$  (blue).  $Re_\tau$  values refer to the *final* statistically steady states. The clearly visible kink in the blue curve resembles the onset of turbulence in the duct. A larger initial  $\Delta t$  for 10 first time-steps for the red curve may unfortunately have smeared the effect. The ten first time-steps have been cut out but the red diamond marks the true kinetic start-value for the red curve (see fig. A.26). Empty, colorless markers represent data used to produce black mean profiles in figure 14.1(a) and 14.1(b). Black markers show data used in figure 14.4(a) and 14.4(b). The measure for kinetic energy is described in eq. 14.1.

As seen, the curves are gradually leveling towards a constant line signifying that the simulations are approaching statistically steady states. The energy we pump into the system (the forcing) should on average in the end be balanced by the dissipative forces. The colorless markers refer to the time steps where we average the profiles to obtain the black curves in figure 14.1(a) and 14.1(b). In figure 14.2 black markers can also

be seen. These signify the iterations (1.000 per marker) which have been used to make plots of comparisons to both the law of the wall and the logarithmic law in the next section. Finally, two dashed lines show the current energy for the  $Re_\tau = 96$  (red, dashed) and  $Re_\tau = 117$  (blue, dashed) LBM simulations we will make comparisons to. As seen, there is still a considerable discrepancy in energy between Oasis and LBM simulations.

### 14.1.3 The law of the wall & the logarithmic law

Up until this point data analysis have been very qualitative. All investigations have resulted in reasonable answers which clearly were to be expected but no falsifiable tests have been carried out.

In order to check if the simulations entail true physics abiding the laws as we know them we now compare the (half) profile to the famous law of the wall and the logarithmic law (re)stated below for convenience.

$$\frac{u}{u_\tau} = \frac{yu_\tau}{\nu} \quad (14.2)$$

$$\frac{u}{u_\tau} = \frac{1}{\kappa} \ln\left(\frac{yu_\tau}{\nu}\right) + B, \quad (14.3)$$

where the *von Kármán constant* is  $\kappa = 0.4$  and the ordinate intersection constant is  $B = 5.5$ . These constants are identical to those used by Kim et al. [33] who favorably compared their DNS data from channel flow ( $Re = 3300$ ) to the two laws. The viscosity is kept at  $\nu = 0.9E-5$  throughout the entire computation (see. tab. 13.1).

There is a caveat to this investigation. Due to the extensive amount of computation time needed we cannot first saturate the simulation at higher Reynolds numbers and wait for statistical equilibrium to carry out precise measurements over a longer time span before finally lowering the Reynolds number to the transitional regime. There simply has not been time for this. Thus, care must be taken when e.g. rescaling to friction units. It should also be mentioned that the two simulations were around  $Re \approx 4500$  and  $Re \approx 5500$  at the time of investigation (black markers fig. 14.2).

What can be expected as outcome of such a comparison? As noted above Kim et al. [33] showed very close agreement several decades ago between the two laws and DNS results using a spectral method. Their friction Reynolds number was  $Re_\tau = 180$ , which corresponded to  $Re = 3300$  - well below the two Oasis simulations'  $Re$  at investigation time. Thus,

in principle one could expect a close agreement in our case. On the other hand it must be stressed that whereas they obtained results from a simulation in statistically steady state we certainly are not in that situation.

### Friction units

To rescale the profile to friction units we need to estimate the wall shear stress,  $\tau_w$ . The relevant lines of (one way) to calculate  $\tau_w$  in FEniCS have already been presented. We have tested several methods on Oasis due to the immense size of the mesh but it has simply been infeasible to carry out the calculation for the complete stress tensor field,  $\sigma_{ij}$ . Instead, we have approximated the wall shear stress by computing the average stream-wise velocity in the (black) top slab;  $\bar{u}_{zslab}$  in figure 14.3. We then approximate the wall stress as:

$$\tau_w \approx \mu \frac{\bar{u}_{zslab}}{(\Delta y)/2}. \quad (14.4)$$

When estimating the shear stress it is obviously a good question how thick the slab should be in the  $y$ -direction. To make a justified guess one can test the range  $\Delta y = [0, 0.05]$ . The resulting viscous length scale,  $\delta_0 = (\nu/u_\tau)$ , is in this range rather close to  $\delta_0 = 0.004$  and since we know from literature that the viscous sublayer is around  $\delta_{sublayer} \approx 5\delta_0$  [34, p. 272] we have chosen the slab thickness:  $\Delta y = 0.004 \cdot 5 = 0.02$ .

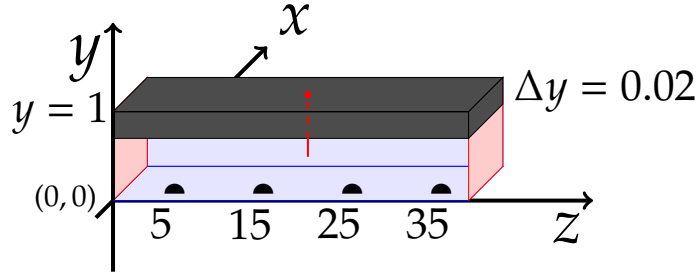


Figure 14.3: Estimation of the average wall shear stress at the  $xz$ -plane for  $y = 1$ . Computing the average stream-wise velocity in the slab;  $\bar{u}_{zslab}$ , we approximate the wall stress as:  $\tau_w \approx \mu \frac{\bar{u}_{zslab}}{(\Delta y)/2}$ . The red line shows the points we have probed for stream-wise velocity generating data seen in figure 14.4(a) and 14.4(b).

Once we have the shear stress we can compute the friction velocity,  $u_\tau = \sqrt{\tau_w/\rho}$ , and finally plot the two laws (eq. 14.2 and 14.3) in a semi-logarithmic figure along with the data. The two resulting plots are seen in

figure 14.4(a) and 14.4(b) for  $Re_\tau = 96$  and  $Re_\tau = 117$  respectively.

The two comparison-plots are very telling of the physics at play. First of all we notice that by first glance the scaling to friction units have gone smoothly. The beauty of nondimensionalized units is that these plots look exactly the same in every textbook or paper. A typical plot ranges from  $u^+ = [0, 25]$  on the vertical axis whereas the  $y^+$ -axis is limited by the setup in consideration. Reassuringly, both figure 14.4(a) and 14.4(b) have  $u^+$ -axis in this range. One should generally find the three outer layers mentioned in part II at the regions:

1. Viscous sublayer (Law-of-the-Wall);  $0 < y^+ < 5$
2. Buffer layer;  $5 < y^+ < 30$
3. Inertial sublayer (Logarithmic law);  $30 < y^+$  .

The universality of these semi-logarithmic turbulent profile law plots make it very easy to compare across setups and is a good test for validating fluid motion.

Starting from the wall; the velocity profile in figure 14.4(a) follows the law of the wall very closely indeed. There does seem to be a slight systematic tendency to overshoot the law value but otherwise it is in very good agreement with the linear relation. The systematic overestimation is most likely due to the way we have estimated the wall shear stress. The curve itself is based on averaging 10 probings (every 100<sup>th</sup> iteration) along the red line seen in figure 14.3. The shear stress however is estimated as the mean of shear approximations at the saved  $u$ -field just before the 10 probings and just after (we save  $u$ -fields every 1.000 iterations). The point is: should the shear stress estimate,  $\tau_w$ , be a bit too low it will result in too low a friction velocity,  $u_\tau$ . Therefore, all  $u^+ = u/u_\tau$  values will be shifted a bit upwards. Overall however, the correspondence between numerical result and theoretical predicted value is striking. Keep in mind that there has been no other offsetting or fitting performed. The blue curve is simply the measured velocities in the duct presented in friction units.

One can also note the almost simultaneous shift in nature from a *convex* to a *concave* curve. This means, that the (start of the) buffer layer and thereby the transition from one law to the other is very well predicted.

Finally, there is the inertial sublayer and the logarithmic law. In this region, the blue curve is clearly off. As mentioned above, much better agreement for the logarithmic law has been shown for comparable  $Re$ . A reasonable guess would be that this discrepancy could be due to the lack of statistical equilibrium. It is perhaps noteworthy that it seems as if

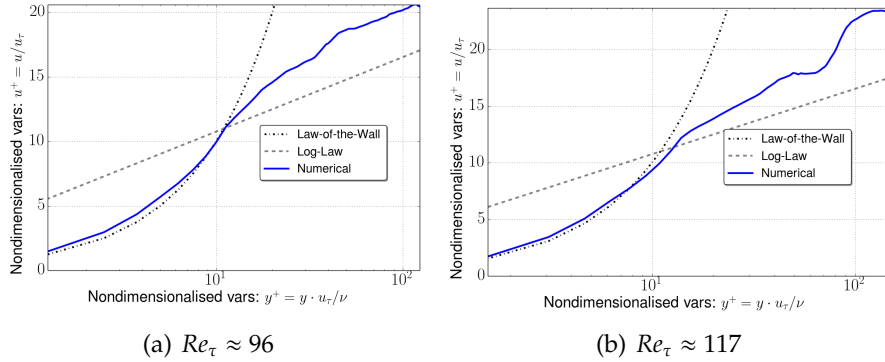


Figure 14.4: Comparison of data measured along the red line in figure 14.3 to the law of the wall (eq. 14.2) as well as the logarithmic law (eq. 14.3). The corresponding time step during the simulation is visualized in figure 14.2 as black markers. A close agreement between the law of the wall and the numerical results from the simulation can be observed in the wall layer. b) After the transition from buffer layer to inertial layer the slope features are smeared by turbulence. Therefore, a (zoom) plot has been made in figure 14.5 where we have scanned a large amount of data (colorless circles fig. 14.2) to minimize effects from turbulent fluctuations.

ultimately the blue slope is parallel to the logarithmic law. In other words, the *von Karman* constant,  $\kappa$ , seems to be well captured by the simulation. It is simply the  $B$  constant in eq. 14.3 that has changed.

A perhaps equally relevant guess when explaining the discrepancy in the inertial layer is found in the paper presenting Oasis [16]. Here, Mortensen et al. present (also slightly overshooting) mean velocity profiles for two mesh resolutions. They show that as the mesh is refined the overshooting is reduced and the profile converges towards their reference curve (taken from [18]). It is here pointed out that underresolved simulations underpredict the dissipation which results in velocities that are too high. Thus, had time allowed it there are at least two obvious ways of improving the measurement: (a) to measure at a statistically steady state and (b) to change mesh resolution. It is actually not a complete surprise that the mesh resolution could be a problem. As stated in section (13.5) the mesh is designed for measurements in the transitional regime at Reynolds numbers well below those characterizing the fluid in this investigation.

Much the same can be said of figure 14.4(b) as was said for figure 14.4(a) and just as before, black markers in figure 14.2 show the data used

in figure 14.4(b). There are however a few new noticeable novelties. We notice that the transition from the wall layer to the inertial layer is perhaps slightly more off compared to the previous figure but more importantly, clear turbulent artifacts have smeared the profile in the inertial region,  $30 \lesssim y^+$ . From the figure we can hardly distinguish the actual slope in this region due to the limited amount of samples we compute the mean from. To circumvent this problem we have scanned through a large amount of the data marked as colorless circles in figure 14.2. For each marker (1000 iterations) we re-estimate the wall stress before taking the mean of the profiles which can finally be seen in the zoom-plot in figure 14.5. Notice, that a change in shear stress will change the viscous length scale,  $\delta_0$ , and thereby slightly change the  $y^+$ -positions for each grid point. This results in some small spikes along the curve close to the buffer layer.

As seen, the offset in the transition seems to be improved now and resembles more closely the transition seen in figure 14.4(a) where the data intersect the crossing of the two laws. The final slope is also now easier to assess compared to the one in figure 14.4(b). Clearly, once the transition from buffer layer to inertial layer is complete we end up with a slope closely resembling the *von Karman* slope typical for an inertial layer.

In conclusion, the velocity profiles from both simulations seem to be on a par with what can be expected from turbulent motions of fluids and several physical traits which are characteristic for turbulence have been observed. The profile has the flatter, fuller characteristic shape compared to laminar profiles (fig. 14.1(a)-14.1(b)) and most importantly, clear turbulent features inherent to different layers with corresponding laws dictating the slope have been favorably compared to our numerical measurements (fig. 14.4(a)-14.5) apart from an overestimation of the  $B$  constant in the logarithmic law. This discrepancy is re-assessed in the discussion (sec. 17.1).

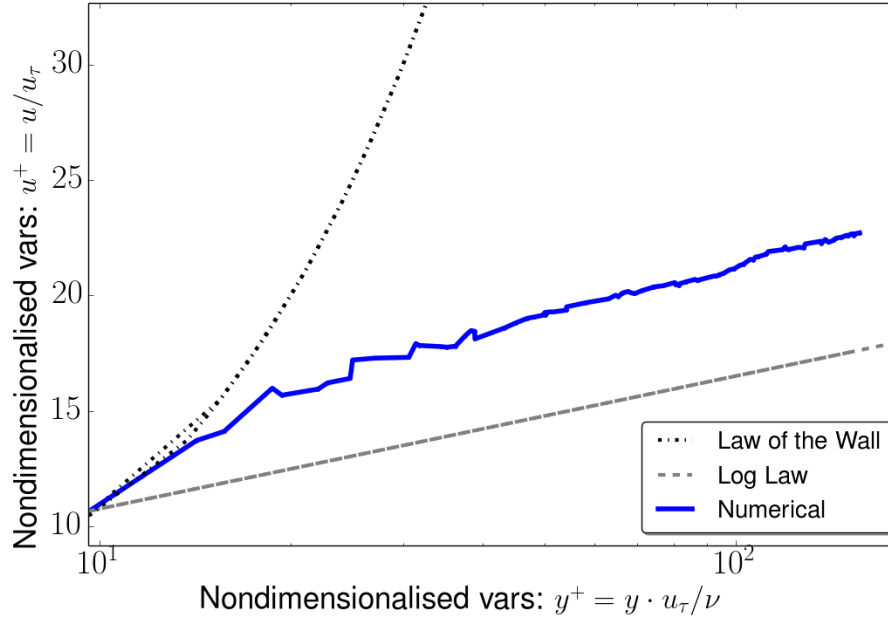


Figure 14.5: An attempt to smooth the curve of figure 14.4(b) by looping through a vast time span. Since the wall shear stress notably changes so does the friction velocity,  $u_\tau$ , making it difficult to obtain a mean value at a given point. The curve does however seem less fluctuating when compared to figure 14.4(b). The two profile laws have been drawn using the characteristic viscous length,  $\delta_0$ , corresponding to the very last iteration. We learn in this plot that the slope in the logarithmic plot is (approximately) the same as for the actual law which is reassuring. Relating this to the logarithmic law, it is only the  $B$  constant that changes. The *von Kármán* constant,  $\kappa$ , stays the same. The time span of the simulation used in this zoom is shown in figure 14.2 as colorless circles.



## COMPARISON TO THE LATTICE BOLTZMANN METHOD



COMPARING computational methods can be a very delicate endeavor indeed since it can be difficult to find neutral metrics that do not favor one side over the other. An advantage is that the meshes used are identical and that the server used in the comparison is the very same one. An uncertainty on the other hand is that the server performance depends on the workload. We have tried to keep it steady throughout the investigation to minimize this effect.

### 15.1 Motivation for comparison

We have already touched upon the motivational aspect in section 3.1. The most obvious reason for making a comparison to the LBM is that the implementation used at NBI has not been tested yet, at least not in the turbulent regime. A test similar to the analytical recovery test of the unsteady Hagen-Poiseuille flow we presented in part IV for the Chorin-solver has been carried out (see fig. 11.4). Thus, the laminar region has been covered [48] but an iteration scheme can easily be well-functioning in one region only to break down in another one. Since the convective term and thus the nonlinear forces in the turbulent regime are very dominating some (in)abilities for a solver could be revealed.

## 15.2 Metrics of comparison

The original design of the comparison was to look at selected turbulent statistics and investigate if the same statistical results can be obtained with a higher  $\Delta t$  with the Oasis-solver than for the LBM. This is interesting since the limit for  $\Delta t$  for the LBM implementation is around  $\Delta t \approx 0.001$  whereas the Oasis-solver merely is restricted by the physics in question (e.g. characteristic time-scales due to viscosity) and therefore often can use a much higher  $\Delta t$ . Back in section 6.1.2 we made a (rough) estimate for  $\Delta t \approx 0.2$  as an upper limit to the time-step at  $Re = 2600$  which should be close to the final state for the  $Re_\tau = 117$  simulation (tab. 13.1). This is naturally only an estimate but the point is that one probably can obtain meaningful statistical results with a much higher  $\Delta t$  than the LBM limit. Thus, the original goal for a comparison was (apart from a general verification of the LBM) to (a) show that similar statistics could be obtained with a higher  $\Delta t$  and (b) locate the upper  $\Delta t$ -limit above which the physics would not be accurate.

As is evident from the kinetic energy plot in figure 14.2 the Oasis simulations are still not in a statistically steady state and the corresponding LBM simulations are certainly still well below in kinetic energy. What is perhaps worse is that the two LBM simulations are *also* not in a statistically steady state and there currently is not enough computational capacity at NBI to run them up to steady-state. Thus, a complete one to one verification cannot be made right now. What we can do is to show that the slopes for the Oasis simulations are converging towards the LBM profiles even if the statistically steady state is somewhere in between the two. First of all however, we will look at the performance of the two implementations. How much computation time is needed per iteration for a given mesh? And perhaps more interesting: How does the computation time-scale with the number of processors utilized?

The computational performance is measured in (actual) computation time per 1000 iterations. The following comparison is carried out with a  $\Delta t$  for Oasis ten times larger than for the LBM, i.e. for Oasis  $\Delta t = 0.01$ .

## 15.3 High-performance computing

Within Computer Sciences there are (at least) two distinct notions of scalability. Either one can fix the total problem size and then measure the computation time as the number of processors is varied (*Strong Scaling*) or one can fix the problem size per processor and then measure the

computation time as the number of processors are varied (*Weak Scaling*). To give an example of the former, benchmark tests have been conducted on the coarse duct mesh ( $\approx 11\text{M}$  elements) as well as on the fine duct mesh ( $\approx 27\text{M}$  elements). As seen in figure 15.1 the scaling for the Oasis-solver is not far from the ideal scenario (black, dashed) both for the coarse (red, dashed) and the fine (blue, dashed) mesh.

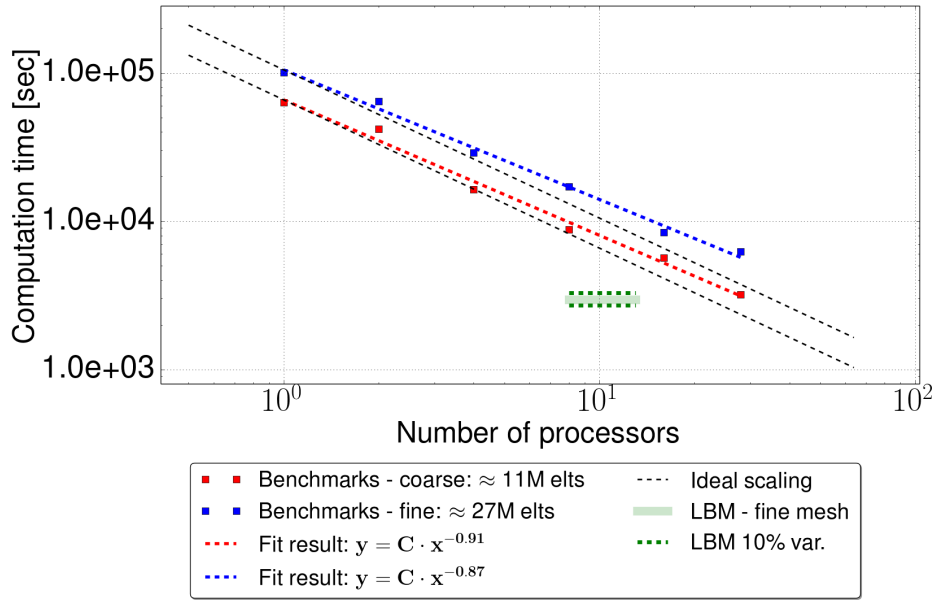


Figure 15.1: Benchmark results and fits to strong scaling law. Every point marks the computation time for one thousand iterations. As seen the Oasis-scaling is not far from the ideal scenario. The LBM is currently running on 8-12 processors and since computation time is almost constant ( $\pm 10\%$ ) for this implementation it is marked with a green bar with 10% confidence lines (green, dashed).

As the figure shows, the LBM (green) is faster on the finest mesh than the Oasis-solver (blue, dashed) by a factor of  $\approx 2$ , at least for the number of processors currently available (28). The LBM has spatial and temporal convergence:  $O(h^2)$  and  $O(\Delta t^2)$  respectively, and we have chosen a linear element for Oasis to match these convergence rates. Clearly, the Oasis-solver shows a scalable dependence on the number of processors that cannot be seen for LBM as it is currently implemented. If this is not improved, choosing the fastest method depends on the number of processors available. Interestingly, since the Oasis-solver runs with a

time-step that is a factor 10 larger it actually simulates the fluid 5 times as fast as the LBM *if* the statistics obtained can be deemed intelligible. Should the previous estimate of a maximal time-step of  $\Delta t = 0.2$  hold true, the Oasis-solver would simulate a factor 100 faster than the LBM. The final test cannot as mentioned be made yet as simulations are currently still under way.

## 15.4 Profile comparisons

A first profile comparison between Oasis and LBM can be seen in figure 15.2. For LBM three profiles are shown:  $Re_\tau = 96$  (red),  $Re_\tau = 117$  (blue) and a slightly stronger forcing corresponding to  $Re_\tau = 127$  (green, dashed). As expected, the  $Re_\tau = 96$  certainly looks most laminar. For Oasis two profiles are shown for both simulations with  $Re_\tau = 96$  and  $Re_\tau = 117$ . An early profile (transparent) and a profile of a more recent sampling (transparent, dashed) are shown. Both simulations can be seen to gradually loose velocity and converge inwards towards the LBM profiles. Of particular interest is the LBM  $Re_\tau = 127$  (green, dashed) and the Oasis  $Re_\tau = 96$  (red, dashed) profile. We included the LBM  $Re_\tau = 127$  simulation as it has been running for a considerable amount of time and should be closer to its steady state. This simulation has not been introduced previously and was not part of the original comparison-design. Since it is closer to steady state we can get a better glimpse of what a steady-state LBM profile would look like. It seems that at this stage of the convergence the Oasis  $Re_\tau = 96$ -profile is quite comparable to the green, dashed one meaning that it does seem like the two methods can produce similar turbulence (green, dashed and red, dashed clearly have similar form).

To better check the proportions between turbulent and laminar effects in the profiles we plot them on normalized axes. We include an exact laminar profile for comparison. Here we clearly see that the  $Re_\tau = 96$  LBM (red) is very close to a laminar profile and should still be far from its steady state. The  $Re_\tau = 117$  LBM (blue) profile is clearly more turbulent (we have excluded previous laminar data) but there is simply not enough (turbulent) data ready as of this moment to obtain a smooth profile without turbulent artefacts. In the figure we can now clearly tell how turbulent a profile is simply by counting from the left. The outmost profile is the early  $Re_\tau = 117$  Oasis (blue, transparent) line closely followed by the early  $Re_\tau = 96$  Oasis (red, transparent) line just as one would expect. After these we see their later versions before the LBM lines. One last note of interest: There is a clear asymmetry in all profiles

but the  $Re_\tau = 96$  LBM (red) and the completely laminar line. Thus, we easily recognize the bump in the figures just as for the previous investigations.

Seeing that the initial Oasis and LBM simulations were so far apart they have recently been put to a halt in order to pursue a comparison with the more developed  $Re_\tau = 127$  LBM simulation (green, dashed in figure 15.2). Thus, two new Oasis simulations were initiated with a forcing corresponding to  $Re_\tau = 96$  and to  $Re_\tau = 127$  respectively, both with  $\Delta t = 0.1$  which is 100 times the LBM  $\Delta t$ . They were started from the final state of the 'old'  $Re_\tau = 96$  simulation since it seemed in figure 15.2 to match in profile with the green, dashed  $Re_\tau = 127$  LBM ditto. The two new Oasis simulations can be seen in figure 15.4 as red and green lines respectively (the lines have circle markers clearly showing the larger  $\Delta t$ ).

In figure 15.4 one can easily see that the  $Re_\tau = 127$  LBM (green, dashed) simulation has been running for a longer time compared to the other LBM curves. Hence, it would be natural if it was closer to steady-state. Finally, the two  $Re_\tau = 127$  simulations have been compared in figure 15.5. The LBM simulation is green, dashed both in figure 15.4 and 15.5. Since there still is a minor gap in figure 15.4 we cannot expect to see a complete agreement which also is not the case in figure 15.5. It does however make sense that the Oasis profile seems more turbulent since it according to figure 15.4 is more energetic.

In conclusion, it has generally been shown that the lack of steady-state in the simulations (not surprisingly) results in a lack of complete agreement between the profiles. We have however seen a very close match in figure 15.2 between two profiles (the red and green dashed profiles). The explanation can be found in figure 15.4 where we have plotted the kinetic energy at the corresponding time-step of the simulations ( $Re_\tau = 96$  Oasis and  $Re_\tau = 127$  LBM) as red stars. As seen, the red star values are very close indeed. We have thus reason to suspect that a good, complete agreement will be obtained in steady state.

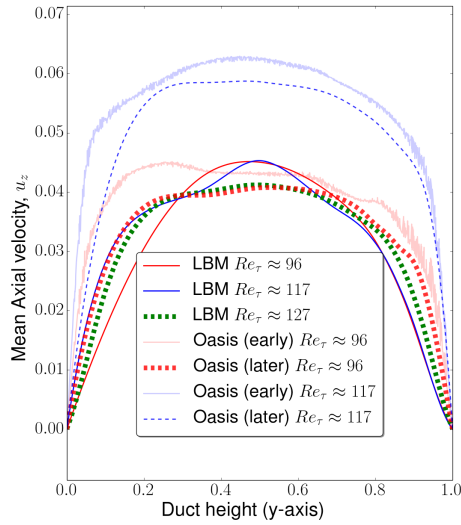


Figure 15.2: Profile comparison: For LBM three profiles are shown:  $Re_\tau = 96$  (red),  $Re_\tau = 117$  (blue) and a slightly stronger forcing corresponding to  $Re_\tau = 127$  (dashed, green). For Oasis, profiles at an early (transparent) and a later (i.e. more recent) time (transparent, dashed) are shown to visualize the convergence towards statistical equilibrium.

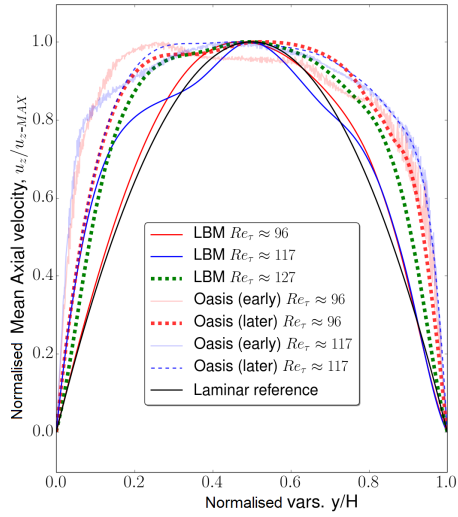


Figure 15.3: Profiles comparison in a rescaled dimensionless plot. Here we can clearly see the proportions of the turbulent and laminar effects for each simulation. The data is completely the same as in figure 15.2 only now all profiles are normalized.

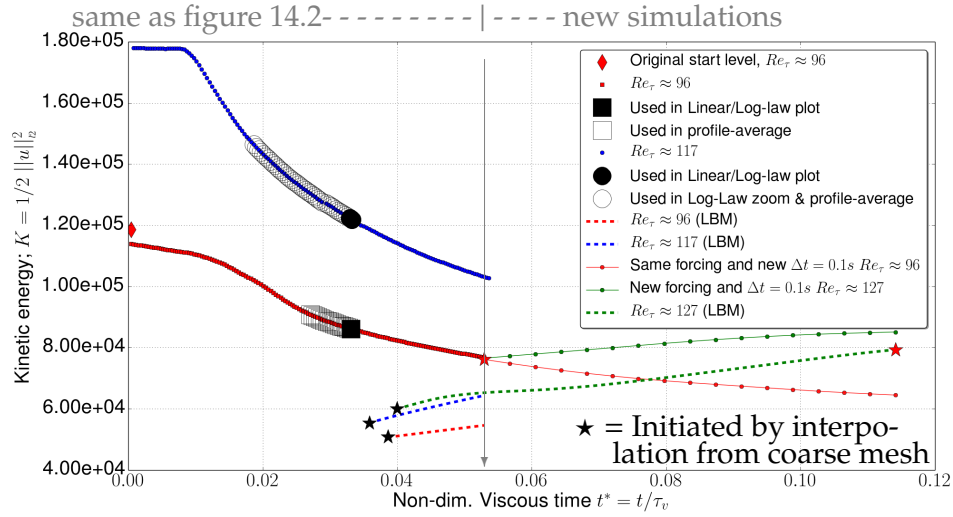


Figure 15.4: This figure is an extension of figure 14.2. New simulations are the three lowest entries in the legend. As seen, the two  $Re_\tau = 127$  simulations (green) are closer than the other simulations (blue, red). The resulting profile match is given in figure 15.5. The  $Re_\tau = 96$  simulation (now with  $\Delta t = 0.1$ ) seems close to steady state. The measure for kinetic energy is described in eq. 14.1.

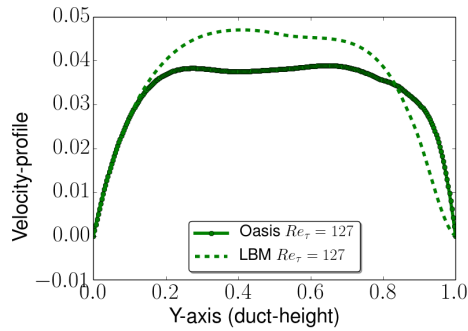


Figure 15.5: A final profile comparison stemming from the most recent green data in figure 15.4. Despite agreement near  $y = 0$  the Oasis profile clearly is still more turbulent. This is to be expected since the kinetic energy (and thereby velocity) is higher as seen in fig. 15.4.

## ONSET OF TURBULENCE

**F**INALLY, we present results from measurements of speed and structure of turbulent fronts in duct flow. These will be related to results from the current research field which was introduced in sec. 7.

Characterizing turbulent intensity is often taken as the kinetic energy in the spanwise plane. Thus, as a metric for the level of turbulence intensity in the flow we use:

$$q^2 = \int_{A_c} u_x^2 + u_y^2 dS, \quad (16.1)$$

as do e.g. [13, 12, 14]. First however, a word of caution: Clearly, the Stokes flow profile we use to initialize cannot be deemed turbulent - yet one will find four maxima when  $q^2$  is computed for the profile. This is of course nothing but an artifact from the four deformation bumps in the mesh. Figure 16.1 visualizes the conundrum. One could easily subtract this from all subsequent measurements. To not tamper too much with the data we have instead chosen to impose a threshold well above these bump-intensities when we wish to track the slug-edge fronts.



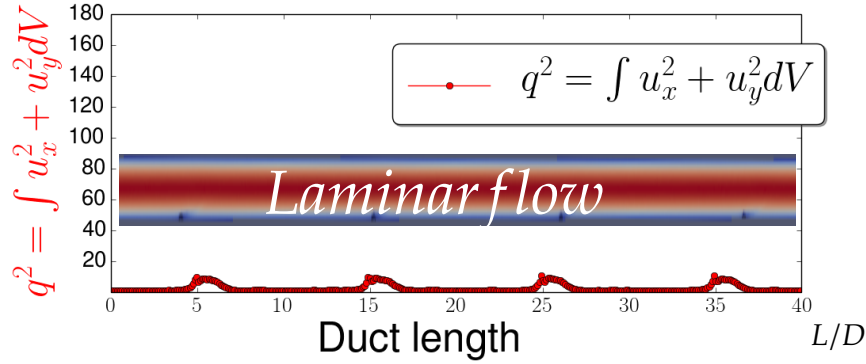


Figure 16.1: Clearly, the metric for turbulence intensity;  $q^2(z) = \int u_x^2 + u_y^2 dV$ , gives a maximum for each bump which is *not* turbulence. In the figure an inset showing the velocity field for the entire duct (scaled) is included. The inset shows the very first frame of the simulation for  $Re_\tau \approx 117$ . Obviously, it is laminar, since this simply is the steady Stokes flow right after scaling to the desired forcing. The turbulence-indicator,  $q^2(z)$ , has been described further in eq. 16.1. We write  $\int dV$  here since we have partitioned the mesh in small slices each with some thickness. Therefore the slice really has a volume and is not just a cross-sectional area,  $dS$ .

## 16.1 Front dynamics

To measure and subsequently compare the slug characteristics for both simulations we measure slug edge speed and look at the characteristic shape of the slug.

### 16.1.1 Strong slugs

At the initialization we see a sudden, rapid transition from laminar initial profile to full turbulence (see e.g. the kink of the blue curve in fig. 15.4). If the puff - weak slug - strong slug classes are amenable at all, we certainly would expect the characteristics to mirror those of the strong slugs. As pointed out by Song et al. [12] a reasonable rule of thumb for periodic meshes compared to experimental setups is that we typically have longer observation times for slowly expanding (weak) slugs *but* shorter time for strong slugs (as these fill the mesh rapidly). This is actually very important in the slug assessment we will carry out as we are clearly in the strong slug regime. Interestingly, Song et al. have meshes of length  $L = 180D$

where we have  $L = 40D$ . Thus, whatever estimates we obtain are very limited by our setup. Furthermore Song et al. [12] note that measuring structures of length 15-20 diameters shorter than the mesh eventually leads to artificial (self-)interaction by the structure as the mesh fills completely with turbulence (the recovery length for complete relaminarization is  $\approx 20D$ ). Thus, any results should be interpreted cautiously in this first case as we quickly surpass this threshold.

### Edge speed

To track the slug edges we impose a threshold (dashed, blue) as seen in figure 16.2 (LHS). The red  $q^2$ -curve clearly intersects the threshold in several places (marked as transparent blue squares). Using a clustering algorithm we end up with (in this case) six fronts and choose the leftmost and rightmost front for the slug as the upstream and downstream front respectively. Both fronts are marked as fully drawn blue squares (downstream has a cross). The inset above shows the corresponding velocity profile at the given time-step. Furthermore, the mean centerline velocity is shown. As explained in sec. 7 the two curves mirror each other (see e.g. figure 7.2).

The two fronts are tracked until the strong slug almost fills the duct (fig. 16.2 RHS). The front speeds,  $U_f$ , are usually reported as the ratio with the mean flow speed:  $c = U_f/u_{avg}$ . The final front speed results for both simulations are seen in table 16.1 along with some values from the literature for the sake of comparison. To be clear with respect to procedure an example of computation of the upstream front speed for the  $Re_\tau = 117$  simulation is given: As seen in the legends of figure 16.2 LHS and RHS the upstream front moves  $16.8 - 14.3 = 2.5$ . Since the frames are 8000 iterations apart (and  $\Delta t = 0.01$ ) the front speed is:  $2.5/(8000 \cdot 0.01) = 0.03125$ . Now, to get the speed-ratio we divide with the mean flow speed which is  $u_{avg} = 0.075$  to get  $c = (0.03125)/(0.075) \approx 0.4$  which is the value listed in table 16.1 for the  $Re_\tau = 117$  simulation.

As seen, it has not been feasible to estimate the upstream edge speed for  $Re_\tau \approx 96$  since the edge seems to be standing still - or at least seems to fluctuate around the same position right behind one of the deformation bumps ( $z = 5$ ). This is illustrated in figure 16.3 where the final upstream edge front (black) is behind(!) the first upstream edge front (blue). The intermediate frames are shown in red. The problem is not just that one would obtain a negative speed - but that the speed here clearly depends on the choice of frames. The red intermediate frames can be seen on either side of the initial (blue) edge. Thus, it fluctuates back and forth and

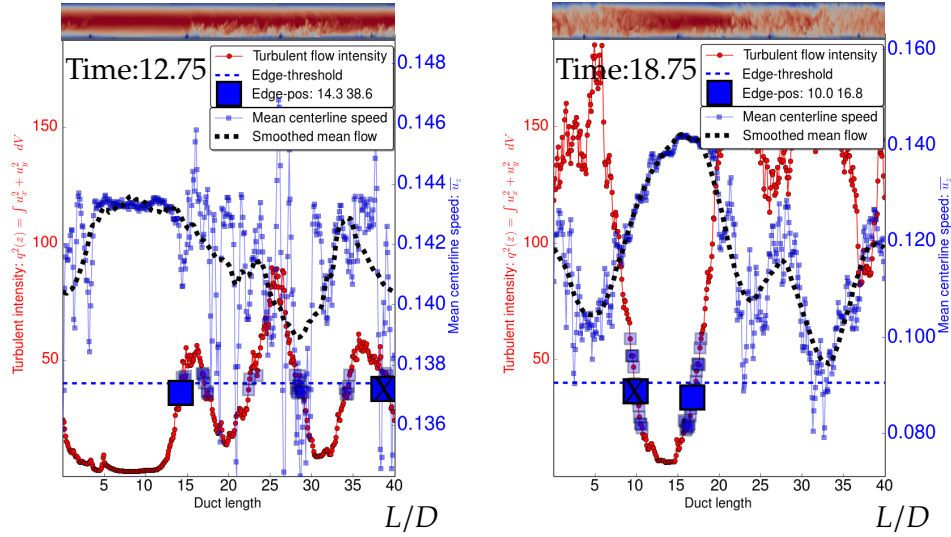


Figure 16.2: An example of strong slug edge tracking. The time indication in top left corner is *advective time*, referring to non-dimensionalized time units as explained in sec. 7.1.1. A threshold (dashed, blue) well above the artificial  $q^2$ -values due to the bumps is chosen. Points on the red curve closest to threshold are marked as transparent blue squares. A clustering algorithm finds mean position for each edge-group. The downstream edge is marked as a solid, blue square with a cross. The upstream edge is marked as a solid, blue square without a cross. The turbulence indicator,  $q^2(z)$ , has been described further in eq. 16.1. We write  $\int dV$  here since we have partitioned the mesh in small slices each with some thickness (see fig. 16.1).

seems simply to be dependent on the bump at  $z = 5$ . In comparison the downstream front in figure 16.3 can be seen to regularly move down the duct at constant speed. Thus, it has not been meaningful to determine the upstream edge speed for  $Re_\tau = 96$ .

It is noteworthy that the edge speeds are all comparable to those known from the literature. It is not possible to make a one-to-one match as (at least to the author's knowledge) there are no duct edge speeds reported for  $3500 < Re$ . One *cannot* just compare with edge speeds for pipes as it is evident in the literature that edge speeds are considerably higher in ducts for a given  $Re$  compared to the speeds in pipes. What one can do however, is to use the prediction made by the model from Barlow et al.

Table 16.1: **Overview of results from Oasis simulations on duct meshes:** Whenever computable the upstream front speed (UF) and the downstream front speed (DF) have been found. In the interest of comparison values from the literature is given along with the relevant Ref. Values obtained with Oasis in the present work is marked with **O**. Notice that some values have been added by eyesight off of figures which are then referenced. These values are thus not fit results by authors but (very) approximate values read from figures. **S** stands for simulative data whereas **E** stands for experimental data.  $F_1$  is figure 3a in [13] and  $F_2$  is figure 3 in [12].

<u>Turbulence</u> <u>class:</u>	UF	Strong Slug DF
Pipe	$0.62u_{avg}$ [12] <b>S</b> $F_2$	$1.55u_{avg}$ [12] <b>S</b> $F_2$
at $Re$ :		5500
Duct	$0.7u_{avg}$ [13] <b>S</b> $F_1$	$1.5u_{avg}$ [13] <b>S</b> $F_1$
at $Re$ :		3500
SIM: $Re_\tau = 96$	-	$1.8u_{avg}$ <b>O</b>
at $Re$ :		5000
SIM: $Re_\tau = 117$	$0.4u_{avg}$ <b>O</b>	$1.9u_{avg}$ <b>O</b>
at $Re$ :		8300

[13]. They only have experimental duct edge speed values up to  $Re = 3500$  but extrapolating the model-slope (from their fig. 3a) to  $Re \approx 5000$  one can see that the upstream speed should be around  $1.7u_{avg}$  which is not far from our result. Also,  $Re \approx 8000$  would yield a guess not far from  $1.9u_{avg}$  which again is in agreement with what we have found here. The pedagogical initiative would now be to show experimental and simulative data as well as the fitted model from Barkley in a figure with our results superimposed. This would allow for an easy comparison. The author *is* in contact with D. Barkley in order to gain access to their results - but as he is currently traveling we have agreed to postpone such a figure until late August.

Obviously, these results can be improved. An advancement such as averaging over 20 runs certainly would help since results easily can be marred by fluctuations. However, these tentative results do indeed call for further research in the area - and the timing is ripe, so to speak, which is evident from the amount of papers currently being published on the topic.

### Slug edge profile

Another way to characterize slugs are by considering their profile. The black line in figure 16.3 shows the profile of a strong slug almost filling the entire mesh. We can in a qualitative way compare this slug outline to the description recently given by Song et al [12]. They describe strong slugs as having two strong fronts where also the downstream front "features a strong peak in  $q(z)$ " [12, p. 11]. This does indeed also seem to be the case for us. Clearly, both edges for the slug show distinct peak structure. It could have been very interesting to study the slug as it expanded further but the mesh length did not allow for this. Song et al. study the slugs from length  $10D \lesssim L \lesssim 150D$  which allows them to see the slug expand into three characteristic regions: an upstream edge, a slug bulk, and a downstream edge. It could have been interesting to rediscover these three characteristic regions of our slugs but it seems the slug in figure 16.3 simply consists of two edges since it is still in a somewhat embryonic state. This would actually be congruent with the slug description in Song et al. where they speak of a  $15D$  distance between slug edge and slug bulk. Thus, two edge widths  $2 \cdot 15D = 30D$  would almost fill our mesh. Seen in that light, it is almost expected that our slugs have no bulk region.

Luckily we do have a second simulation and thereby a second chance to study the slug profile. The strong slug right after initiation of the  $Re_\tau = 117$  simulation can be seen in figure 16.4. Again the plot shows more than one mesh length but a marker above the figure shows the (actual) length of the mesh. Also above the figure one can find a subfigure showing the velocity profile in order to compare with the figure curves.

If we scrutinize the figures from Song et al. closely we see that even though they operate with a  $15D$  width for edges the actual peaks can sometimes even be discerned at a smaller length (see fig. 6 [12]). This is also the case in figure 16.4 where we now easily can rediscover the three characteristic regions for a slug: edge - bulk - edge. Amazingly, we have caught a glimpse of how truly different slugs can be despite pertaining to simulations with somewhat comparable setups. This is a good example of how much the turbulence fluctuates which also leads to large fluctuations in slug characteristics; a picture very much in agreement with what we can find in the literature.

### Weak slugs

The hope was to rediscover patches of laminar spaces within the duct after a long time simulating the transition from the initial profile. As seen in figure 15.4 we are still not in steady state even for the  $Re_\tau = 96$  Oasis

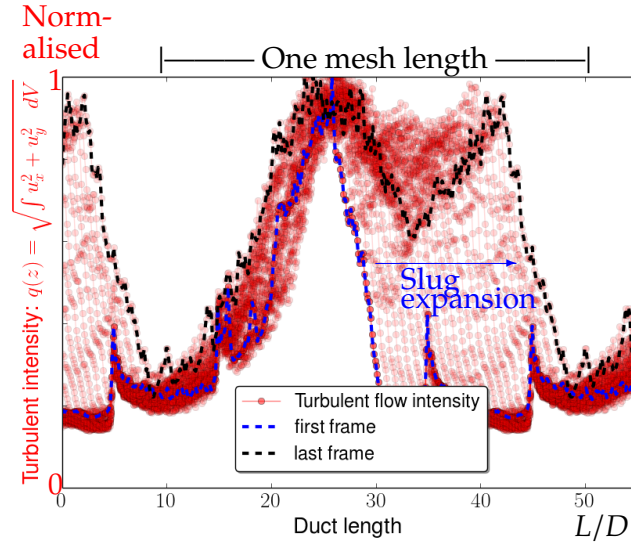


Figure 16.3: A strong slug expanding at  $Re \approx 5000$  right after initiation. Notice that on the  $y$ -axis  $q(z)$  is not squared to better compare it with figures from the literature (e.g. fig. 10 in [12]). This is a slug *shape* study. To better observe the edge profile all iterations are normalised. The  $q(z)$  indicator has been described further in eq. 16.1.

(red line) simulation which should have the lowest final state around  $Re = 2400$ . As was shown in table 7.1,  $Re = 2400$  is right at the transition between weak slug and strong slug which means that even when we have obtained our steady state the patches of relaminarization would be very unlikely. A bulletproof way of obtaining relaminarization would of course be to simply lower the forcing even further. As of now however, no weak slugs have been discovered.

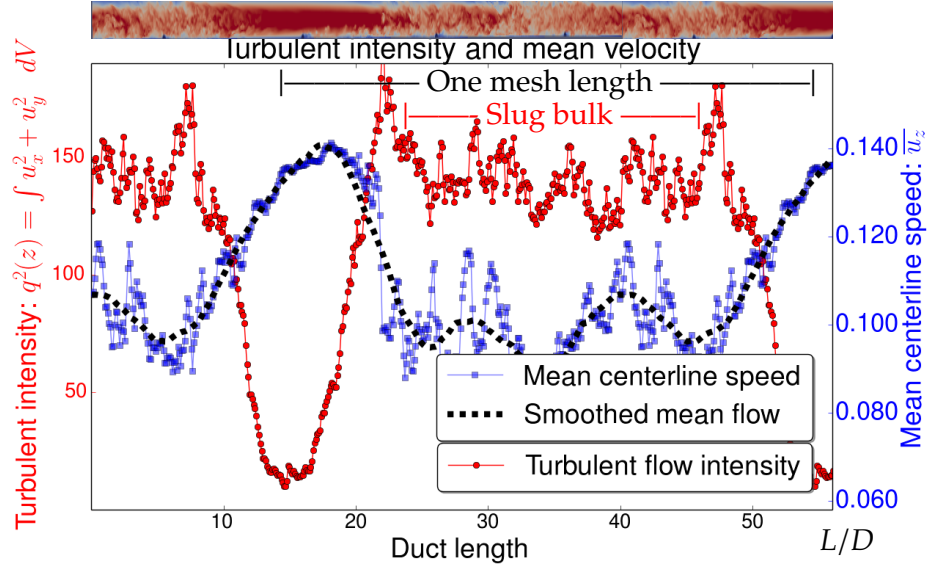


Figure 16.4: Edge study from the  $Re_\tau = 117$  simulation (Notice however, that at initiation  $Re \approx 8300$ ). Indicators of turbulent (red) and laminar (blue) flow (laminar profiles have a higher centerline velocity than turbulence) are shown. Above is given the velocity field for the duct at the same time-step. In the terminology of FD we observe a strong slug almost filling the entire duct. Importantly, the characteristics for a strong slug are easily recognized: Two strong fronts separated by a slug-bulk. The  $q^2(z)$  indicator has been described further in eq. 16.1. We write  $\int dV$  here since we have partitioned the mesh in small slices each with some thickness (see fig. 16.1).

## DISCUSSION & OUTLOOK



ow that the main results have been presented we give a critical assessment hereof. Afterwards, we outline the ensuing research at NBI and a conclusion for the present work.

### 17.1 Discussion

#### 17.1.1 Turbulence simulation results

The first question that comes to mind when assessing the mean profiles in section 14 is 'can we expect a good agreement with the law of the wall and the logarithmic law?' We have here pointed to the work by Kim et al. [33] who obtained a very good agreement with both laws for  $Re = 3300$  (channel flow). Following their lead we have used the same constants  $\kappa = 0.4$  and  $B = 5.5$  for the log law. Therefore, the discrepancy is in principle unwarranted and we should be able to pinpoint the cause.

Possible reasons for discrepancy are:

1. The measurements were not conducted in a statistically steady state and few ( $\approx 1000$ ) iteration were used.
2. Despite using the top-most half of the probing line ( $0.5 < y < 1$ ) effects from the bumps cannot be ruled out.
3. The mesh is designed for  $Re \lesssim 2600$  and Oasis developers point in particular to the mesh resolution as an important factor.



4. The paper presenting Oasis also reports an overestimation for the logarithmic layer (even for their finest mesh)[16, fig. 10].

Certainly, all points above are valid but to sum up one can perhaps combine 3. and 4. in the following considerations:

Kim et al. [33] used a resolution of 129 grid points in the spanwise direction as did Mortensen and Valen-Sendstad when testing Oasis [16]. Likewise, both simulated  $Re_\tau = 180$  ( $Re = 3300$ , channel flow). We have on the other hand only 80 grid points in the spanwise direction which is below their resolution. On the other hand the flow was characterized as having  $Re = 5500$  at the time of our investigation (for the zoom plot in fig. 14.5) which is a considerable increase in Reynolds number. Therefore, since Mortensen and Valen-Sendstad observed an overestimation for their setup it would be logical to expect an even more significant discrepancy in the present case. Comparing figure 14.5 with figure 10 in their paper [16] we do indeed seem to have a bit higher overestimation. The final question then becomes: ‘Why does Oasis overestimate the velocity profile in the logarithmic layer and what can be done to mitigate this?’ The two solutions proposed by the Oasis developers are employing Large Eddy Simulation (LES) techniques or refining the mesh. LES should be viewed in relation to DNS. In the latter one hopes to resolve all scales of turbulence whereas in the former one *does not* resolve all scales and thereby gain an increase in computational performance. To counter the resulting lack of dissipation in LES one simply adds viscosity and thereby hopes to gain results comparable to those of DNS but obtained much faster. Oasis does have utilities in the source code for LES but these have not been tested (and are to the author’s knowledge not described anywhere). Likewise, the mesh-refinement has not been tested in the present work and would clearly be an improvement. In conclusion, the discrepancy in the logarithmic law could very well be explained by the above.

### 17.1.2 LBM comparison

We will now consider how to properly compare two computational methods. Both methods (Oasis and the LBM) have been verified in the laminar regime (see fig. A.20 for Oasis and [48] for LBM) not to mention the thorough verification done by the Oasis developers.

In this work we have only focused on two things: (1) computational performance and (2) profile similarity.

With respect to computational performance it is clear from figure 15.1 that the LBM is a factor  $\approx 2$  faster than Oasis. The comparison was carried

out on the same server and on the same mesh. Thus, it should hold little bias.

Assessing the profile similarity is on the other hand less straightforward. Clearly, there is a huge discrepancy between Oasis and LBM profiles in figure 15.2. This is true both for the  $Re_\tau = 96$  (red) and the  $Re_\tau = 117$  (blue) simulation even though the Oasis profiles can be seen to gradually converge towards the LBM profiles. The reason for the discrepancy is obvious in figure 14.2: Neither Oasis nor LBM simulations were in steady state and since they approach the state from different sides the profile difference was inevitable. The last (futile) attempt in figure 15.5 to obtain a complete profile match between the two green curves in figure 15.4 failed for the same reason the first comparison did: A lack of steady state in the simulations.

The closest one-to-one match between profiles was observed in figure 14.2 between the Oasis-simulation  $Re_\tau = 96$  (red, dashed) and the LBM-simulation  $Re_\tau = 127$  (black, dashed). The explanation can be found in figure 15.4 which clearly shows that the simulations have comparable kinetic energy (see the red stars). Thus reassuringly, we find that the methods produce similar profiles for similar kinetic energy levels. Should one point to a minor discrepancy in figure 14.2 then the Oasis-simulation  $Re_\tau = 96$  (red, dashed) seems a bit more turbulent than LBM-simulation  $Re_\tau = 127$  (black, dashed). This could be expected from a simulation transitioning down from higher  $Re$  which could have more turbulent modes than a simulation transitioning from laminar to turbulence. In general, we have reason to conclude that the profile analysis is reasonable although it certainly could be improved.

Which computational method is then best? Since the spatial and temporal order of convergence are alike one should prefer the LBM if the maximum meaningful time-step is  $\Delta t = 0.001$ , since it is twice as fast as Oasis. However, should the time-step limit be higher as in our case it is a severe limitation for LBM not to be able to increase its  $\Delta t$  and one must conclude that Oasis quickly becomes superior. To get a feeling for this advantage one can consider the two green curves in figure 15.4: The LBM curve has been obtained by simulating almost continuously since April the 11th (almost 4 months) whereas the green Oasis curve has been obtained in four days. Seen in that light it is hard not to prefer Oasis.

On a side note: in FEM (and thereby Oasis) it is remarkably easy to change the *spatial* order of convergence. One simply chooses a higher order element. Not all methods provide such a lenient adjustment. For turbulence simulations this is less attractive since the computation

time increases. Thus, we merely mention this advantage for the sake of completeness.

### 17.1.3 Front dynamics

In chapter 16 we determined slug-edge speeds and analyzed the profile characteristics. These results are by all means tentative. We have no real measure of uncertainty. We could try to estimate the error on our edge locations which would in turn result in an error in the estimated edge speeds. It is not that the author cannot propagate the error - this error metric is simply beside the point altogether. The interesting phenomenon to capture with an error/uncertainty measure is the fluctuations intrinsic to turbulence. Thus another much more relevant error metric could be the standard deviation from consecutive simulation under same conditions. As mentioned, Song et al. have no less than 20 simulations per  $Re$ . This allows them to describe the simulations as an ensemble of runs resulting in a mean and a standard deviation. Clearly, the metric captures the high degree of fluctuations intrinsic to the problem. Since we have two slug examples at considerably disparate  $Re$  we cannot hope to report such a metric. Thus, 10-20 repetitions for each initiation would be a considerable upgrade to our results.

Furthermore, Song et al. can document a change in the downstream speed as the length changes. We have not even attempted to measure this since they can scan slug lengths from  $\approx 10D \rightarrow 150D$ . Our mesh restricts us from carrying out these investigations which is another limitation in the present work.

Finally, in particular the downstream front for the  $Re_\tau = 96$  simulation is a bit high and should according to the model by Barkley et al. have been in the range  $1.6u_{avg} < u < 1.7u_{avg}$ . One could point to many reasons for the discrepancy. To name a recent example; the results from Song et al. [12] obtain a different characteristic altogether of the upstream fronts evolution in time compared to Nishi et al. [8]. The former hardly see any transient behavior whereas the latter observe a clear transient behavior. To explain this discrepancy Song et al. point to differences in initialization and in the choice of disturbance applied to the system. This reflects that, although undesired, it is not in the least uncommon to see (minor) discrepancies when comparing results across platforms. Seen in that light the slightly too high downstream front speed for the  $Re_\tau = 96$  simulation seems perhaps less grave.

As stated in Barkley et al. the crucial step in the attempt to connect

the different flow regimes is to characterize the *origin* of turbulence (i.e. the onset of turbulence) [13, p. 552]. On the basis of the presented investigations we can only conclude that further research characterizing this onset (more quantitatively) should be possible with our current implementation of Oasis. As mentioned, the author is in correspondance with D. Barkley in order to obtain access to the experimental duct data so a direct fit can be made. As the latter is currently traveling this result is estimated to be obtained in late August 2016. Thus, the present results (tab. 16.1) can (eventually, hopefully) be viewed as further solidifying the model by Barkley et al. A general word of caution should however be issued here at the end: The two slugs we identified were for  $Re \approx 5000$  and  $Re \approx 8000$  which hardly can be said to be within the transitional regime. Thus, it is difficult to tell whether our results should even have a perfect match with the Barkley-model.

## 17.2 Ensuing research

One of the purposes of this thesis was to lay the groundwork for truly large scale turbulence simulations to be carried out in the fall 2016 at NBI. Despite already having a well-functioning fluid solver (LBM) investigations of other solvers' ability to assist in the upcoming project were desired as well as general tentative investigations within the research field 'Onset of Turbulence' to clarify what research projects could be deemed feasible. This should contribute to the all-important *design* of the project(s) without which one cannot hope to obtain publishable results; the computation time is simply too vast.

It is the author's clear opinion that such an assisting solver has been identified in Oasis. It has an intuitive interface, uses state-of-the-art algebraic backends and most importantly *complements* the LBM e.g. with its leniency in time-stepping.

Currently, two projects seem imminent judging by the current state of the research field and could be done this fall: (1) a study of the inherent *hysteresis* within the transitional regime and (2) an investigation of and an attempt to model the all-important *Reynold stresses* which would help explain the production of localized turbulence.

The proposed project of hysteresis investigation is by far the computationally most demanding one. By initiating a few dozen runs through the transitional turbulent regime - both from above and from below - one would be able to get a thorough description of the *Re*-thresholds for the various regimes among many other interesting objectives. It is very clear

in the literature that hysteresis *is* present in the transition [27, 28] and that results may differ depending on the choice of initialization (whether we sweep from above or below). A specific example is seen in tab. 7.1 (\* values) which represent both approaches. The role of the LBM in this project is debatable due to the time-step limitation. To conduct the number of sweepings needed it is very important to choose the highest possible  $\Delta t$ . However, it could easily be carried out using Oasis exclusively.

The second proposed project revolves around the Reynold stresses. We have thus come full circle and are back to the ever-occurring *closure problem* which was mentioned several times throughout the introduction (sec. 5.3.1, 6.2.1 and 7.2.2). Taking an approach from Statistical Physics the design for this project would be to choose a select number of  $Re$  in the transitional regime and run steady-state simulations. In turn, a clear(er) description (probability density function) of e.g. the term  $-\int_V u'_i u'_j \frac{\partial \bar{u}_i}{\partial x_j} dV$  from eq. 7.1 could be obtained. This improved insight is then to be coupled with a *model*. In the most recent (March 30th, 2016) review by P. Manneville [28] he dedicates a whole section to "Understanding *via* modeling" (sec. 4.3). He then goes on to conclusively state on the Reynolds stresses: "Playing a role in turbulent spot growth, they demand further scrutiny". This project certainly fits the bill. Here, one could find good use for both the LBM and Oasis. One would probably have to utilize Oasis to compute the steady-state profiles for all  $Re$  and then simply interpolate initial fields to the LBM in order to save time. However, once at steady state it would only further solidify the project to have complementing methods providing a way to check self-consistency.

In summary, further (exciting) research is pending at NBI. A rough estimate would be to have both projects simulated by late November 2016. It all hinges on decisions for mesh resolution, number of repetitions etc. and thus a final design should be agreed upon.

## CONCLUSION

In this study we conducted Direct Numerical Simulations (DNS) on a deformed duct mesh using the fluid solver 'Oasis'. In order to fully master this state-of-the-art solver, a comprehensive study of (1) the research field ('Onset of Turbulence'), (2) the Finite Element Method (FEM) and finally (3) Computational Fluid Dynamics (CFD) were completed and have been presented.

The analyses of the data from the final DNSs represent the main result. First of all, the data was verified using the law of the wall as well as the logarithmic law and clear physical traits were observed. Furthermore, a solver comparison was carried out between Oasis and the in-house solver at Niels Bohr Institute (NBI) and areas where they complement each other were identified. Our study has thereby laid the groundwork for large scale turbulence simulations which will be carried out at NBI during the fall of 2016. Finally, investigations into the onset of turbulence were done in order to characterize the localized turbulence right on the cusp of fully developed turbulence. It was straightforward to fit the results into the broader frame set by the relevant literature which merits the pending research projects. During the final investigations several considerations pointing out the strengths and weaknesses of the current setup were made. This in turn is all-important to the design of DNS-projects due to the vast computation time. As a culmination of the thesis two new projects have been proposed both addressing central, unanswered phenomena to the research field, 'Onset of Turbulence'.



## APPENDIX

### A.1 Physics in ducts and pipes



THIS section complements the theory in part II and introduces among other things the notion of *friction*, *pressure loss* and *pumping power*:

In many of these calculations the average velocity,  $u_{avg}$ , is of interest which analytically can be found by the *conservation of mass* principle:

$$\dot{m} = \rho u_{avg} A_c = \int_{A_c} \rho u(\mathbf{x}) dA_c \quad (\text{A.1a})$$

$$u_{avg} = \frac{1}{A_c} \int_{A_c} u(\mathbf{x}) dA_c \quad (\text{A.1b})$$

Here,  $\dot{m}$  is mass flow rate and  $A_c$  is the cross-sectional area [51, eq. 8.1]. In FEM simulations it is important to remember that the grid is unstructured when calculating  $u_{avg}$  from eq. A.1b. This means the elements vary in size and should give a corresponding weight to the found  $u$ -profile. One can therefore not just take the mean but must instead integrate (`assemble()`) over the combined area. Should one be interested in a particular cross section, say the inlet of a pipe the (FEniCS)code-pendant to eq. A.1b is:  
`u_avg = assemble(u*ds(mark["inlet"]))/(rad**2*pi)`  
Of course the most immediate use of  $u_{avg}$  that spring to mind is the Reynolds number,  $Re = u_{avg} L_0 / \nu$ . The characteristic length which as

mentioned is the diameter for pipes is in general given by the *hydraulic diameter*:

$$D_h = \frac{4 A_c}{p} \quad (\text{A.2})$$

where  $p$  is the wetted perimeter. For a square duct  $D_h$  simplifies to  $D_h = 4 \cdot (a \cdot a)/(4 \cdot a) = a$ , for side width  $a$  which is 1 for our meshes.

### Pressure loss

In eq. 5.3  $\Delta P = P(0) - P(L)$  so  $u(r)$  is positive when pressure decreases. This drop in pressure is as mentioned called the pressure loss,  $\Delta P_L$ , and can be expressed in terms of average velocity thanks to eq. 5.3:  $\Delta P_L = 8\mu L(u_{avg}/R^2)$  [20, eq. 16.32]. This relation tells us that more viscous fluids experience a larger drop in pressure. The general form for  $\Delta P_L$  for both laminar and turbulent flow is [51, eq. 8.21-22]:

$$\Delta P_L = f \frac{L}{D} \frac{\rho u_{avg}^2}{2} \quad (\text{A.3a})$$

$$f = (8\tau_w)/(\rho u_{avg}^2) \quad (\text{A.3b})$$

where both the *Darcy friction factor* in eq. A.3b and thus the wall shear stress,  $\tau_w = -\mu(du/dr)|_R$ , is seen.

The pressure loss is interesting since it gives the required pumping power,  $\dot{W}_{pump}$ , to maintain the flow. Once the volume flow rate is found,  $\dot{V} = u_{avg} \cdot A_c$ , its product with  $\Delta P_L$  gives:  $\dot{W}_{pump} = \dot{V} \cdot \Delta P_L$ . The  $\dot{V}$  is also known as *volumetric discharge rate* and  $\dot{W}_{pump}$  also goes under the name "Poiseuille dissipation",  $\mathcal{P}$ . The *Drag force*,  $\mathcal{D}$ , which of course is closely related to the pressure loss, can also easily be computed and is;  $\mathcal{D} = \Delta P_L \cdot A_c = 8\mu L(u_{avg}/R^2) \cdot (\pi R^2) = 8\pi\mu u_{avg} L$  [20, eq. 16.7,31,37-38].

### Friction factor

Knowing  $u$  one can calculate  $\tau_w$  and in turn the Darcy friction factor,  $f$ . Often times however, once  $u$  is found one does not need to calculate  $f$  corresponding to a given cross-sectional shape since any text book will have a table with the most common values. For regular pipe and duct the (laminar) values are:  $f_{pipe} = 64/Re$  and  $f_{duct} = 56.92/Re$  [51, tab. 8.1]. Please note, that the *friction coefficient*,  $C_f = f/4$ , is related to the Darcy friction factor but differ and thus should not be interchanged. Using eq. A.3b we easily get the corresponding definition for  $C_f$ :

$$C_f = \frac{\tau_w}{(1/2)\rho u_{avg}^2} \quad (\text{A.4})$$



It is called the *Fanning friction factor* and is defined as the average shear wall stress over the *dynamical pressure*,  $(1/2)\rho u_{avg}^2$  [20, eq. 16.47]. How to compute the average wall shear stress? Well, simply computing  $\tau_w = -\mu(du/dr)|_R$  using eq. 5.3 or dividing drag force with total surface area will both give:  $\tau_w = \frac{\mathcal{D}}{2\pi RL} = 4\mu \frac{u_{avg}}{R}$ . Inserting the result in eq. A.4 gives:

$$C_f = \frac{16}{Re} \quad (A.5)$$

### Turbulent friction factors

The final functional form of the fanning friction factor in turbulent regime changes compared to its laminar counterpart in eq. A.5. We write  $C_f(Re)$  to explicitly stress that it is a function of  $Re$  [20, eq. 16.47]:

$$C_f(Re) = \frac{\tau_w}{(1/2)\rho u_{avg}^2} = \frac{16}{Re} \cdot \mathcal{F}(Re) \quad (A.6)$$

Above,  $\mathcal{F}$  is the dimensionless *drag factor* that in the laminar regime is simply  $\mathcal{F}(Re) = 1$ . In the limit  $Re \rightarrow \infty$  the  $C_f(\infty)$  saturates to a constant:

$$C_f(\infty) \approx 0.028 \left( \frac{\eta}{2R} \right)^{(1/4)} \quad (A.7)$$

Interestingly, the drag factor,  $\mathcal{F}(Re)$ , is now a mere constant found with roughness scale,  $\eta$ , and radius. The dependence on viscosity is gone - despite viscosity being the very reason for friction. This is however, first in the limit  $Re \rightarrow \infty$  (above 100.000) [20, p. 276]. Also the Darcy friction factor,  $f$ , final form in fully turbulent flow changes a lot. It cannot be obtained theoretically but based on experiments, the implicit Colebrook equation A.8 can be found [51, eq. 8.50]. As seen, it depends on  $Re$  and the relative roughness,  $(\eta/D)$ .

$$\frac{1}{\sqrt{f}} = -2 \cdot \log_{10} \left( \frac{(\eta/D)}{3.7} + \frac{2.51}{Re \sqrt{f}} \right) \quad (A.8)$$

Usually, one consults the *Moody chart* instead of the implicit equation above. It gives the Darcy friction factor,  $f$ , as a function of  $Re$  and  $(\eta/D)$ . Should the conduit be a duct instead of a pipe one can simply insert the hydraulic diameter mentioned above. Once  $f$  is found using e.g. the Moody chart we can use eq. A.3a, which is perfectly valid also for turbulent flows, to obtain  $\Delta P_L$  and we have come full circle to where we began for laminar flow (assuming we can measure the flow rate).

## A.2 Transient Hagen-Poiseuille flow

**I**n part IV we build a transient NS-solver using the splitting method proposed by A. Chorin. ascertain ourselves of a correct implementation we subsequently test it in a cylindrical setup by comparing the simulation result to an analytical solution.

To obtain an analytical transient solution to the cylindrical pipe for an incompressible Newtonian fluid we assume a purely axial velocity profile,  $\mathbf{u} = u \cdot \hat{\mathbf{z}}$ , where the  $\hat{\mathbf{z}}$  is parallel to the cylindrical axis. So, since  $u_r = u_\theta = 0$  we only look at  $u_z$ . Neglecting body forces and the convective term in NS-equations we get the resulting  $z$ -momentum equation in eq. A.9 where we in the step between the equations used the scaling  $p \rightarrow \rho \cdot p$  and the  $\nu = \mu/\rho$  for simplicity:

$$\begin{aligned}\rho \partial_t v_z &= -\partial_z p + \mu \nabla^2 v_z \\ \partial_t v_z &= -\partial_z p + \nu \nabla^2 v_z\end{aligned}\tag{A.9}$$

The corresponding BCs and ICs to the governing eq. A.9 are seen in eq. A.10a and A.10b.

$$v_z(R, t) = 0 \quad \forall t > 0 \tag{A.10a}$$

$$v_z(r, t) = 0 \quad \forall t \leq 0 \tag{A.10b}$$

The above stated equations can be solved with a Hankel transform. We refer interested readers to Refs [31, 30]. Once transformed to  $\lambda_n$ -space and back to real space again we finally get the radial velocity profile  $u(r, t)$  in eq. A.11 [48, eq. 45].

$$u(r, t) = (1 - r^2) - 8 \sum_{n=1}^{\infty} \lambda_n^{-3} \frac{J_0(\lambda_n r)}{J_1(\lambda_n)} e^{-\lambda_n^2 \nu t} \quad : 0 \leq u(r, t) \leq 1 \tag{A.11}$$

where  $J_n$  is the  $n$ 'th order Bessel function of first kind,  $\lambda_n$  the  $n$ 'th root of  $J_0$ . Finally we note, that the equation is scaled so that  $u_{max} = 1$  and  $r_{max} = R = 1$  compared to the steady state expression in e.g. eq. 5.3. Thus, we should expect a radial velocity profile which should converge towards the steady state parabolic Hagen-Poiseuille flow for a pipe since the exponential term,  $e^{-\lambda_n^2 \nu t} \rightarrow 0$  for  $t \rightarrow \infty$ . It is the expression in eq. A.11 we use to conduct the comparison.

### A.3 FEM examples using self-written code

**T**HROUGHOUT part III we have solved numerous FEM examples with increasing difficulty to check the functionality of our solver. To save space, the more thorough description is given here whereas only final solutions are visualized for the sake of intuition in the main text.

#### A.3.1 FEM 2D static example

To give a static 2D example we now solve the Poisson equation. Having already found the weak formulation we only need to specify our BC as in eq. A.12a and A.12b where  $L_x$  and  $L_y$  defines a rectangular mesh.

$$u(x, y) = 0 \quad : (x, y) \in (x, 0) \cup (x, L_y) \cup (0, y) \quad (\text{A.12a})$$

$$\frac{\partial u(x, y)}{\partial n} = q_0 \quad : (x, y) \in (L_x, y) \quad (q_0 \text{ is positive}) \quad (\text{A.12b})$$

To assemble the system equation we need to specify what goes into the element vector,  $\mathbf{f}_e$ . Choosing  $g(\mathbf{x}) = 1$  for simplicity the resulting integrals from eq. 8.32 and 8.33 can be seen in eq. A.13 and A.14 where  $h_{ij}$  is the distance between the two nodes in question. Please note, that the boundary integral only contributes to "Neumann boundaries" if we require that the test function,  $w(\mathbf{x})$ , vanish on boundaries where  $u(\mathbf{x})$  is known (Dirichlet boundaries) [19, p. 6].

$$\mathbf{f}_e = \frac{A}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (\text{A.13})$$

$$q_0 \int_{y_i}^{y_j} \left[ \frac{y_j - y}{y_j - y_i} \right] dy = q_0 \cdot \frac{h_{ij}}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (\text{A.14})$$

Notice that the natural boundaries are parallel to the y-axis in this case which makes the integral very simple. It is simply a line (1D) integral where we have used the definition of  $u(\mathbf{x})$  from eq. 8.17. The result is however valid also for Neumann boundaries along an arbitrary direction in the 2D-plane as long as  $h_{ij}$  defines the actual distance between the two affected nodes [54, p. 93]. The line integral obviously only contributes with a vector of length two to  $\mathbf{f}_e$  since an element only has two nodal points on the boundary.

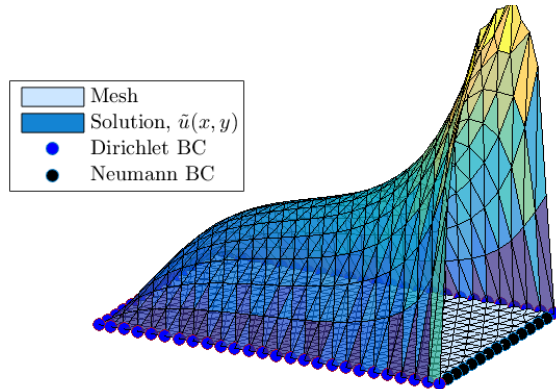


Figure A.1: The Poisson equation solved with  $g(\mathbf{x}) = 1$ . The BCs are seen in eq. A.12a and A.12b.

### Imposing Boundary Conditions:

There are many ways of dealing with BCs. One approach that generally works is to first assemble the system equation 8.30 due to the two volume integrals in eq. 8.24. Afterwards, the boundary integral (Neumann BCs) can be added to  $\mathbf{f}$  for nodes on the boundary. Once the entire system equation is assembled one simply impose the Dirichlet conditions by replacing the existing element in  $\mathbf{f}$  with the constrain value for all constrained nodes. The corresponding row in  $\underline{\mathbf{K}}$  is also erased and a 1 is inserted at the position corresponding to the given node. The solution for the problem is visualized in figure A.1.

### A.3.2 FEM 2D transient examples

In the following two examples showcasing respectively the Backward Difference scheme and Crank-Nicolson scheme are given after which the two Stokes' problems will be presented.

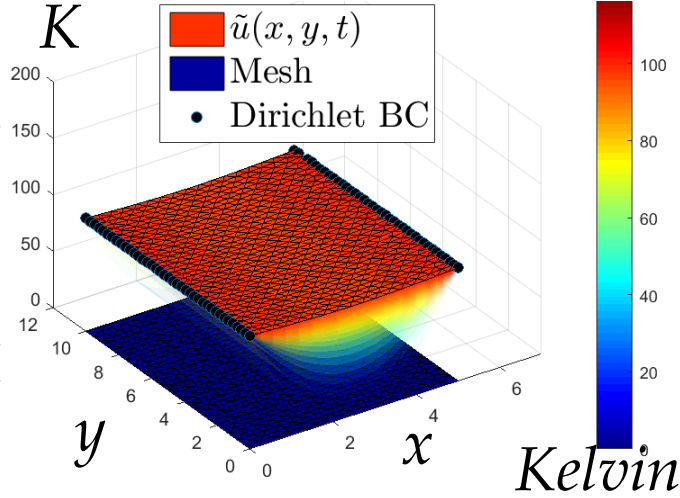
#### Heat equation using backward difference

A solution employing backward difference to the heat equation is seen in figure A.2. Here, the initial condition was  $u(\mathbf{x}, 0) = 0$  and the BCs were two opposing sides of the rectangular mesh with constant temperature and two isolated sides,  $\frac{\partial u}{\partial n} = 0$ . As expected the temperature rises to the final, constant temperature equal to the boundaries.

#### Heat convection using Crank-Nicolson

As a summery the introduction to *transient* problems we solve a classical heat convection problem. Thus, the problem equation is the heat equation

Figure A.2: The heat equation solved for the initial condition  $u(x, 0) = 0$ . As for the BCs: Two opposing sides of the rectangular mesh with constant temperature and two isolated sides with,  $\frac{\partial u}{\partial n} = 0$ . The red plane at temperature 100 is the final solution. The previous planes are seen below in transparent colors.



from eq. 8.35a with the corresponding system equation restated in eq. A.15.

$$\underline{\underline{\mathbf{M}}} \dot{\mathbf{u}} + \underline{\underline{\mathbf{K}}} \mathbf{u} = \mathbf{f} \quad (\text{A.15})$$

The mesh is still a simple square but now the BCs are slightly altered compared to the previous example. The Dirichlet conditions are stated in eq. A.16 and Neumann conditions in eq. A.17a and the A.17b. For this problem we have initial conditions  $u(x, y, 0) = 0$ .

$$u(x, y, t) = u_0 \quad : (x = 0, y) \wedge (x = L_x, y) \quad (\text{A.16})$$

$$\frac{\partial u(x, y, t)}{\partial n} = 0 \quad : (x, y = 0) \quad (\text{A.17a})$$

$$\frac{\partial u(x, y, t)}{\partial n} = a(u - u_0) \quad : (x, y = L_y) \quad (\text{A.17b})$$

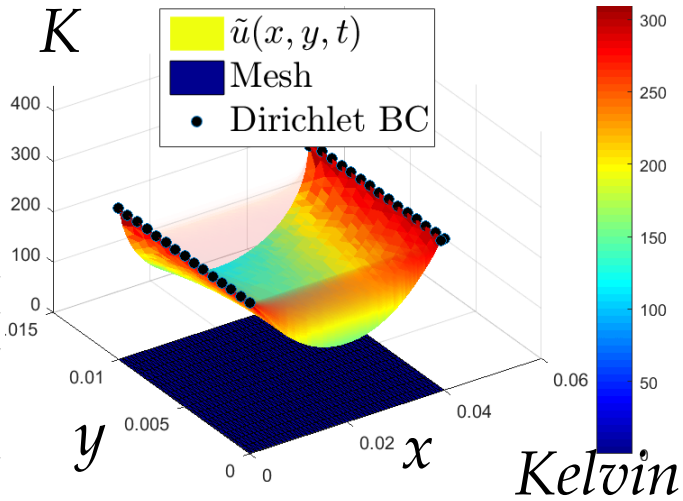
The first Neumann condition in eq. A.17a is also known as an isolated boundary whereas the second Neumann condition is heat transfer, i.e. heat convection where the heat flux is proportional to the ambient temperature,  $u_0$ , and the constant  $a$  is the heat convection coefficient. We have previously shown how to implement Neumann conditions in the static 2D Poisson example and the procedure here is the same, only now the BCs are slightly

more complicated. To obtain  $\mathbf{f}$  we must again solve eq. 8.33 but we state the RHS of eq. A.17b as  $(a \cdot u - b)$  before inserting it. The result for each element is seen in eq. A.18.

$$\begin{aligned}
 \int_{\Gamma_n} w(\mathbf{x}) \frac{\partial u(\mathbf{x})}{\partial n} d\Gamma &= \int_{\Gamma_n} w(\mathbf{x}) \cdot (a \cdot u - b) d\Gamma \\
 &= a \cdot \int_{x_i}^{x_j} \begin{bmatrix} \frac{x_j-x}{x_j-x_i} \\ \frac{x-x_i}{x_j-x_i} \end{bmatrix} \begin{bmatrix} \frac{x_j-x}{x_j-x_i} & \frac{x-x_i}{x_j-x_i} \end{bmatrix} dx \cdot \begin{bmatrix} u_i \\ u_j \end{bmatrix} - b \cdot \int_{x_i}^{x_j} \begin{bmatrix} \frac{x_j-x}{x_j-x_i} \\ \frac{x-x_i}{x_j-x_i} \end{bmatrix} dx \\
 &= a \cdot \frac{h_{ij}}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} u_i \\ u_j \end{bmatrix} - b \cdot \frac{h_{ij}}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}
 \end{aligned} \tag{A.18}$$

It is easily seen that the last term in eq. A.18 is equal to eq. A.14. In this case we integrate along the  $x$ -direction but the result is the same. This term must be added to the vector in our system equation. The first term is seen to be a  $2 \times 2$  matrix since the element has two nodes on a line boundary. Calculating the term is straightforward using the same convention as in the 1D-example by expressing  $u(\mathbf{x})$  as in eq. 8.17. The resulting matrix *must be added to the matrix* in our system equation. Finally, we are ready to assemble the system equation in eq. A.15. Using the Crank-Nicolson scheme in eq. 8.52 we obtain the solution seen in figure A.3 for  $u_0 = 300$  Kelvin.

Figure A.3: The heat convection problem solved for the BCs in eq. A.16 to A.17b. As seen, the final solution has considerably smaller temperatures at the non-isolated boundary where heat diffuses to the surrounding environment. The previous planes are seen above in transparent colors.



### A.3.3 Stokes' problems and analytical comparison

To conclude the introduction to FEM we will solve Stokes' first and second problem. Both problems are treated in depth in the *Continuum Mechanics* course at Nbi and a more thorough description can be found in the corresponding notes [31]. For both problems the starting point is a simplified version of the incompressible NS-equations seen in eq. A.19a and A.19b. This type of flow is called *Stokes flow*. Here, the body forces have been left out as well as the convection term since we assume low Reynolds number for the fluid,  $Re \ll 1$ .

$$\frac{d\mathbf{u}}{dt} + \frac{1}{\rho_0} \nabla p = \nu \nabla^2 \mathbf{u} \quad (\text{A.19a})$$

$$\nabla \cdot \mathbf{u} = 0 \quad (\text{A.19b})$$

To further simplify matters we only look at shear forces, i.e. we assume the pressure gradient is zero, and choose a very manageable setup seen in figure A.4.

As seen in the figure the  $x$ - $z$  plane is moving parallel to the  $x$ -direction yielding a parallel shear stress in the fluid. Assuming the motion of the  $x$ - $z$  plane can be described by the oscillatory expression:  $U_0 \cdot \cos(\omega t)$  and that the fluid is at rest in the limit  $y \rightarrow \infty$  we can sum up our BCs and initial conditions as in eq. A.20a to A.20c.

$$v_x(0, t) = U_0 \cos(\omega t) \quad (\text{A.20a})$$

$$v_x(y, t) \rightarrow 0 \quad : y \rightarrow \infty \quad (\text{A.20b})$$

$$v_x(y, 0) = 0 \quad (\text{A.20c})$$

With all simplifications taken into account the NS-equations simplify even further resulting in a diffusion equation as seen in eq. A.21.

$$\frac{\partial v_x(y, t)}{\partial t} = \nu \frac{\partial^2 v_x(y, t)}{\partial y^2} \quad (\text{A.21})$$

#### Analytical solution

One can apply the Laplace transform with respect to time to get the general solution seen in eq. A.22 where  $A(s)$  and  $B(s)$  are functions of  $s$ . By use of BCs we discard  $B(s)$  and find  $A(s)$  to obtain the complete (Laplace

transformed) solution to Stokes' problems seen in eq. A.23.

$$\hat{v}_x(y, s) = A(s) e^{-\sqrt{s/\nu} y} + B(s) e^{\sqrt{s/\nu} y} \quad (\text{A.22})$$

$$\hat{v}_x(y, s) = \frac{U_0 s}{s^2 + \omega^2} \cdot e^{-\sqrt{s/\nu} y} \quad (\text{A.23})$$

Now we have found the solution. We only have to use the inverse Laplace transform since we would like to express the solution in real time,  $t$ . To this end, we express our solution as a product of two terms;  $\hat{v}_x(y, s) = \hat{g}_1(s) \hat{g}_2(y, s)$  and make use of the *Convolution Theorem* to obtain eq. A.24. Here naturally,  $g_1 = \mathcal{L}^{-1}[\hat{g}_1]$ .

$$\mathcal{L}^{-1}[\hat{g}_1(s) \hat{g}_2(y, s)] = \int_0^t g_1(t - \tau) g_2(\tau) d\tau \quad (\text{A.24})$$

Finding  $g_1(s)$  is straightforward but for  $g_2(y, s)$  care has to be taken when choosing the integration path after which the solution can readily be found [31, p. 6]. The inverse Laplace transforms of  $g_1$ ,  $g_2$  and of the final solution  $v_x(y, t) = \mathcal{L}^{-1}[\hat{v}_x(y, s)]$  are listed in equation A.25 to A.27. It is this final analytical solution to the fluid equation that we will compare our FEM results with for both problems.

$$g_1(t) = U_0 \cos(\omega t) \quad (\text{A.25})$$

$$g_2(y, t) = \frac{y}{\sqrt{4\pi\nu}} t^{-3/2} e^{-y^2/(4\nu t)} \quad (\text{A.26})$$

$$v_x(y, t) = \frac{U_0 y}{\sqrt{4\pi\nu}} \int_0^t \cos[\omega(t - \tau)] \tau^{-3/2} e^{-y^2/(4\nu\tau)} d\tau \quad (\text{A.27})$$

### Stokes' First Problem

The first case to solve is for steady movement of the (red) boundary plate seen in figure A.4. Thus, we have  $\omega = 0$  yielding the analytical solution in eq. A.28. Here, *erfc* is the *complementary error function*. For simplicity we put  $\nu = 1$ .

$$v_x(y, t) = \text{erfc}\left[\frac{y}{2\sqrt{\nu t}}\right] \quad (\text{A.28})$$

To make a comparison to the analytical expression we must obtain a FEM solution to the diffusion problem in eq. A.21 with BC and IC in eq. A.20a to A.20c. Having already solved a diffusion equation with FEM



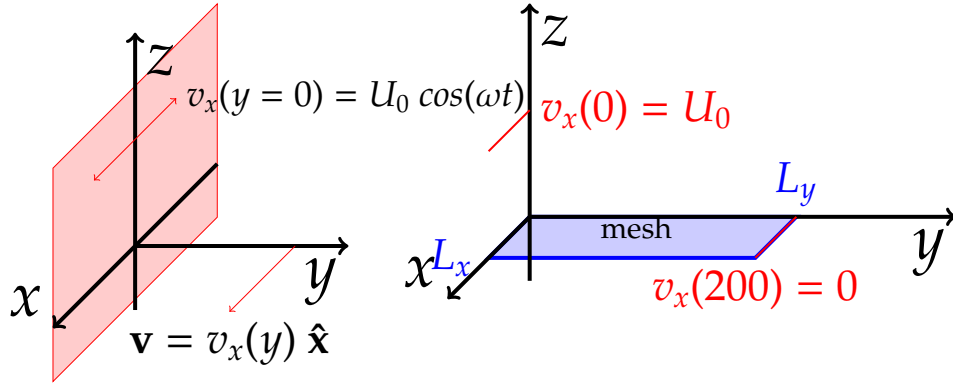


Figure A.4: Setup for Stokes' first and second problem.

Figure A.5: FEM setup for Stokes' first problem. BC are written in red. Mesh is shown in blue.  $L_y$  is set to 200, which emulates  $\infty$ .

with more complicated BC it is straightforward to obtain a solution. We choose a 2D grid with (arbitrary) width,  $L_x = 20$ , and length,  $L_y = 200$  to approximate infinity. The FEM setup can be seen in figure A.5. The FEM system equation for the heat equation is for good measure restated in eq. A.29 and using Crank-Nicolson stencil we get the resulting equation (re)stated in eq. A.30. The solution at a given time,  $t$ , can be seen in figure A.6. Previous iterations are seen as transparent planes below the solution. As expected there is now  $x$  dependence.

$$\underline{\underline{\mathbf{M}}} \dot{\mathbf{u}} + \underline{\underline{\mathbf{K}}} \mathbf{u} = \mathbf{f} \quad (\text{A.29})$$

$$(\underline{\underline{2\mathbf{M}}} + \Delta t \underline{\underline{\mathbf{K}}}) \mathbf{u}^{t+\Delta t} = \Delta t (\mathbf{f}^{t+\Delta t} + \mathbf{f}^t) + (\underline{\underline{2\mathbf{M}}} - \Delta t \underline{\underline{\mathbf{K}}}) \mathbf{u}^t \quad (\text{A.30})$$

To compare the solution to the analytical expression in eq. A.28 we draw both for a given time,  $t$ . As seen, there is agreement between the two functions. Naturally, there's still a numerical deviation between the two, which diminishes as the grid is refined.

The physical interpretation relevant for the first Stokes problem is a *spreading boundary layer* of momentum diffusion. One can choose to describe the boundary layer thickness by the  $y = \delta_{99}$  distance where the velocity is reduced to 1% of its original value. The theoretical value (dashed, red) should be [20, eq. 15.18]:

$$\delta_{99} = 3.64 \cdot \sqrt{\nu t} \quad (\text{A.31})$$

which can be compared to the simulation result for verification of ditto. The theoretical value amounts to  $\delta_{99} = 63.1m$  whereas the simulation gives

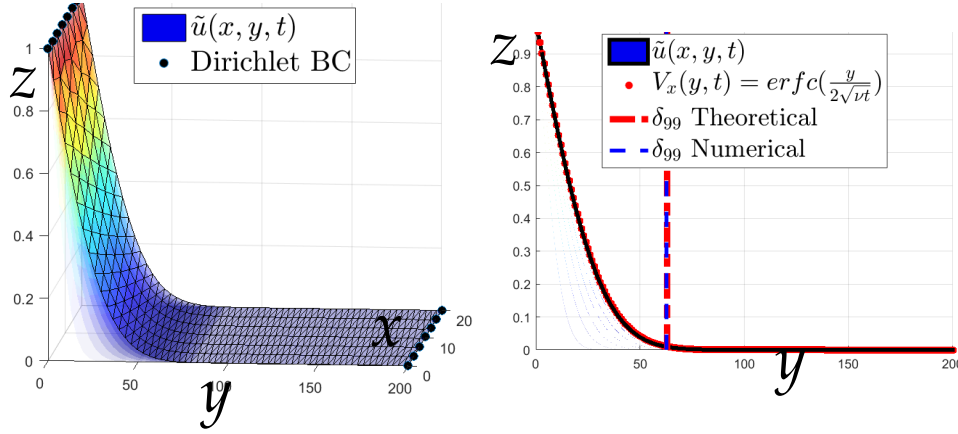


Figure A.7: FEM solution of Stoke's

Figure A.6: Stoke's first problem. The previous planes are seen below solution,  $V_x(y, t)$ , from eq. A.28. in transparent colors.

(dashed, blue)  $\delta_{99} s = 62.7m$ . A small discrepancy is to be expected since we only locate the nearest nodal value. For a more precise result one could simply interpolate across the element or refine the mesh.

### Stokes' Second Problem

In the second case things become slightly more complicated since we have a finite frequency,  $\omega = \omega_0$ , with which the red plate in figure A.4 is oscillating. We can use the analytical solution in eq. A.27 to get an expression for a steady state solution when time approaches infinity,  $t \rightarrow \infty$ . In this case all transient effects should have died out. The analytical expression is seen in eq. A.32 [20, eq. 15.12]. As seen, we hope to find an oscillating solution that is exponentially decaying. The physical interpretation is a damped shear wave spreading into the fluid from the plane. Using almost the same procedure and setup as sketched in figure A.5 we get the solution for a given time,  $t$ , seen in figure A.8. There is one obvious and one subtle change from the first problem: Obviously our BC we impose have changed. Since they're now time dependent we must add a relaxation-loop in our script, i.e. we allow our system to relax to a given state before once again changing the BC to the following value.

$$v_x(y, t) = e^{-ky} \cos(\omega t - ky) \quad : k = \sqrt{\omega/(2\nu)} \quad (\text{A.32})$$

Once again we compare the solution to the analytical expression in eq. A.32 by drawing both for a given time,  $t$ . As seen in figure A.9, there is agreement between the two functions. The numerical deviation between

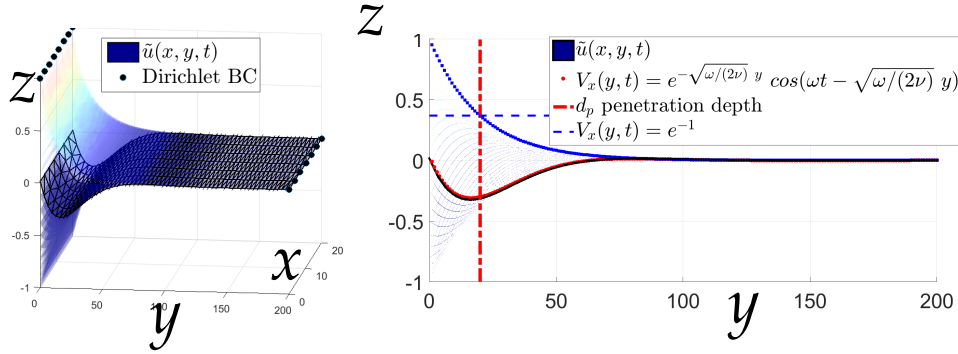


Figure A.8: Stoke's second problem. The previous planes are seen below in transparent colors.

Figure A.9: FEM solution of Stoke's second problem compared to the exact solution,  $V_x(y, t)$ , from eq. A.32. The blue squares is the exponential decay part of the analytical solution.

the two is more visible since we accumulate additional error for each relaxation loop. The loop, has a certain threshold below which it ignores the discrepancy to continue the simulation. Lower thresholds lead to a more accurate solutions just as finer grid minimizes error.

One can also assess the solution by calculating the *penetration depth* which is  $d = 1/k$  (red, dashed) and describes the decay length of the exponential. The blue dashed line shows when the decay term is reduced with a factor of  $e^{-1}$ . Obviously, the lines cross on the blue decay line. Though not visible in this figure there is a small gap between the decay from previous planes plotted in transparent colors and the blue decay line. Overall however, the FEM solution reproduces the expected solution within some error margin [20, p. 248].

## A.4 Flow around objects

**T**HE test case revolves around force calculations on an object. Here, the geometry, chosen to be a NACA-airfoil profile, certainly is more complex and poses a well-studied problem in the industry: *How to find the most aerodynamic angle?*

### A.4.1 Importance of functionals

The answer can readily be found by calculating a few *functionals* of the solution,  $u$ . It is for obvious reasons important to know how to calculate derived quantities of the solution like e.g. the flux vector field. This is the

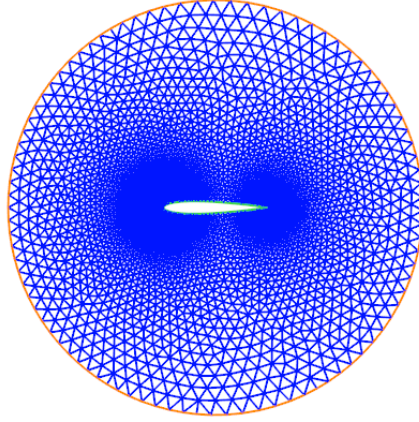


Figure A.10: Setup for the NACA airfoil with a circular mesh around it.

enormous benefit of performing CFD. One can obtain the answer to almost anything without even having to do an actual experiment! Authors: M. Mortensen, K-A Mardal and H. P. Lantangen even go as far as stating [17, p. 420]:

*"The results of carefully executed DNS have in the fluid mechanics community the same status as carefully executed experiments."*

We have already seen examples of functional computation such as the recently introduced `errornorm()` or the flux out of some particular part of the boundary. Thus, we will use this test case to show how to answer relevant posed questions such as the above stated and to introduce some important energy functionals for turbulent flow.

#### A.4.2 Setup and variational problem

The mesh with the airfoil obstacle in the center can be seen in figure A.10. Just as in the previous example we define our sub domains, i.e. outer boundary,  $\Gamma_o$ , obstacle boundary,  $\Gamma_i$  and domain,  $\Omega$ , each with a unique color in the figure.

To ensure that the obstacle is immersed in a uniform flow we assign a constant velocity,  $\mathbf{u}_{\Gamma_o}$ , on the outer boundary (orange) as defined in eq. A.33. For the present case  $U_0$  is set to 1 and  $\Theta_0$  is a fixed angle determining inflow direction. The obstacle boundary (green) is a no-slip boundary with  $\mathbf{u} = 0$ .

$$\mathbf{u}_{\Gamma_o}(\mathbf{x}) = U_0 \cos(\Theta_0) \hat{\mathbf{x}} + U_0 \sin(\Theta_0) \hat{\mathbf{y}} \quad : \quad \mathbf{x} \in \Gamma_o \quad (\text{A.33})$$

To complete the BCs we must also specify the pressure in a least one point. We therefore set the point to the far left to 0:

```
p_0 = Constant((0,0))
bc_p=DirichletBC(W.sub(1),p_0,"x[0]<-1+DOLFIN_EPS","pointwise")
```

In other words we apply Dirichlet conditions on the velocity field for both inner and outer boundaries (apart from one reference point). This changes our variational formulation from eq. 10.7b slightly since the test function must vanish on essential boundaries. Using the notation from eq. 10.7b we could simply try to write:

```
L = 0
```

it is however useful to remember that we simplified the equations a bit to start with. Thus, usually the linear form reads:

$$L((v, q)) = \int_{\Omega} f \cdot v \, dx - p_0 \int_{\Gamma} n \cdot v \, ds$$

Should we have no Neumann boundaries the linear form could be defined like:

```
L = inner(force, n)
```

with `force` being a dummy zero-vector if no forces were included (we actually also included a `force` term in the first test case). The reason for including a `force` here, even though we just simplified the equations before posing the problem is, that FEniCS will not recognize our `L = 0` as a linear form in a variational problem otherwise. The variational problem thus reads: (now with included dummy-force)

Seek  $(\mathbf{u}, p) \in W$  such that:

$$a((\mathbf{u}, p), (\mathbf{v}, q)) = L((\mathbf{v}, q)) \quad \forall (\mathbf{v}, q) \in W \quad (\text{A.34a})$$

$$\text{with } \mathbf{u} = \mathbf{u}_{\Gamma_o}(\mathbf{x}) \quad \text{on } \Gamma_o \quad (\text{A.34b})$$

$$\text{and } p = p_0 \quad \text{on } \Gamma_p \quad (\text{one point}) \quad (\text{A.34c})$$

where  $W = V \times P$  should be a mixed function space such that  $\mathbf{u} \in V$  and  $p \in P$ , and:

$$a((\mathbf{u}, p), (\mathbf{v}, q)) = \int_{\Omega} \nabla \mathbf{u} : \nabla \mathbf{v} - p \nabla \cdot \mathbf{v} + q \nabla \cdot \mathbf{u} \, dx \quad (\text{A.35a})$$

$$L((\mathbf{v}, q)) = \int_{\Omega} \mathbf{f} \cdot \mathbf{v} \, dx \quad (\text{A.35b})$$

### A.4.3 Flow solutions around an airfoil

Once the linear form,  $L((\mathbf{v}, q))$ , has been altered the solver can be invoked just as before. A solution to  $\mathbf{u}$  and  $p$  for  $\theta_0 = \frac{\pi}{5}$ ,  $U_0 = 1$  and  $p_0 = 0$  is seen in figure A.11(a) (showing magnitude of  $\mathbf{u}$ ) and A.11(c) (for  $p$ ). As expected, the velocity goes to zero around the obstacle and the pressure is close to zero (the  $p_0$ -value) except for the immediate vicinity of the obstacle.

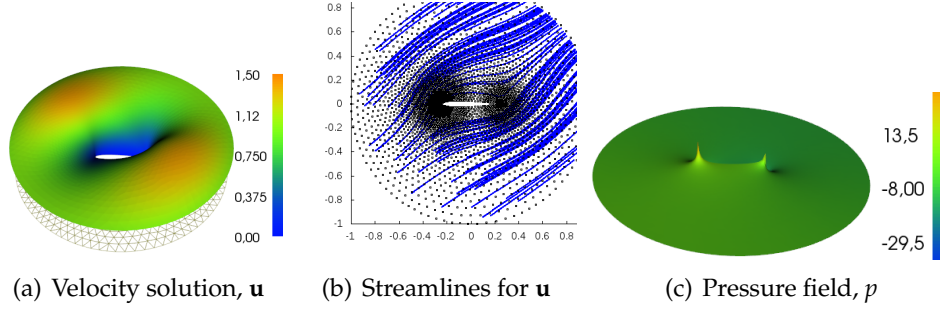


Figure A.11: Solutions to velocity and pressure field in *Flow around obstacle* test case. Assessing the solution,  $\mathbf{u}$ , at boundaries is highly telling. For incompressible fluids, the normal of the velocity field,  $u_n = \mathbf{u} \cdot \mathbf{n}$  must be continuous across interfaces (to *incompressible* obstacles). This is of course trivially satisfied as we have no-slip boundaries. These no-slip BCs at the obstacle boundary mean that the viscous fluid always stays at the boundary - i.e. it mirrors the materials velocity on the other side of the interface. This "velocity"-continuity across surfaces are usually expressed in brackets:  $[\mathbf{u}] = 0$  [20, p. 251].

To get a more intuitive visualization of the velocity field it is advantageous to plot the streamlines. We therefore randomly generate some particles within the domain and advect them. Once the velocity has been integrated up we obtain the streamlines as seen in figure A.11(b).

#### A.4.4 Functional-based analysis

We are now ready to dive into the dynamics of the incompressible Newtonian fluid.

##### Isotropic viscous stress

To measure how sheared the fluid is we start out by computing the total (Newtonian) stress as defined in eq. A.36 [20, eq. 15.20]:

$$\sigma_{ij} = -p \delta_{ij} + \mu(\nabla_i v_j + \nabla_j v_i) \quad (\text{A.36})$$

The total stress tensor is easily computed in FEniCS. The following lines will do:

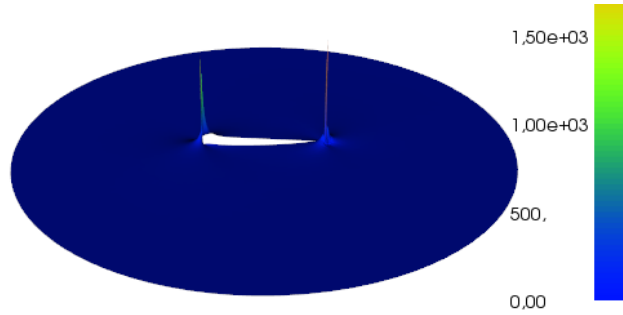


Figure A.12: The second deviatoric stress invariant defined in eq. A.37 visualizing how sheared the fluid is.

```
mu = 1 # dummy mu
stress_visc = mu*2*sym(nabla_grad(u))
stress = -p*Identity(2) + stress_visc
```

where the last line corresponds to eq. A.36. The `sym()` is an operator from the Unified Form Language (UFL) that FEniCS uses to express variational forms as well as functionals. A quick look in the online documentation will reveal that the command computes the symmetric part of a tensor:  $\text{sym}(\mathbf{A}) = \frac{1}{2}(\mathbf{A} + \mathbf{A}^T)$ , as is required. Now, to plot this in FEniCS (more advanced plotting tools will be introduced later) we have to create a function space for the tensor and project  $\sigma$  into the space before we can plot it:

```
T = TensorFunctionSpace(mesh, 'CG', 1)
stress_projected = project(stress, T)
plot(stress_projected[i,j])
```

here  $i$  and  $j$  can both take the values 0 or 1 in 2D thus yielding 4 combinations to be plotted. Since we would like just one plot to express the shear we compute instead the *Second Deviatoric Stress invariant*,  $J_2$ , as defined in eq. A.37 where  $\sigma_{visc}^2$  is the viscous stress previously computed as `stress_visc`.  $J_2$  is a metric for how sheared the fluid is and the visualization is seen in figure A.12. Not surprisingly, the fluid is sheared the most close to the obstacle.

$$J_2 = \frac{1}{2} \text{tr}(\sigma_{visc}^2) \quad (\text{A.37})$$

### Aerodynamic profile

To examine the question posed in the very beginning of this test case about the aerodynamic profile of the airfoil one can calculate the traction on the obstacle boundary and decompose it into a drag force,  $F_{drag}$ , and a lift force,  $F_{lift}$ , using the direction from the imposed surrounding uniform flow.

Here,  $F_{drag}$  is parallel to the imposed inflow direction and  $F_{lift}$  orthogonal to ditto [20, p. 287]. These forces sum up to the so called *reaction force*,  $\mathcal{R}$ , that the fluid exerts on a body [20, eq. 17.2]:

$$\mathcal{R} = \oint_S \sigma \cdot d\mathbf{S} = F_{drag} + F_{lift} \quad (\text{A.38})$$

According to Cauchy's stress hypothesis the force is connected to the stress tensor field as seen in eq. A.39 (also evident from eq. A.38), where  $d\mathbf{S} = \hat{\mathbf{n}} dS$  and  $\hat{\mathbf{n}}$  is the normal to the surface.

$$dF_i = \sum_j \sigma_{ij} dS_j \quad (\text{A.39})$$

We therefore need only dot the stress,  $\sigma$ , with a normal,  $\hat{\mathbf{n}}$ , and decompose the computed *stress vector*. The few lines below should do the trick:

```
ds = Measure("ds")[subdomains]
n = FacetNormal(mesh)
traction = dot(stress, n)
F_drag = assemble(dot(traction, n_flow) * ds(mark["inner_boun"]))
F_lift = assemble(dot(traction, t_flow) * ds(mark["inner_boun"]))
```

here `n_flow` and `t_flow` are predefined directions as described earlier. The definition of `ds` is compulsory since we want to sum-up (`assemble()`) along a boundary and surface normal, `n`, is simply defined from the mesh we use.

We are now ready to analyze the airfoil profile. The few lines of calculation can be done while varying the inflow angle,  $\Theta_0$ , to locate the optimal aerodynamic angle. The result is seen in figure A.13. For *creeping* flow such as this, the  $F_{drag}$  and  $F_{lift}$  derived from the same contact forces can be expected to be "of the same order of magnitude" far from containing boundaries ( $\Gamma_o$ ) [20, p. 286]. This seems to be congruent with our results. As expected, the lift force is oscillating around 0 while the drag force (per definition a positive entity) is smallest when the flow is along the center axis of the airfoil. Also the anticipated results.

Has this calculation any relevance in the industry or is it merely a pedagogical CFD-exercise? Actually, the above can be seen as a minimal example in *aerodynamic shape optimization* - applicable within aircraft, race car and wind turbine production just to name a few. Changing viewpoint in the above example one can interpret the fluid angle as fixed but see the airfoil geometry as changing by continuously varying a design-parameter. In this case the angle. We then look at a *cost function* we want to maximize



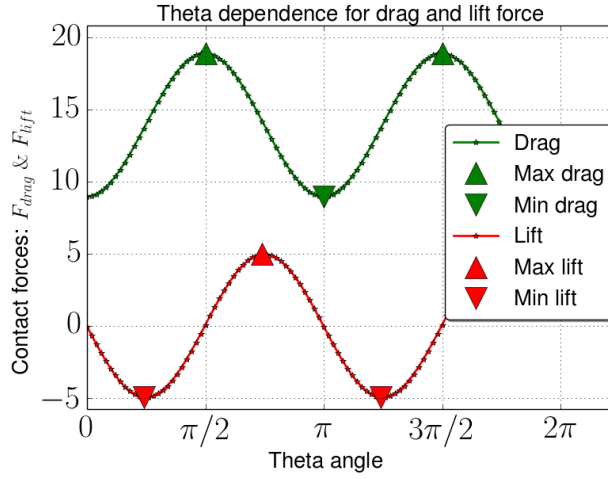


Figure A.13: Drag force and lift force as theta is varied from  $[0; 2\pi]$ . The symmetry from the setup and airfoil can be recognized in the computed functionals. Notice, the drag force has maximal magnitude when inflow is perpendicular to the long axis of the airfoil (at  $\Theta_0 = \pi/2 \wedge 3\pi/2$ ).

or minimize, e.g. lift force or drag force. The above would then be an example of older methods in CFD for shape optimization by coupling the CFD code to some numerical optimization method (in the above we locate extrema in figure A.13). Usually however, one changes many design variables and thus the ensuing computations: finding gradient of cost function to discern steepest descend direction and then finding the resulting minimum in the many-dimensional parameter landscape is computationally *very* demanding. Luckily, smarter methods such as *Adjoint Solvers* exists and is currently a hot research topic. Here, one can define an *adjoint equation* and solve it only once to obtain the cost function gradient. Thus saving enormous amount of computation time. We will not implement an adjoint solver in the present work as it is a daunting task but it could in principle be coupled to the solver at this point [29].

### Energy functionals

Apart from the `errornorm()` functional the various energy functionals related to  $\mathbf{u}$  are perhaps the most important ones. Two with utmost significance is the kinetic energy,  $K(\mathbf{u})$ , and its rate of change  $\frac{d}{dt}K(\mathbf{u})$ , defined in eq. A.40a and A.40b respectively [46, eq. 2.11-12] (here  $\langle \cdot, \cdot \rangle$  denotes the L2-inner product of two vector fields on our domain):

$$K(\mathbf{u}) = \frac{1}{2} \langle \mathbf{u}, \mathbf{u} \rangle = \frac{1}{2} \|\mathbf{u}\|^2 \quad (\text{A.40a})$$

$$\frac{d}{dt}K(\mathbf{u}) = -\nu \|\nabla \mathbf{u}\|^2 + \langle f, \mathbf{u} \rangle \quad (\text{A.40b})$$

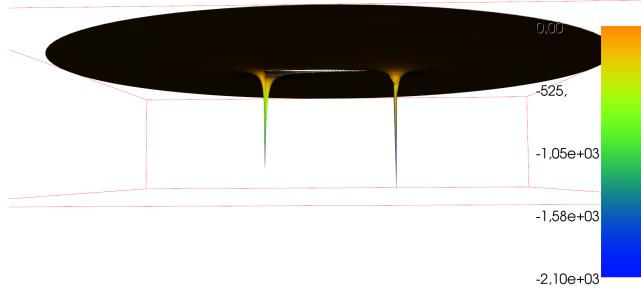


Figure A.14: The rate of change of kinetic energy as defined in eq. A.40b. Since there is no body forces included we expect a wholly negative field.

Both entities can be computed in a similar fashion to the previous functionals:

```
P = FunctionSpace(mesh, "CG", 1)
E_kin = project(0.5*inner(u,u),P)
nu = 1. # dummy nu
E_change = project(-nu*inner(nabla_grad(u),nabla_grad(u)), P)
```

Plotting the kinetic energy yields a plot similar but in scaling to figure A.11(a) and will be omitted here. The resulting plot for the rate of change in kinetic energy is seen in figure A.14. Since we have omitted body forces we get an entirely negative field close to 0 save for the extrema near the airfoil-tips. Obviously, we are solving a steady state problem and as such we shouldn't worry too much about the *rate-of-change* of anything in the present case - but the computation procedure can be introduced regardless.

It is not too custom to visualize the rate-of-change field as in figure A.14. It is however very important to be able to calculate the total dissipation of kinetic energy by integrating over the entire domain,  $\Omega$ . The total kinetic energy and the total energy dissipation rate,  $\epsilon$ , (with a sign-change) can be seen in eq. A.41a and A.41b [17, eq. 22.9,45-46]:

$$\bar{K} = \int_{\Omega} \frac{1}{2} u \cdot u \, dx \quad (\text{A.41a})$$

$$\epsilon = \nu \int_{\Omega} \nabla u : \nabla u \, dx \quad (\text{A.41b})$$

We are now ready to find the (volumetric) average of dissipation, which is said to be the "single most important measure of a turbulent flow" [17, p. 436]:

```
eps = assemble(nu*inner(nabla_grad(u), nabla_grad(u))*dx)/(pi*r**2)
print "Total dissipation: %.2f " %eps
```

Total dissipation: 3.92

Thus, in a transient case we would loose kinetic energy with a rate of 3.92 (per volume) for each time-step. Reassuringly, since it would be truly troublesome if we created energy.

### Convection term

Why is the dissipation even interesting? One could point to many reasons but perhaps it is easily seen when discussing *Convection*: The convective term can be put on the form in eq. A.42 for some vector field  $a$  being convected. In turn, it can be shown that the convection term is nothing but *transport of kinetic energy*,  $K(u)$ , as seen in eq. A.43 [17, eq. 22.4,8] meaning that the convection term should not contribute to the accumulation of  $K(u)$  - it should only "push it around".

$$B(u, a) = u \cdot \nabla a \quad (\text{A.42})$$

$$\langle B(u, u), u \rangle = \int_{\Omega} u \cdot \nabla K(u) dx \quad (\text{A.43})$$

It can actually be shown that only the viscosity term in the NS-equations should lead to dissipation of kinetic energy whereas the convective transport should obey energy conservation [17, p. 422]. Most conveniently, this is a property we can now readily check should we have the need since we have the means of calculating dissipation,  $\epsilon$ . For inviscid fluids (e.g. the Euler equations) we should therefore *not* have dissipation and we would have obtained  $\epsilon = 0$ .

The above considerations of dissipation is also important when assessing the chosen iterative scheme. As explained in the literature, the choice of iteration scheme will in turn effect the discrete solution. Some schemes (e.g. central difference) are dispersive but conserve the discrete kinetic energy in time. Other schemes (e.g. assymetric schemes) will be dissipative and lead to loss of energy over time [17, p. 423].

While on the topic of *convection* it might be fruitful to assess the above completed exercise of an object in a uniform flow. Although we obtained meaningful results - there is reason for concern physically. As stated by the Stokes' paradox, unbounded Stokes flow cannot occur in 2D, thus the

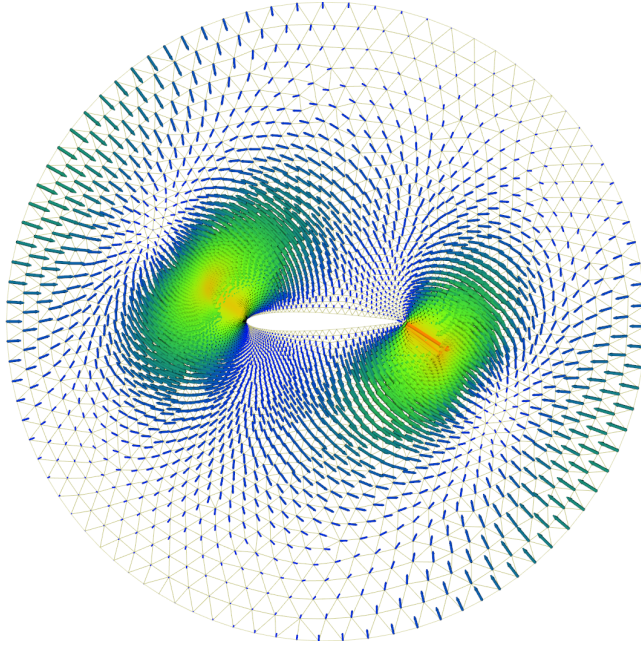


Figure A.15: The convection term  $(\mathbf{u} \cdot \nabla)\mathbf{u}$  computed from the Stokes flow solution  $\mathbf{u}$ . The term has been projected into a vector space and is shown on top of the mesh. As seen, the convective forces are (presumably) largest near the obstacle and will push the fluid away from the no-slip boundary.

simulation lacks realism. Where exactly is it our solution has the highest discrepancy compared to a true NS-solution? To get a feeling for this one can compute  $(\mathbf{u} \cdot \nabla)\mathbf{u}$  as a functional of (the Stokes flow)  $\mathbf{u}$  and then plot the resulting field. This can be visualized in just three lines with the result seen in figure :

```
V_conv = VectorFunctionSpace(mesh, 'Lagrange', 1)
conv = project(nabla_grad(u)*u, V_conv)
plot(conv)
```

### Drag force and Reynolds dependence

Including the inertia forces by adding the convection term,  $(\mathbf{u} \cdot \nabla)\mathbf{u}$  allows us to study the change in  $F_{drag}$  as we vary the Reynolds number,  $Re$ . The altered test problem equations can be seen in eq. A.44a and A.44b. Thus, we now have the parameter,  $nu$ , but the density,  $\rho$ , is still left out (or scaled into the pressure;  $(1/\rho) \cdot p \rightarrow p$ ).

$$\nu \nabla^2 \mathbf{u} - (\mathbf{u} \cdot \nabla)\mathbf{u} = \nabla p \quad (\text{A.44a})$$

$$\nabla \cdot \mathbf{u} = 0 \quad (\text{A.44b})$$

Since the equations have changed, we must revise the way we feed FEniCS the variational problem. More importantly, a different solver must

be called, since the advection term adds non-linearity. Actually, an entirely different type of solver is needed; an *iterative* solver which we will explain further in the following test case. Now, it suffices to state the altered (and non-linear) variational problem and present the drag force dependence on Reynolds number,  $Re$ .

Find  $(\mathbf{u}, p) \in W$  such that:

$$F((\mathbf{u}, p); (\mathbf{v}, q)) = 0 \quad \forall (\mathbf{v}, q) \in W \quad (\text{A.45a})$$

$$\text{with } \mathbf{u} = \mathbf{u}_{\Gamma_o}(\mathbf{x}) \quad \text{on } \Gamma_o \quad (\text{A.45b})$$

$$\text{and } p = p_0 \quad \text{on } \Gamma_p \quad (\text{one point}) \quad (\text{A.45c})$$

where  $W = V \times P$  should be a mixed function space such that  $\mathbf{u} \in V$  and  $p \in P$ , and:

$$\begin{aligned} F((\mathbf{u}, p); (\mathbf{v}, q)) = & \int_{\Omega} \nu(\nabla \mathbf{u} : \nabla \mathbf{v}) + (\mathbf{u} \cdot \nabla \mathbf{u}) \cdot \mathbf{v} \\ & - p \nabla \cdot \mathbf{v} + q \nabla \cdot \mathbf{u} \, dx \end{aligned} \quad (\text{A.46})$$

As seen, for non-linear problems there is a new canonical form for the variational problem:  $F((u, p), (v, q)) = 0$ . The solver is correspondingly called like:

`solve(F == 0, bcs, J=J)`

where the non-linear functional  $F$  is the LHS in the weak formulation and  $J$  is the Jacobian to be defined before invoking the solver. As seen in eq. A.46, a constant for the viscosity,  $\nu$ , now figures in the equations. To analyze the dependence on the Reynolds number,  $Re$ , for  $F_{drag}$  we calculate the drag force as before but now for a fixed angle:  $\Theta_0 = 0$  and compute the Reynolds number as:  $Re = 1/\nu$ . This is repeated for Reynolds numbers  $[10^{-5}; 10^1]$  (see x-axis) and visualized in figure A.16. We can finally check the proposed proportionality  $F_{drag} \propto Re^{-1}$  for small  $Re$  and a given  $\Theta_0$ . Luckily, for small  $Re$  values there does indeed seem to be the proposed scaling law:  $F_{drag} \propto Re^{-1}$  [21].

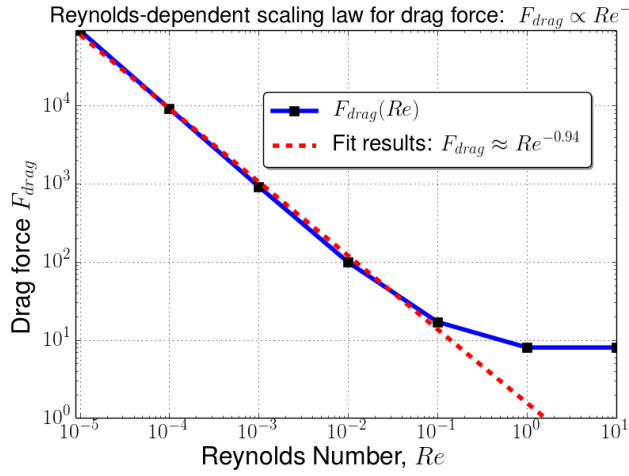


Figure A.16: Drag force's dependence on  $Re$  for a fixed angle  $\Theta_0 = 0$ . The fit result;  $r = -0.94$  is fairly close to the suspected  $F_{drag} \approx Re^{-1}$  proportionality.

## A.5 Lid-driven cavity



HERE we present the famous, classical test case, 'Lid-driven cavity flow', for fluid solvers.

The last test case for the FEniCS implementation is one of the traditional and most known test problems; the Lid-driven cavity. This short example merely serves the purpose of showing how to obtain a more controlled solution to a non-linear problem.

### A.5.1 Setup

In its essence it is a box with three no-slip boundaries while the final side (lid) has a prescribed velocity, thus  $\mathbf{u}_{wall} = 0$  and  $\mathbf{u}_{lid} = U_0 \cdot \hat{\mathbf{x}}$ . The setup can be seen in figure A.17.

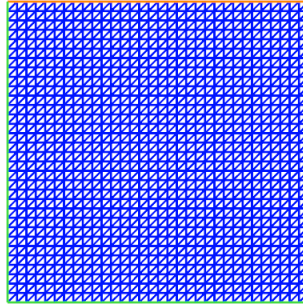


Figure A.17: Setup for the classic lid-driven cavity test.

### A.5.2 Variational problem

Since we have already taken the trouble to introduce inertia, our starting point will simply be the previously mentioned steady flow equations A.44a and A.44b. Thus, the variational problem reads:

Find  $(\mathbf{u}, p) \in W$  such that:

$$F((\mathbf{u}, p); (\mathbf{v}, q)) = 0 \quad \forall (\mathbf{v}, q) \in W \quad (\text{A.47a})$$

$$\text{where } \mathbf{u} = \mathbf{u}_{\text{lid}} \quad \text{on } \Gamma_{\text{lid}} \quad (\text{A.47b})$$

$$\text{and } \mathbf{u} = \mathbf{u}_{\text{wall}} \quad \text{on } \Gamma_{\text{wall}} \quad (\text{A.47c})$$

$$\text{while } p = p_0 \quad \text{on } \Gamma_p \quad (\text{one point}) \quad (\text{A.47d})$$

where  $W = V \times P$  should be a mixed function space such that  $\mathbf{u} \in V$  and  $p \in P$ , and:

$$\begin{aligned} F((\mathbf{u}, p); (\mathbf{v}, q)) = & \int_{\Omega} \nu(\nabla \mathbf{u} : \nabla \mathbf{v}) + (\mathbf{u} \cdot \nabla \mathbf{u}) \cdot \mathbf{v} \\ & - p \nabla \cdot \mathbf{v} + q \nabla \cdot \mathbf{u} \, dx \end{aligned} \quad (\text{A.48})$$

### A.5.3 Newton's Method

Once the BCs have been correctly imposed the non-linear solver can be called to obtain  $\mathbf{u}$  and  $p$ . Above was given a compact one-liner for solving the problem. This is unfortunately not always enough to solve the problem as often times parameters must be tuned a bit. Newton's method is seen in eq. A.49a and A.49b (where  $N$  is degrees-of-freedom) [19, eq. 72-73]. Here,  $W_i$ , is the unknown nodal values we are solving for and  $F_i$  is the discrete version of eq. A.48 arising when inserting  $u = \sum_{j=1}^{N_u} U_j \phi_j$  and ditto for pressure. As seen, the relaxation parameter,  $\omega$ , enters. It is by default 1 but for some problems it can be necessary to use under-relaxation by lowering  $\omega$  to obtain convergence. The Jacobian,  $J = \frac{\partial F_i}{\partial W_j}$ , also enters in the method and must be given to the solver. Luckily, computing the Jacobian,  $J$ , can easily be done by exploiting that UFL can differentiate forms. We simply call the UFL function,

`J = derivatie(F,w)`

where we give it the specified problem,  $F$ , and the solution vector,  $w$ . Clearly, the automated `derivatie()` function is very handy for tedious expressions.

$$\sum_{j=1}^N \frac{\partial F_i}{\partial W_j} \delta W_j = -F_i \quad : i = 1, \dots, N \quad (\text{A.49a})$$

$$W_j^{k+1} = W_j^k + \omega \delta W_j \quad (\text{A.49b})$$

Having defined the Jacobian as above we are now ready to call the solver and set various parameters:

```
problem = NonlinearVariationalProblem(F, w, bcs, J)
solver = NonlinearVariationalSolver(problem)
solver.parameters['newton_solver']['relative_tolerance'] = 1e-5
solver.parameters['newton_solver']['relaxation_parameter'] = 0.7
```

Solving non linear problems is treated extensively in the FEniCS guide section 3.4 covering both Picard iteration and several versions of Newton's method and is hereby recommended for interested readers [19].

#### A.5.4 Solution to Lid-Driven Cavity

Having set up the non-linear solver as described above, it is a matter of calling:

```
solver.solve()
```

to get the solution. We randomly generate particles in the domain to obtain the streamlines for  $\mathbf{u}$  seen in figure A.18 and A.19 for Reynolds numbers 10 and 400 respectively. As expected, the enforced velocity of the lid results in a circulative motion in the encapsulated fluid and as the Reynolds number increases we get a much more lively fluid.



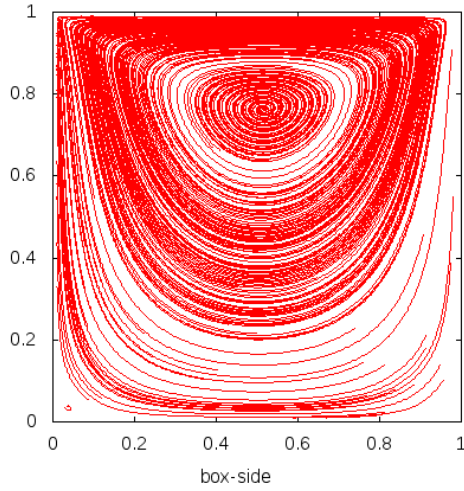


Figure A.18: Classic lid-driven cavity test with  $Re = 10$ .

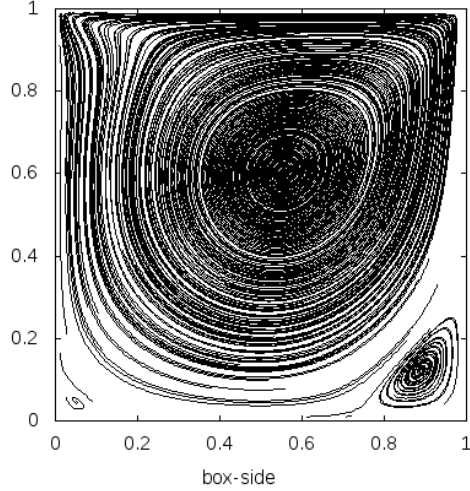


Figure A.19: Classic lid-driven cavity test with  $Re = 400$ .

## A.6 Supplementary Oasis information

**T**HIS section contains additional investigations and information on the Oasis solver. All the information is of course relevant background knowledge but on the other hand not instrumental to the understanding of the thesis' main scope. Thus, it has been reallocated to the appendix.

### A.6.1 Oasis verification

In the verification section of Oasis (12.2.2) we presented a convergence plot towards the steady state solution of  $u(x, y, t)$  in a duct. Should the figure (12.3) give any reason for concern or just to be thorough one can of course repeat the exercise we did when presenting our own Chorin-solver. Setting up a (cylindrical) pipe with the same settings as before gives a plot very similar to figure 11.4. The result for Oasis can be seen in figure A.20.

### A.6.2 Convection term

Previously, it has been shown how to obtain the steady state flow of the full NS-equations (in "Lid-Driven Cavity"). We have however solved a linear system here due to scaling-simplicity and simulation time. Visualizing

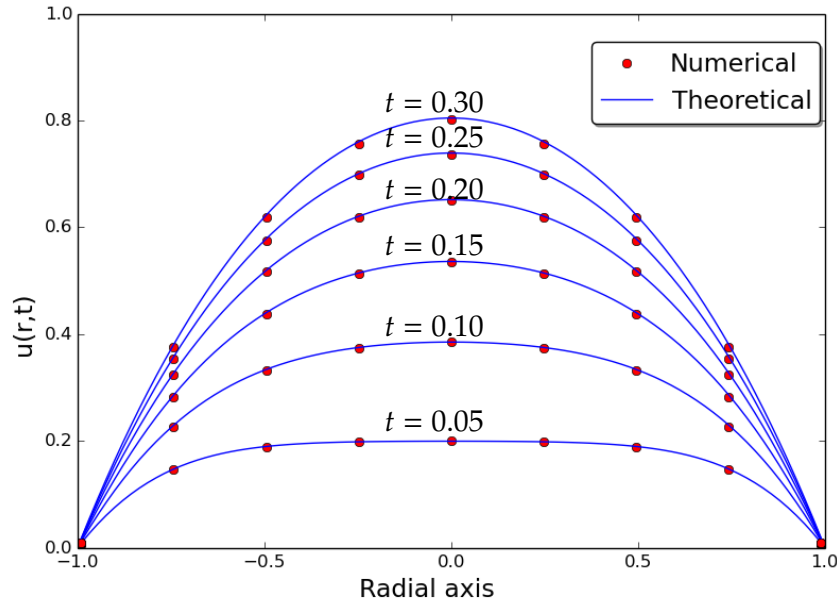


Figure A.20: Comparison between NS simulation in a cylindrical mesh using Oasis ('Numerical') and the analytical solution in eq. 11.5 ('Theoretical'). There is a clear agreement between simulation and analytic velocity profile. The corresponding comparison between the Chorin solver and the analytical result is seen in figure 11.4.

the projection of the convection term,  $(\mathbf{u} \cdot \nabla)\mathbf{u}$ , onto the space we work in have already been introduced in the exercise "Flow around object" and can allude to where we could expect a discrepancy between solution to Stokes and NS-equation. A zoom of one processor cut through the middle after having projected the convection term is seen in figure A.21. For obvious reasons, the term is zero at the wall boundaries. The red maxima values are concentrated in the vicinity just above the bump as one would expect. This plot only shows the magnitude of the functional. Extracting the  $y$ -component as in figure A.22 we find close to the same maximum for a "vertical" slice and learn that a major part of the forces would have pushed the fluid toward the center of the duct. One could come up with many such "sanity"-checks to a solution by looking at functionals of  $\mathbf{u}$  and checking if they result in physically meaningful interpretations.

### A.6.3 Other initialization choices

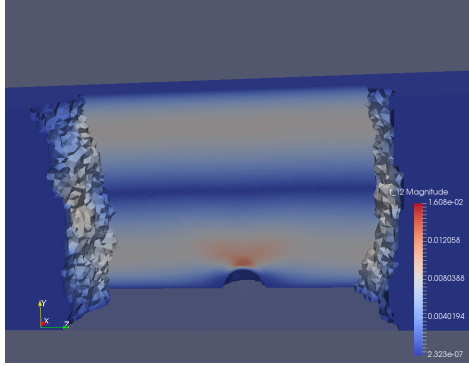


Figure A.21: The resulting plot of a slice through one of the 28 processors after having projected the convection term,  $(\mathbf{u} \cdot \nabla)\mathbf{u}$  onto the velocity vector function space. The red maxima values are concentrated in the vicinity just above the bump as expected.

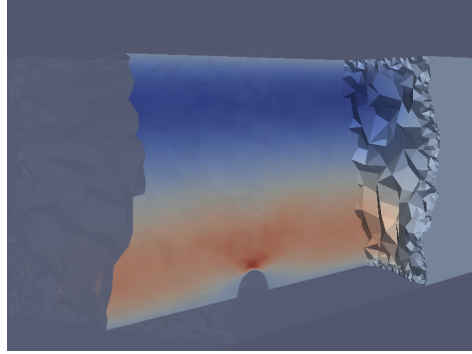


Figure A.22: A slice showing magnitude of y-component of the projected convection term through the same processor as seen in figure A.21 (this time on the coarse duct mesh).

To solve for the steady state profile and subsequently interpolate it to the mesh using the correct scaling is not the only way to start simulating. Two other common choices will briefly be mentioned below.

### Optimal initialization

Perhaps the best way to start a simulation is to simply run it from zero. When the LBM was used on these ducts it was by simulating from zero (on the coarse mesh) and then interpolating to the finer mesh once turbulence was eminent. The reason for not doing this is as mentioned the vast amount of time needed to get to the onset of turbulence. Once done, it can however give an immediate plot of how a starting point profile ideally should look like. An example is given in figure A.23. The simulation time is  $T = 280s$  and to obtain fully developed turbulence one probably has to exceed  $T = 2000s$  so a lot of simulation is still needed before one can make turbulence measurements. In figure A.23 one recognizes the higher velocity above the bump. Due to mass conservation every slice of pipe must experience the same flux. This is only possible if velocity increases when cross sectional area diminishes. This is exactly what we see in figure A.23

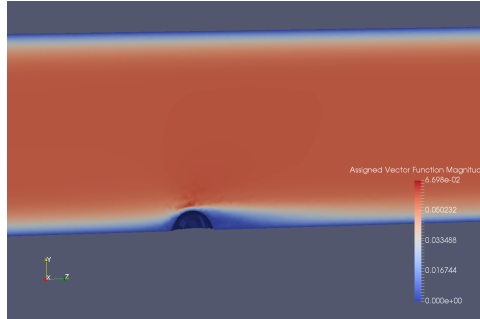


Figure A.23: A velocity profile from a simulation initiated at zero. One recognizes the higher velocity above the bump. Due to mass conservation every slice of pipe must experience the same flux. This is only possible if velocity increases when cross sectional area diminishes. Hence, the higher velocities above the bump. The iteration time for this particular frame is  $T = 280$ .

### Initialization from perturbed state

It is very common to initialize a simulation from a perturbed state. A most relevant example is found in the Oasis git in the `channel.py`-script where a base flow is created and noise is added from a random stream function. Noise can also enter the system in undesired ways as mentioned below. A key concept here is what *kind* of noise we add:

### Server pitfalls

NBI has several servers of which 'Spock' and 'Scissors' have been utilized to produce the majority of the results. The two have slightly different versions of software since it is problematic to completely stop one server and thereby denying all employees access for an amount of time in order to update the software. As a result care must be taken when exchanging data between the two.

The velocity field in figure A.24 is a snap shot of a simulation right after we scale the unity steady state fields to the desired scaling. After a few iterations the result folder was then copied to Scissors and then simultaneously a few iterations were computed on both servers with the same parameters. The figure shows the resulting velocity field for Scissors (LHS) and Spock (RHS) in order to compare them. As seen the difference is striking and is due to a difference in the degree of freedom mapping.

It is very custom to perturb a starting-point velocity profile to provoke the onset of turbulence. As such, it seems a tempting choice to use this discrepancy in mapping to emulate a perturbation. Running the simulation on Scissors for a few thousand iterations also produces velocity profiles that could look like fully developed turbulence. An example is

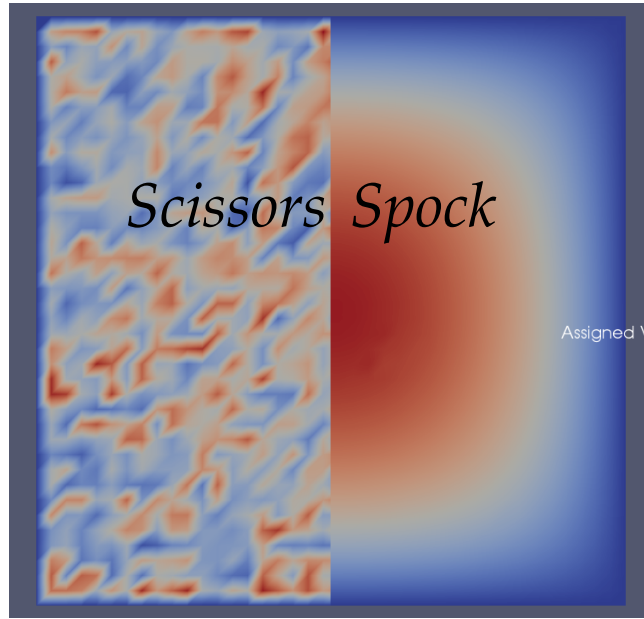


Figure A.24: Comparison of Degree-of-freedom-to-vertices mapping (`dof_to_vertex_map()`) for the two servers: Spock and Scissors. Clearly, the laminar profile from Spock is distorted.

given in figure A.25.

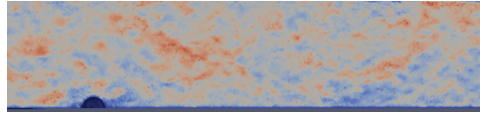


Figure A.25: A velocity field from Scissors after having copied a starting point from Spock and iterated a few thousand time-steps.

This is however, not unequivocally a good idea as it could violate the incompressibility of the fluid,  $\nabla \cdot \mathbf{u} = 0$ . Should one desire to start from a perturbed state it is usually, a good idea first to create a random stream function,  $\Psi$ , as an `expression()`, and then interpolate it to the domain to get a discrete function,  $\Psi_d$ , defined for each node in the mesh. We then get the initial velocity,  $\mathbf{u}_0$ , simply by taking the curl as in eq. A.50a. As stated in eq. A.50b the divergence of the curl of (any)  $\mathbf{u}_0$  will be zero and we thereby avoid violating the incompressibility.

$$\mathbf{u}_0 = \nabla \times \Psi_d \quad (\text{A.50a})$$

$$\nabla \cdot (\nabla \times \Psi_d) = 0 \quad (\text{A.50b})$$

### A.6.4 Kinetic energy plots for Oasis and LBM

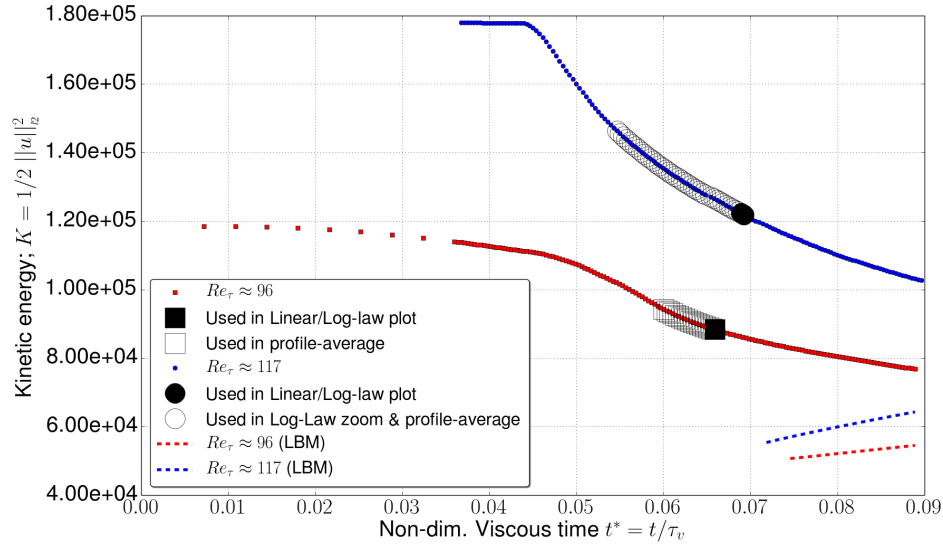


Figure A.26: Kinetic energy for  $Re_\tau = 96$  (red) and  $Re_\tau = 117$  (blue).  $Re_\tau$  values refer to the *final* statistically steady states. The clearly visible kink on the blue curve resembles the onset of turbulence in the duct. A larger initial  $\Delta t$  for 10 first time-steps for the red curve has unfortunately smeared the effect. The ten first time-step have been cut out but the red diamond marks the true kinetic start value for the red curve. Empty, colorless markers resemble data used to produce black mean profiles in figure 14.1(a) and 14.1(b). Blue markers show data used in figure 14.4(a) and 14.4(b).

## A.7 Scripts

**A**LL scripts are available upon request. I have tried to give code-examples throughout the thesis and thus, in principle one should be able to put together several scripts. Should that exercise seem less amusing one can naturally also get the scripts used in the thesis. Only one script will explicitly be provided: It is the Oasis script that generated turbulence data. It is put together of Oasis code blocks and further information can be found there.

```

1 ''' Matlab/Python scripts have amounted to over 10.000 lines which would pose quite a problem for ↵
   the rain forest should numerous people want to print the entire thesis appendix '''
2
3 print "One can simply write a mail to bkl886@alumni.ku.dk for any desired code producing any one ↵
   figure"
4
5 raw_input("<<press Enter to exit>>")

```

```

1 __author__ = "Mads Holst Aagaard Madsen <bkl886@alumni.ku.dk>"
2 __date__ = "2016-05-04"
3 __copyright__ = "Copyright (C) 2016 " + __author__
4
5 from ..NSfracStep import *
6 import mshr
7 from os import getcwd, makedirs
8 import cPickle # to load parameters
9 from fenicstools import interpolate_nonmatching_mesh
10 from fenicstools import StatisticsProbes
11 from numpy import linspace, repeat, where, resize, size, shape
12 import numpy as np
13 import h5py as h5l
14
15 restart_folder = 'LBM2400_results/data/1/Checkpoint'
16 # restart_folder = None
17
18 # Create a mesh here
19 def mesh(refine=10, **params):
20     mesh = Mesh()
21     #mesh_str = "mesh/coarse_duct.h5"
22     mesh_str = "mesh/fineest_duct.h5"
23     hdf = HDF5File(mesh.mpi_comm(), mesh_str, "r")
24     hdf.read(mesh, "/mesh", False)
25     return mesh
26
27 class PeriodicDomain(SubDomain):
28     def inside(self, x, on_boundary):
29         return bool(near(x[2], 0) and on_boundary)
30     def map(self, x, y):
31         y[0] = x[0]
32         y[1] = x[1]
33         y[2] = x[2] - 40.0
34
35 constrained_domain = PeriodicDomain()
36
37 ### If restarting from previous solution then read in parameters #####
38 if restart_folder:
39     restart_folder = path.join(getcwd(), restart_folder)
40     f = open(path.join(restart_folder, 'params.dat'), 'r')
41     NS_parameters.update(cPickle.load(f))
42     NS_parameters['T'] += 15000 * NS_parameters['dt']
43     NS_parameters['dt'] = 0.1
44     NS_parameters['checkpoint'] = 1000
45     NS_parameters['save_step'] = 1000
46     NS_parameters['restart_folder'] = restart_folder
47     NS_parameters['update_statistics'] = 100
48     NS_parameters['save_statistics'] = 1000
49     globals().update(NS_parameters)
50 else:

```

```

51 # Override some problem specific parameters
52 nu = 9.e-6
53 NS_parameters.update(dict(
54     T = 1000.0,
55     dt = 0.1,
56     nu = nu,
57     checkpoint = 1000,
58     plot_interval = 1,
59     save_step = 1000,
60     folder = 'LBM2400_results',
61     max_iter = 1,
62     velocity_degree = 1,
63     use_krylov_solvers = True
64 )
65 )
66
67 def walls(x, on_boundary):
68     return on_boundary and ((near(x[1],0.0) or near(x[1],1.0) or near(x[0],0.0) or near(x[0],1.0)) or (x[2]>0 and x[2]<40.0))
69
70 def create_bcs(V, sys_comp, **NS_namespace):
71     bcs = dict(ui, []) for ui in sys_comp
72     bc0 = DirichletBC(V, 0., walls)
73     bcs['u0'] = [bc0]
74     bcs['u1'] = [bc0]
75     bcs['u2'] = [bc0]
76     return bcs
77
78 F_0 = 1.2e-5 # approx Re=2400
79 def body_force(**NS_namespace):
80     return Constant((0.0,0.0,F_0))
81
82 def initialize(V,q_interpolate_nonmatching_mesh, q_1, q_2, VV, t, nu, dt, **NS_namespace):
83     """Initialize solution.
84     Use found unity steady state:
85     """
86     if restart_folder is None:
87         m2 = Mesh()
88         mesh_str = "mesh/fineest_duct.h5"
89         hdf = HDF5File(m2.mpi_comm(), mesh_str, "r")
90         hdf.read(m2, "/mesh", False)
91         V_old = VectorFunctionSpace(m2, "CG", 1)
92         P_old = FunctionSpace(m2, "CG", 1)
93         p = Function(P_old, 'steady/results/' + mesh_str[5:11] + '/p.xml')
94         u0x = Function(P_old, 'steady/results/' + mesh_str[5:11] + '/v_x.xml')
95         u0y = Function(P_old, 'steady/results/' + mesh_str[5:11] + '/v_y.xml')
96         u0z = Function(P_old, 'steady/results/' + mesh_str[5:11] + '/v_z.xml')
97         # scaling to turbulent pseudo steady state:
98         F_0 = 1.2e-5 # approx Re=2400
99         ux_array = u0x.vector().array()
100         uy_array = u0y.vector().array()
101         uz_array = u0z.vector().array()
102         p_array = p.vector().array()
103         ux_array *= F_0/nu
104         uy_array *= F_0/nu
105         uz_array *= F_0/nu
106         p_array *= F_0
107         u0x.vector()[:] = ux_array
108         u0y.vector()[:] = uy_array
109         u0z.vector()[:] = uz_array
110         p.vector()[:] = p_array
111         for ui in q_:

```



```

112         if ui == 'p':
113             vv = interpolate_nonmatching_mesh(p,VV[ui])
114             q_[ui].vector()[:] = vv.vector()[:]
115             q_1['p'].vector()[:] = q_['p'].vector()[:]
116         elif ui == 'u2':
117             vv = interpolate_nonmatching_mesh(u0z,VV[ui])
118             q_[ui].vector()[:] = vv.vector()[:]
119             q_1[ui].vector()[:] = vv.vector()[:]
120             q_2[ui].vector()[:] = vv.vector()[:]
121         else:
122             vv = interpolate_nonmatching_mesh(u0x,VV[ui])
123             q_[ui].vector()[:] = vv.vector()[:]
124             q_1[ui].vector()[:] = vv.vector()[:]
125             q_2[ui].vector()[:] = vv.vector()[:]
126
127     def pre_solve_hook(V, u_, mesh, newfolder, MPI, mpi_comm_world,**NS_namespace):
128         if MPI.rank(mpi_comm_world())==0:
129             try: # only make dir if it's not there:
130                 makedirs(path.join(newfolder,"Stats"))
131             except:
132                 pass
133
134             info_red("pre_solve_hook1")
135             dim = 3
136             L = 40.
137             resolution = 200
138             xx = linspace(0,L,resolution)
139             x = resize(repeat(xx,dim),(resolution,dim))
140             x[:,1] /= np.max(x[:,1]) # probe from (0.5,1,20) -> (0.5,0.5,20) :
141             #x[:,1] = 1-x[:,1] # from top plane and down to middle
142             x[:,0] = 0.5
143             x[:,2] = 20
144             info_red("pre_solve_hook2")
145             stats = StatisticsProbes(x.flatten(), V, True)
146             mark = {"generic":0,
147                    "y_wall":1}
148             info_red("pre_solve_hook3")
149             class YWall(SubDomain):
150                 def inside(self,x,on_boundary):
151                     return (on_boundary and near(x[1],1.))
152             info_red("pre_solve_hook4")
153             y_wall_subdomains = MeshFunction("size_t",mesh,3-1)
154             info_red("pre_solve_hook5")
155             y_wall_subdomains.set_all(mark["generic"])
156             ywall = YWall()
157             info_red("pre_solve_hook6")
158             ywall.mark(y_wall_subdomains, mark["y_wall"])
159             info_red("pre_solve_hook7")
160             ds2 = Measure("ds")[y_wall_subdomains]
161             const_func = Function(FunctionSpace(mesh,'CG',1))
162             info_red("pre_solve_hook8")
163             const_array = const_func.vector().array()
164             info_red("pre_solve_hook9")
165             const_array = 1
166             info_red("pre_solve_hook10")
167             const_func.vector()[:] = const_array
168             info_red("pre_solve_hook11")
169             y_area2 = assemble(const_func*ds2(mark["y_wall"]))
170             info_blue("Area: {}".format(y_area2))
171             y_area = 1*40
172             n = FacetNormal(mesh)
173             return dict(stats=stats,x=x,np=np,h5l=h5l,ds2=ds2,mark=mark, y_area=y_area, n=n)

```

```

174
175 def magnitude(vec):
176     return sqrt(vec**2)
177
178
179 def temporal_hook(u_, p_, timestep, nu, dt, t, q_, mesh, VV, V, stats, update_statistics, newfolder, save_statistics, ←
    check_if_reset_statistics, folder, x, h5l, np, mark, ds2, y_area, MPI, mpi_comm_world, n, **NS_namespace):
180     info_red("tstep = {}".format(tstep))
181     if check_if_reset_statistics(folder):
182         info_red("Resetting statistics")
183         stats.clear()
184
185     if timestep % update_statistics == 0:
186         stats(q_['u0'], q_['u1'], q_['u2'])
187         try: # for the processor with the probe-line-info
188             info_blue(np.max(stats.array(1).flatten())) # "1" for present-v
189                                                         # "0" is integ. sum
190             info_red('max val for probe. (processor: {})'.format(MPI.rank(mpi_comm_world())))
191         except: # other processors pass
192             pass
193
194     if timestep % save_statistics == 0:
195         info_blue("saving stats...")
196         statsfolder = path.join(newfolder, "Stats")
197         h5f = h5l.File(statsfolder + "/dump_mean_{}.h5".format(tstep), "w")
198         g1 = h5f.create_group('u_profile')
199         sub_folder_name1 = "u_z"
200         sub_folder_name2 = "x_ax"
201         try:
202             info_blue("num of evals {}".format(stats.number_of_evaluations()))
203             info_red('from processor {}'.format(MPI.rank(mpi_comm_world())))
204             data_mat = np.array(stats.array().flatten()[2::9])
205             data_mat /= (1.*stats.number_of_evaluations())
206             g1.create_dataset(sub_folder_name1, data=data_mat) # we want the third entry: "w" (u.z)
207             g1.create_dataset(sub_folder_name2, data=x)
208             info_blue("saving done...")
209         except:
210             pass

```

## BIBLIOGRAPHY

- [1] M. Avila and B. Hof. *Nature of laminar-turbulence intermittency in shear flows*. Am. Phys. Soc. vol. 87:063012(5), 2013.
- [2] Kierkegaard J. B. *Theories of Active Matter with Applications to Endothelial Cell Motility*. Copenhagen University, 2014.
- [3] Pope S. B. *Turbulent flows*. Cambridge University Press, 2003.
- [4] A. J. Chorin. *Numerical solution of the Navier-Stokes equations*. Math. Comp. vol. 22:745–762., 1968.
- [5] O. Dauchot and E. Bertin. *Subcritical transition to turbulence: What we can learn from the physics of glasses*. Am. Phys. Soc. vol. 86:036312, 2012.
- [6] K. Avila D. Moxey A. de Lozar M. Avila D. Barkley and B. Hof. *The Onset of Turbulence in Pipe Flow*. SCIENCE, 2011.
- [7] B. M. DeBlois. *Linearizing the convection term in the Navier-Stokes equations*. Comput. Methods Appl. Mech. Engrg. 143:289-297, 1996.
- [8] M Nishi B Unsal F Durst and G Biswas. *Laminar-to-turbulent transition of pipe flows through puffs and slugs*. J. Fluid ;mrhc. vol. 614:425-446, 2008.
- [9] K. Erleben. *Finite difference methods - A short introduction at Department of Computer Science - University of Copenhagen*. Course slides, 2012.
- [10] D. A. Steinman et al. *Variability of Computational Fluid Dynamics Solutions for Pressure and Flow in a Giant Aneurysm*. J. Biomech. Eng. vol. 135, 2013.
- [11] M. Misztal K. Erleben K. et al. *Multiphase Flow of Immiscible Fluids on Unstructured Moving Meshes*. IEEE Transactions on Visualization and Computer Graphics 20(1):1-16, 2014.

- [12] B. Song D. Barkley B. Hof and M. Avila. *Speed and structure of turbulent fronts in pipe flow*. arXiv:1603.04077v1 - under review for J. Fluid Mech., 2016.
- [13] D. Barkley B. Song V. Mukund G. Lemoult M. Avilanand B. Hof. *The rise of fully turbulence flow*. NATURE vol. 536, 2015.
- [14] H-Y. Shih T-L Hsieh and N. Goldenfeld. *Ecological collapse and the emergence of travelling waves at the onset of shear turbulence*. NATURE PHYSICS vol. 12:245-248, 2016.
- [15] Erleben K. Misztal Marek K. and Bærentzen J. A. *Mathematical Foundation of the Optimization-based Fluid Animation Method*. Eurographics ACM SIGGRAPH Symposium on Computer Animation p.1-10, 2011.
- [16] Mortensen M. Valen-Sendstad K. *Oasis: A High-level/high-performance open source Navier-Stokes solver*. Computer Physics Communications vol 188:177-188, 2015.
- [17] Logg A. Mardal K.A. and Wells G. N. *The FEniCS Book; Automated Solution of Differential Equations by the Finite Element Method*. fenicsproject.org, 2011.
- [18] R. D. Moser J. Kim and N. N. Mansour. *Direct Numerical simulation of turbulent channel flow up to  $Re_\tau = 590$* . Physics of Fluids vol. 11(4):943-945., 1999.
- [19] H. P. Langtengen. *FEniCS Tutorial (v. 1.6.0)*. fenicsproject.org, 2016.
- [20] B Lautrup. *Physics of Continous Matter*. Taylor & Francis Group 2nd edition, 2011.
- [21] G Linga. *Project in Continuum Mechanics: Simulating Fluid Flow in Complex Geometries using FEniCS*. -, 2016.
- [22] Mortensen M. *Lecture notes on discretization*. <http://www.uio.no/studier/emner/matnat/math/MEK4470/h14/lecturenotes.pdf>, 2015.
- [23] Mortensen M. *Oasis - User Manual*. <https://github.com/mikaem/Oasis/blob/master/doc/usermanual.pdf>, 2015.
- [24] Mortensen M. *Oasis: GitHub-repository*. <https://github.com/mikaem/Oasis>, 2016.

- [25] Mortensen M. *Oasis: Wiki-page*. <https://github.com/mikaem/oasis/wiki>, 2016.
- [26] Mortensen M. *Oasis: Wiki-page for the fractional step algorithm*. <https://github.com/mikaem/Oasis/wiki/NSfracStep>, 2016.
- [27] P. Manneville. *On the transition to turbulence of wall-bounded flows in general, and plane Couette flow in particular*. European Journal of Mechanics vol. 49:345-362, 2015.
- [28] P. Manneville. *Transition to turbulence in wall-bounded flows: Where do we stand?* The Japan Society of Mechanical Engineering Review - DOI:10.1299/mer.15-00684 - March 30, 2016.
- [29] J. Reuther A. Jameson J. Farmer L. Martinelli and D. Saunders. *Aerodynamic Shape Optimization of Complex Aircraft Configurations via an Adjoint Formulation*. AIAA paper 96-0094; RIACS Technical Report 96.02; Research Institutet of Advanced Computer Science NASA Ames Research Center, 1996.
- [30] J. Mathiesen. *Week 12. CM-exercise*, 2012.
- [31] J. Mathiesen. *Short and random Comments*. CM-notes, 2016.
- [32] M. K. Misz. *FEMnotes: Introduction to the Finite Element Method*. -, 2016.
- [33] J. Kim P. Moin and R. D. Moser. *Turbulence statistics in fully developed channel flow at low Reynolds number*. J. Fluid Mech vol. 177(4):133-166., 1987.
- [34] A. S. Monin and A. M. Yaglom. *Statistical Fluid Mechanics - Mechanics of Turbulence vol. 1*. Dover Publications Inc., 1971.
- [35] D. Moxey and D. Barkley. *Distinct large-scale turbulent-laminar states in transitional pipe flow*. PNAS vol. 107:8091-8096, 2010.
- [36] N. A. Mortensen F. Okkels and H. Bruus. *Reexamination of Hagen-Poiseuille flow: Shape dependence of the hydraulic resistance in microchannels*. Physical review E vol. 71:057301-1-4, 2005.
- [37] Feynman R. P. *The Feynman Lectures on Physics Vol. 1*. the California Institute of Technology, 1963.

- [38] Y. Pomeau. *The transition to turbulence in parallel flows: A personal view*. C. R. Mecanique vol. 343:210-218, 2015.
- [39] Y. Pomeau. *The long and winding road*. NATURE PHYSICS vol. 12, 2016.
- [40] J. Kim R. D. Moser and M. Lee. *Direct Numerical simulation of turbulent channel flow up to  $Re_\tau \approx 5200$* . J. Fluid Mech. vol. 774:395-415, 2015.
- [41] O. Reynolds. *An Experimental Investigation of the Circumstances Which Determine Whether the Motion of Water Shall Be Direct or Sinuous, and of the Law of Resistance in Parallel Channels*. Phil. Trans. R. Soc. Lond. vol 174:935-982, 1883.
- [42] O. Reynolds. *On the Dynamical Theory of Incompressible Viscous Fluids and the Determination of the Criterion*. Phil. Trans. R. Soc. Lond. vol 186:123-164, 1894.
- [43] L. F. Richardson. *Weather prediction by numerical process*. The University pres - Cambridge, 1922.
- [44] Svenningsen M. S. *Simulation of cell dynamics using a FEM solution to the Navier-Stokes equations for incompressible fluids*. Copenhagen University, 2015.
- [45] M. Sano and K. Tamai. *A universal transition to turbulence in channel flow*. NATURE PHYSICS vol. 12:249-254, 2016.
- [46] J. C. Simo and F. Armero. *Unconditionally stability and long-term behavior of transient algorithms for the incompressible Navier-Stokes and Euler equations*. Computer Methods in Applied Mechanics and Engineering 111:111-154, 1994.
- [47] K. Valen-Sendstad D. A. Steinman. *Mind the gapGap: Impact of Computational Fluid Dynamics Solution Strategy on Prediction of Intracranial Aneurysm Hemodynamics and Rupture Status Indicators*. Am. J. Neuroradiol. vol 35:536-543, 2014.
- [48] M. K. Misztal A. Hernandez-Garcia R. Martin H. O. Sørensen and J. Mathiesen. *Detailed analysis of the lattice Boltzmann method on unstructured grids*. Journal of Computational Physics vol. 297:316-339., 2015.

- [49] R. Temam. *Sur L'approximation de la Solution des Equations de Navier-Stokes par la Méthode des Pas Fractionnaires(I) & (II)*. Archive for Rational Mechanics and Analysis vol. 32:135-153, 1969.
- [50] the PETSc Team. *Portable, Extensible Toolkit for Scientific Computation*. <https://www.mcs.anl.gov/petsc/>, 2016.
- [51] [https://www.uio.no/studier/emner/matnat/math/MEK4450/h11/undervisningsmateriale/modul-5/Pipeflow\\_intro.pdf](https://www.uio.no/studier/emner/matnat/math/MEK4450/h11/undervisningsmateriale/modul-5/Pipeflow_intro.pdf). *Flow in pipes*. University of Oslo; Course MEK4450 on Offshoreteknologi - teaching-material, 2011.
- [52] FEniCS Project (v. 1.6.0). *Automated, efficient solutions of differential equations*. <http://fenicsproject.org/>, 2016.
- [53] T. v. Kármán. *Nachrichten von der Gesellschaft der Wissenschaften zu Gottingen*. Physics of Fluids vol. 11(4):943-945., 1930.
- [54] Kwon Y. W. and Bang H. *The Finite Element Method using MATLAB*. CRC Press LLC, 1997.
- [55] A. M. Yaglom. *Hydrodynamic Instability and Transition to Turbulence*. SPRINGER, 2012.