Master Thesis

Advanced Techniques for Investigating Structures in Computational Fluid Dynamics

Author:
Rasmus Ellebæk Christiansen, s072162

Supervisors:
Professor Morten Brøns
Associate Professor Allan P. Engsig-Karup

DTU Compute - Department of Applied Mathematics and Computer Science.
29th July 2013

Technical University of Denmark

Abstract

The problems of incompressible fluid flow past a cylinder in free flow and past a cylinder near a moving wall in two dimensions are studied numerically. Here the velocity and pressure fields are obtained using a spectral element method based solver. The study is performed for Reynolds numbers in the low end of the periodic shedding regime and is mainly concerned with analysing vortex structures existing behind the cylinder, with main focus on the vortices surviving downstream. A vortex is defined uniquely as an extremum in vorticity and this definition is used to implement an algorithm for identifying and tracing vortices. A way of assuring that periodicity in the numerical solution has been reached using the tracing method is presented.

The Reynolds number, time and ratio of cylinder diameter to distance from the wall are identified as the only independent parameters of the problems. Dynamical systems theory is used to analyse changes in the vortex structures as the independent parameters are varied. Here, two types of bifurcations are identified and characterised using existing theory. Also, all different structures for vorticity extrema and saddle-points observed in the considered parameter range are identified and described.

The stabilizing effect of the wall is mapped. The effects of the wall and Reynolds number on the shedding frequency, creation point and pathway followed by the vortices are investigated. The decrease in magnitude of a vortex as it travels through the fluid is investigated and a nearly constant decrease rate is found far downstream.

A small scale investigation of collocation based uncertainty quantification is performed where the Reynolds number is taken to be the input parameter containing uncertainty. First, the cylinder in free flow is treated and the Strouhal number and pressure induced drag are considered as functions of the uncertain Reynolds number. Secondly, the cylinder near the moving wall is treated and the path followed by a vortex downstream and its magnitude as a function of the Reynolds number investigated.

Used theoretical concepts from dynamical systems theory, polynomial approximation theory, the spectral element method and uncertainty quantification are included. So are explanations of the software used and developed for the simulations and post processing readying the data for the final analysis.

Most of the methods used in this work have not been adapted by the industry (yet).

Abstrakt

To todimensionelle modelproblemer omhandlende en inkompressibel væske der flyder omkring en cylinder hhv i åbent flow og omkring en cylinder nær en væg i bevægelse studeres numerisk. Hastigheds- og trykfelterne bestemmes ved brug af en spektral element metode baseret løser. Studiet udføres for Reynoldstal i den lave ende af regimet for periodisk hvirveldannelse og beskæftiger sig hovedsageligt med analyse af de hvirvelstrukturer, der eksisterer bagved cylinderen med fokus på hvirvlerne der overlever nedstrøms af cylinderen. En hvirvel defineres unikt som et ekstremum i vorticitet, og denne definition bruges til at implementere en algoritme til at identificere og følge hvirvler. En måde til sikring af fuldstændig periodicitet i flowet ved brug af denne metode præsenteres.

Reynoldstallet, tiden og cylinder diameter over cylinder afstand til væggen identificeres som problemernes eneste uafhængige parametre. Dynamisk system teori bruges til at analysere ændringer i hvirvelstrukturerne når de uafhængige parametre varieres. Her identificeres og karakteriseres to typer bifurkationer ved brug af eksisterende teori. Yderligere identificeres og beskrives alle strukturer for vorticitets ekstrema og saddelpunkter, der er observeret under undersøgelserne.

Den stabiliserende effekt af væggen kortlægges. Effekten af væggen og Reynoldstallet på hvirveldannelsesfrekvensen, positionen af hvirvlernes dannelsespunkter, samt vejen disse følger gennem væsken, undersøges. Ydermere undersøges ændringerne i hvirvlernes intensitet når de rejser gennem væsken, og en næsten konstant formindskelsesrate findes når hvivlerne er rejst langt nedenstrøms.

En mindre undersøgelse af kolokationsbaseret usikkerhedskvantificering udføres. Her antages det, at Reynoldstallet er en usikker input parameter. Først behandles cylinderen i åbent flow, hvor Strouhal tallet og det trykinducerede drag betragtes som funktioner af det usikre Reynoldstal. Herefter betragtes cylinderen nær væggen i bevægelse. Vejen, som en hvirvel følger nedenstrøms, samt hvirvlens intensitet, betragtes som funktioner af det usikre Reynoldstal.

Anvendte teoretiske koncepter fra dynamisk system teori, polynomial approksimations teori, spektral element metoden og usikkerhedskvantificering er inkluderet. Det samme er tilfældet for en beskrivelse af softwaret anvendt til simulering og efterbehandling af data ved forberedelse til den endelige analyse.

De fleste metoder der anvendes i dette arbejde anvendes (endnu ikke) i industrien.

Contents

Introduction

This thesis presents and investigates advanced methods from different mathematical fields and provides the results of using these to analyse a Fluid Dynamics (FD) problem with focus on vortices appearing in the flow. The first field is Dynamical Systems Theory (DST) [1], [2], which is used to identify structures and bifurcations in vortex creation, annihilation and movement patterns. The second is the relatively new field of Uncertainty Quantification (UQ), with focus on the stochastic collocation approach, see e.g. [3] or [4]. All investigations are based on numerical solutions to the Navier-Stokes equation obtained using the Spectral Element Method (SEM) approach. The methods used and their mathematical foundation will be explained in some detail in chapter 2 while an interested reader is encouraged to consult the references for deeper insight. These methods have not (yet) been implemented for use by the industry.

Another aim of this written work is to provide enough information, examples and code to allow the reader to understand and take the steps necessary to reproduce the work presented[1]. This approach simultaneously allows the direct adaptation of the methods to analyse a range of other FD problems with relative ease.

As stated above all investigations are based on simulation results obtained using a SEM approach. The SEM solver used is the incompressible Navier-Stokes solver provided in the **Nektar++** framework [5]. All post processing is done using software written for this project as well as the open source data analysis tool **Paraview** [6].

The structure in this report is presented next. First, a short section on the notation and abbreviations used throughout the report is included for easy reference and to avoid any unnecessary confusion. Chapter 1 presents the model problems investigated, the underlying physics and a number of quantities of interest in the problem. Chapter 2 provides an overview of the theoretical foundation for the methods used from the areas of DST, SEM and UQ respectively along with an algorithm for tracing vorticity extrema and saddle points. Chapter 3 describes how the problem is discretized to allow for accurate numerical solutions.

Chapter 4 is dedicated to explaining the software developed for the project, the third party software used and the tool-chain setup for performing simulations. Chapter 5 contains a small investigation of the applicability of parallel computing for the solution process using the chosen solver framework.

Chapter 6 provides an overview of the simulations performed throughout the project, how data is visualized and an explanation of the application of the vorticity extrema tracing method.

Chapter 7 presents results of a validation process undergone to verify that results obtained using the **Nektar++** framework can be trusted. Chapter 8 presents the results and analysis for the DST and UQ based investigations. Lastly, Chapter 9 presents a short conclusion and outlook on what future work may be performed to extend the work presented here.

The appendices include the scripts and source code written for the project as well as compilation and usage instructions. It also contains a **.geo** and **.msh** file containing an example of the mesh used in the simulations and an example of XML-files to use as input files for the **Nektar++** solver.

[1]The reader should be aware that the reproduction of the work presented may be very time consuming due to the significant number of large scale simulations needed.

Notation

This section is included to avoid unnecessary confusion caused by the notation used in this work. Scalar quantities are denoted by italic letters, vector quantities are denoted by bold letters.

The following quantities have specific letters associated with them.

- **Time**: Time is denoted t.
- **Position**: Standard Cartesian coordinates are used unless otherwise stated and they are denoted as, $\mathbf{r} = (x, y)$.
- **Directional Unit Vectors**: Unit vectors denoting the x- and y-directions are denoted, $\hat{\mathbf{i}}$ and $\hat{\mathbf{j}}$ respectively.
- **Velocity**: Fluid velocity in Cartesian coordinates is denoted by $\mathbf{u} = (u, v)$.
- **Vorticity**: Vorticity is denoted ω and is defined as $\omega = \nabla \times \mathbf{u}$. In 2D this is a scalar quantity and in 3D it is itself a 3D vector field.
- **Pressure**: A pressure is denoted p.
- **Mass**: A mass is denoted m.
- **Volume**: A volume is denoted V.
- **Density**: Density is denoted by ρ.
- **Viscosity**: Viscosity is denoted by μ.
- **Gab Size**: The distance between a cylinder and a wall is denoted, G.
- **Cylinder Radius**: The radius of the cylinder is denoted, R.
- **Cylinder Diameter**: The diameter of the cylinder is denoted, D.
- **Far Field Velocity**: The velocity of the fluid far from the cylinder is denoted, U_∞.
- **Reference Pressure**: The reference pressure far from the cylinder is denoted, p_∞.
- **Reynolds Number**: The Reynolds number is denoted Re and is defined as $Re = \frac{\rho U_\infty D}{\mu}$.
- **Critical Reynolds Number 1**: Re_{crit} denotes the Reynolds number value which marks the transition from stationary to instationary flow.
- **Critical Reynolds Number 2**: Re_{crit_2} denotes the Reynolds number value where the two-dimensionality of the flow breaks down and three-dimensional effects appear.
- **Set of Parameters**: A set of parameters is denoted P.
- **Stress Tensor**: The stress tensor components are denoted $\tau_{i,j}$.
- **Normal to a Surface**: A surface normal vector is denoted $\hat{\mathbf{n}}$.
- **Tangent to a Surface**: A surface tangent vector is denoted $\hat{\mathbf{t}}$.

- **Drag force**: Drag on an object is denoted F_d.
- **Pressure Drag Coefficient**: This coefficient is denoted $C_{D_p} = \frac{p - p_\infty}{\frac{1}{2}\rho U_\infty^2}$.
- **Mean**: The statistical mean of a quantity, X is denoted, μ_X.
- **Variance**: The variance of a quantity, X is denoted, σ_X^2.
- **Uniform Distribution:** A uniform distribution on the interval $[a, b]$ is denoted, $\mathcal{U}(a, b)$.
- **Gaussian Distribution:** A normal distribution with mean a and standard deviation b is denoted, $\mathcal{N}(a, b)$.

The following is the standard notation for operators:

- **Differentiation**: Differentiation of a quantity, a, with respect to another quantity, b is denoted:
  - **Temporal Particle Derivative**: The Temporal Particle Derivative of a quantity Q is denoted: $\frac{DQ}{Dt}$.
  - **Partial Derivative**: Partial differentiation is denoted in one of the following ways: $\frac{\partial}{\partial b}a$, $\frac{\partial a}{\partial b}$ or a_b.
  - **Absolute Derivative**: The absolute derivative is denoted $\frac{d}{db}a$ or $\frac{da}{db}$.
  - **Spatial Derivative**: The derivative vector of all spatial directions is denoted $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})$
- **Integration**: A curve integral is denoted $\int \cdot \, d\kappa$ and a surface integral $\int \cdot \, d\mathbf{S}$ where $\cdot$ is a place holder for the integrand.
- **L_2-norm**: The standard L_2-norm is denoted $\| \cdot \|_{L_2}$ where $\cdot$ is a place holder for the element to be measured.
- **H_w^k-norm**: The weighted Sobolev-norm is denoted $\| \cdot \|_{H_w^k}$ where $\cdot$ is a place holder for the element to be measured.
- **Definition of function at specific values of independent variables:** If a function f of a set of parameters P is considered on the subset Ω of the full parameter domain this is denoted as $f(P)|_{P\in\Omega}$.

Abbreviations

Abbreviations will be used for names which are encountered often throughout the text. In general the full name will be used the first time a new concept is referenced with the abbreviation written after the name in parentheses. For easy reference the abbreviations are also listed below,

- Navier-Stokes: NS
- Spectral Element Method: SEM

- Dynamical System: DS
- Dynamical Systems Theory: DST
- Uncertainty Quantification: UQ
- Fluid Dynamics: FD
- Computational Fluid Dynamics: CFD
- Partial Differential Equation: PDE
- Boundary Condition: BC
- Degrees Of Freedom: DOF
- Probability Distribution Function: PDF
- generalized Polynomial Chaos: gPC
- Stochastic Collocation Method: SCM

Terminology

Here is a short list of terms used freely throughout the text.

- **Smoothness:** The term p-smoothness or p-smooth denotes a function which has p continuous derivatives. If a function is denoted as infinitely smooth all of its derivatives are continuous.
- **Pre Processing:** All preliminary work performed for a simulation to enable the solution using a numerical method.
- **Post Processing:** All work performed on the solution data obtained by solving the problem using a numerical method preparing it for analysis.
- **Vortex Shedding:** The term vortex shedding refers to the phenomenon that for a sufficiently high Reynolds number any object immersed in fluid will start generating vorticity at its edges and release (shed) the vortices down stream. The frequency with which this shedding occurs is called the shedding frequency.

1 The Problem and Underlying Physics

This chapter presents the problems investigated in this work, it outlines the purpose of the investigations, and it presents the underlying physics and the quantities of interest. Section 1.1 outlines the purpose of the project. Section 1.2 presents the model problems, their geometry and a set of parameters which defines them. Section 1.3 presents and explains the physics of the problems along with a set of model equations and appropriate boundary conditions. Lastly, section 1.4 describes a number of physical quantities of interest for the model problem.

1.1 The Purpose

This project has two main focuses. Firstly, it seeks to present and illustrate the application of a method for the investigation of vortex creation, annihilation and movement patterns for FD problems, by, among others, the utilization of results from Dynamical Systems Theory (DST). This is done based on the strict definition of a vortex as extrema in vorticity, see equation (1.20). The method allows accurate tracing of vortex creation, annihilation and movement patterns and relating both continuous and instantaneous changes in the patterns to the variation of the defining parameters of the FD problem under consideration.

Secondly, a preliminary investigation of the application and performance of Uncertainty Quantification (UQ) based methods to the model problem. Here, the idea is to determine functional dependences and statistics for quantities like pressure-drag and Strouhal number[2] as well as for a vorticis movement path through the domain as a function of an uncertain input parameter. The goal here is to investigate whether the method allows the dependency of the quantities of interest on an uncertain input parameter to be determined with good accuracy with only few model problem realizations. This is in contrast to the classic Monte Carlo method which requires a high number of realizations to obtain accurate statistics, which is infeasible for problems that require long simulation times due to its worst case $O(M^{-\frac{1}{2}})$ convergence rate.

The model problem of fluid flow past a cylinder presented in the following section is well investigated in fluid dynamics, both analytically, experimentally and numerically and thus is believed to be well understood. This means that numerical and experimental data for several quantities exists which can be used to verify the simulations performed for this project, see e.g. [7]. The complication of introducing a moving wall near the cylinder has been investigated by Huang and Sung in [8]. Over the course of these investigations different characteristics of the flow has been investigated for different Reynolds numbers[3], by Henderson in [7] and by Huang and Sung in [8] for different distances to the moving wall. At different parameter values of e.g. the period of shedding base pressure and pressure drag on the cylinder have been measured, calculated and the relationship with the parameters have been recorded[4].

There are many reasons for why investigating the vortex formation and movement patterns in flows is interesting from an engineering standpoint. A large number of real world problems may potentially be better understood if a better understanding of vortex formation and movement is obtained. Here follows a few examples. The movement of flags as a consequence

[2]The Strouhal number is a non-dimensionalised frequency and will be defined later.

[3]The Reynolds number will be defined in section 1.2.

[4]Definitions of all these quantities follow in section 1.4.

of vortex formation. The vortex formation around cables and pylons in bridge construction and the drag and vibrations caused by these vortices. The vortices formed around ship hulls as they travel through the oceans. The vortices formed behind race cars etc. If a better understanding of the influence of the vortices can be obtained it may be possible to using this knowledge to design structures which are more resilient and ships and cars which are capable of moving faster and more efficient due to reduced drag etc.

The use of numerical methods for obtaining this insight is very valuable as the numerical approach is a highly cost efficient way of performing experiments to obtain better understanding of physical processes. If the same knowledge was to be obtained through physical experiments, multiple and in some cases huge set-up's would have to be constructed which quickly becomes both very time consuming and very expensive.

1.2 The Model Problems

The main problem under consideration is that of an incompressible fluid flowing around an infinitely long cylinder oriented along the z-axis positioned near a moving wall modelled in two dimensions. The problem is investigated in the low Reynolds number regime ($Re \in [20, 300]$) after the initial transient flow has disappeared. In this context the Reynolds number is defined as $Re = \frac{\rho U_\infty D}{\mu}$ where ρ is the density of the fluid, U_∞ is the far field velocity of the fluid, D is the cylinder diameter and μ is the dynamic viscosity of the fluid. The fluid in the far field and the wall moves with the same velocity, $U_\infty = (u_\infty, 0)$ while the cylinder remains stationary. This leads to no slip boundary conditions on the cylinder of the form: $(u, v) = (0, 0)$ and likewise no slip on the wall of the form: $(u, v) = (u_\infty, 0)$. It also leads to a velocity condition at the outflow boundary of $\nabla(u, v) \cdot \mathbf{n} = 0$ and a pressure condition of $p = 0$. The boundary conditions are elaborated on in section 1.3. The second model problem is also a 2D model of a cylinder but this time in free flow. This problem is considered in the Reynolds number regime, $Re \in [100, 600]$.

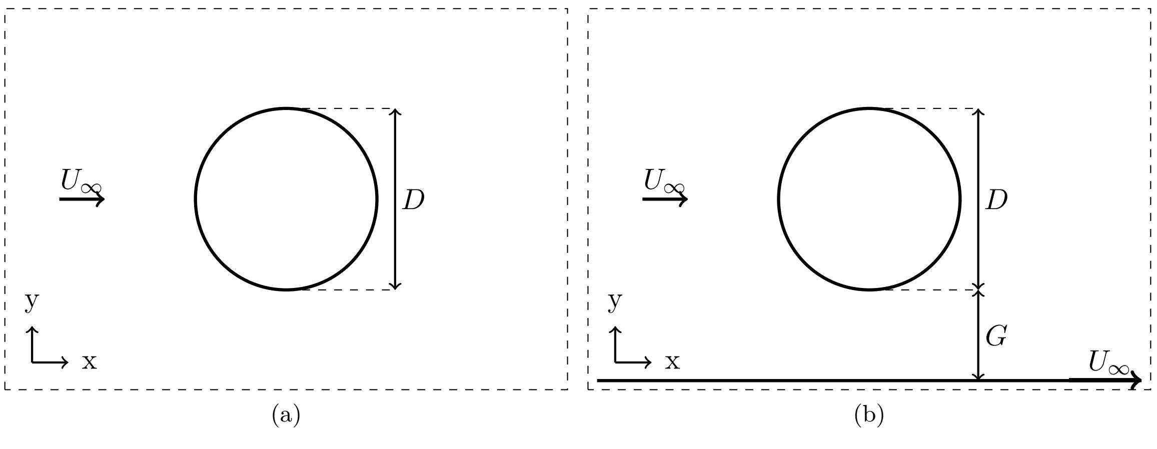


Figure 1.1: Schematics for 2D-domain containing a cross section of an infinitely long cylinder perpendicular to the flow. (a) Cylinder in free flow, (b) Cylinder near a wall.

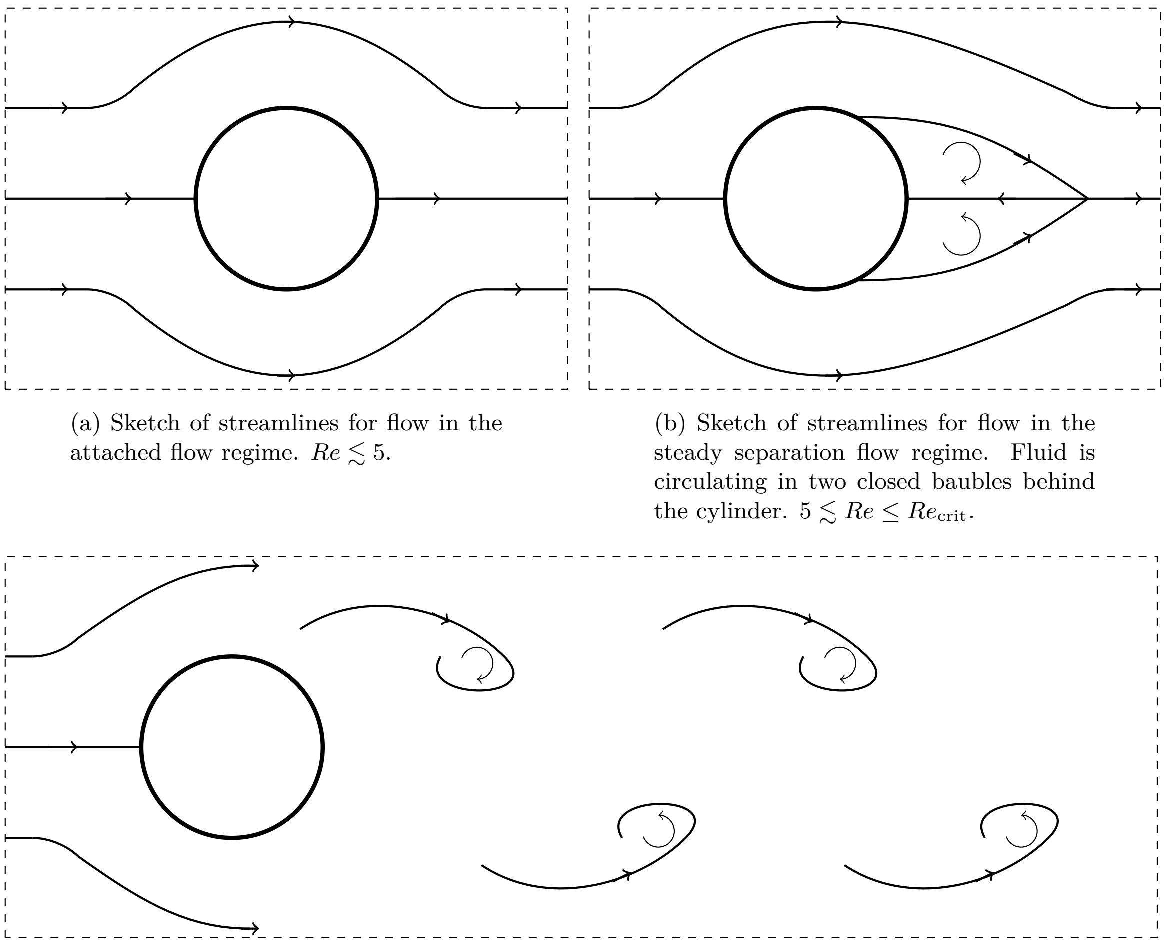
First, the cylinder in free flow is considered in order to validate numerical solutions and establish a reference for the flow structure. This model problem is also used for part of the UQ investigations. Then the main model problem is considered by introducing the wall and an investigation for a decreasing distance between cylinder and wall and varying Reynolds number Re is performed. Increasing the Reynolds number above a critical value changes the flow from stationary to instationary with a periodic behaviour and thus introduces a time parameter t. Below the critical value of the Reynolds number the flow is stationary and thus, based on the definition of a vortex provided in (1.20), no vortices exist. Schematic drawings of the two model problems are provided in figures 1.1a and 1.1b.

In figure 1.1 $U_\infty = (u_\infty, 0)$ is the inflow and far field velocity as well as the velocity of the wall. D is the cylinder diameter and G the size of the gap between the cylinder and wall. D and G are not of any interest as two separate parameters since the dynamics of the flow does not depend on them individually but only on their ratio. This fact is easily realised as D and G are the only geometric parameters in the system. Thus scaling D by a factor a simply corresponds to shrinking G by the same factor a. This means that the parameter of interest in the following is their ratio, chosen as $\frac{D}{G}$. The choice of $\frac{D}{G}$ opposed to $\frac{G}{D}$ is made based on the same considerations as those given by Huang and Sung in [8]. The considerations are that it is easier to illustrate small gap heights more clearly on a plot and that with this choice the case of a cylinder in free flow corresponds to $\frac{D}{G} = 0$ instead of $\frac{G}{D} \to \infty$.

From the present discussion three independent parameters for the problem have been identified. Two of the three are input parameters, i.e. Re and $\frac{D}{G}$. As the problem becomes instationary for increasing Reynolds number time emerges as the third parameter. In the following section where the physics and model equations are presented it is seen that these three parameters along with the boundary conditions are enough to describe the problem completely.

Different flow regimes: As mentioned the problem of the cylinder in free flow has been investigated thoroughly in several studies. For Reynolds numbers below the maximal value of interest in this work four different regimes for the flow as a function of Reynolds number have been characterised. In [9] Brøns et. al. identify, describe and analyse the first three of these regimes. The regimes may briefly be summarised as,

- **Attached Flow:** $Re \lesssim 5$. In this regime the flow is steady and attached to the cylinder. That is, the fluid passes over the cylinder and continues downstream without any circulation. Here, the fluid has a point of attachment on the upstream side of the cylinder and a point of detachment on the downstream side of the cylinder. This flow is sketched in figure 1.2a.
- **Steady-Seperation:** $5 \lesssim Re \leq Re_{\text{crit}}$. In this regime the flow is still steady, however a bauble of recirculation has appeared behind the cylinder. That is, a steady recirculation of fluid is confined to the backside of the cylinder. Here, the fluid has a point of attachment on the upstream side of the cylinder and two points of detachment and a point of attachment on the downstream side of the cylinder. This flow is sketched in figure 1.2b.
- **Periodic Shedding:** $Re > Re_{\text{crit}}$. In this regime the flow is no longer steady but instead it exhibits a periodic behaviour. This periodic behaviour consists of a periodic shedding of vortices from the backside of the cylinder. The vortices travel downstream creating a so-called vortex train behind the cylinder. The position of the points of detachment on the downstream side of the cylinder fluctuate in time and the point of attachment no longer exists. This flow is sketched in figure 1.2c.
- **Breakdown of the Two Dimensionality in the Flow:** For the cylinder in free flow, Henderson [10] identifies $Re_{\text{crit}_2} \approx 190$ to be the Reynolds number at which the two-dimensionality of the flow breaks down. Beyond Re_{crit_2} the flow exhibits three-dimensional behaviour and the 2D simulations are no longer enough to capture the physics completely.



(a) Sketch of streamlines for flow in the attached flow regime. $Re \lesssim 5$.

(b) Sketch of streamlines for flow in the steady separation flow regime. Fluid is circulating in two closed baubles behind the cylinder. $5 \lesssim Re \leq Re_{\text{crit}}$.

(c) Sketch of periodic shedding where vortices are shed from the downstream side of the cylinder. $Re > Re_{\text{crit}}$

Figure 1.2: Sketches of different regimes of fluid flow in the Reynolds number range, $Re \in]0, 600[$ for a cylinder in free flow.

There is no reason to expect that the different regimes should not exist when introducing a moving wall near the cylinder, although the *Re* values at which the flow transitions from one regime to another may be expected to vary. This variation has been investigated for the critical value for periodic shedding, when introducing and moving the wall closer to the cylinder.

An important note is that from the definition of a vortex used in this work, see equation (1.20), no vortices exist in the recirculating flow in the steady-separation regime. Vortices first appear at the transition from the stationary to periodic regimes. As the vortices are the objects of interest in this work the main regime of interest is the periodic shedding regime where a vortex train has appeared behind the cylinder.

Another note is that the range of Reynolds numbers investigated in this work goes beyond the breakdown of the two-dimensionality of the flow. This means that the presented data will not mirror a true three-dimensional flow exactly. However, it is estimated that the maximal investigated Reynolds number is close enough to the two-dimensional breakdown for results to still be a reasonable approximation to the behavior of the actual three-dimensional flow.

1.3 Physics and Model Equations

The physics believed to govern all fluid flow on the length scale of interest in this project is classical Newtonian mechanics. That is, all length scales are assumed to be large enough to be able to consider the fluid as a continuum. This is in contrast to length scales where the individual molecules making up the fluid must be considered. A difference from classical mechanics is that for fluids one usually considers a velocity-field formulation instead of considering individual particles as is the usual approach in mechanics. This leads to a new form for the particle temporal derivative of a quantity, Q, for the fluid given (in 2D) as,

$$\frac{DQ}{Dt} = \frac{\partial Q}{\partial t} + u\frac{\partial Q}{\partial x} + v\frac{\partial Q}{\partial y} = \frac{\partial Q}{\partial t} + (\mathbf{u}\cdot\nabla)Q. \tag{1.1}$$

A full description and derivation of the difference between the particle and field approach may be found in [11, section 1-3].

1.3.1 General Navier-Stokes Equation

By applying Newtons second law of motion, $\sum_i F_i = m\mathbf{a} = m\frac{D\mathbf{u}}{Dt}$, to a "box of fluid" at an arbitrary position in an inertial frame of reference and dividing by the volume of the box one obtains a general form of the Navier-Stokes (NS) equations (1.2), which govern the motion of the fluid [11, section 2-4]:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \nabla\cdot\boldsymbol{\tau} + \mathbf{f}. \tag{1.2}$$

Here, $\mathbf{u}$ is the flow velocity, ρ the fluid density, p the pressure, $\boldsymbol{\tau}$ a stress tensor and $\mathbf{f}$ represents external body forces acting on the fluid.

Each term in the NS-equation may be interpreted physically as follows.

The term: $\frac{\partial \mathbf{u}}{\partial t}$. This term quite simply describes the change in the velocity of the fluid caused directly by a change in time. This term is clearly non-zero for all instationary problems and zero for stationary problems.

The term: $\mathbf{u}\cdot\nabla\mathbf{u}$. This term is called the convective acceleration term or the non-linear advection term. This term represents the fact that the fluid may be accelerated due to its position in the domain independently of time. That is, even for stationary problems ($\frac{\partial \mathbf{u}}{\partial t} = 0$) a fluid may change velocity throughout the domain. As an example of this one may consider a steady flow of incompressible liquid from one reservoir to another through a diverging channel. See figure 1.3 for an illustration of the channel.

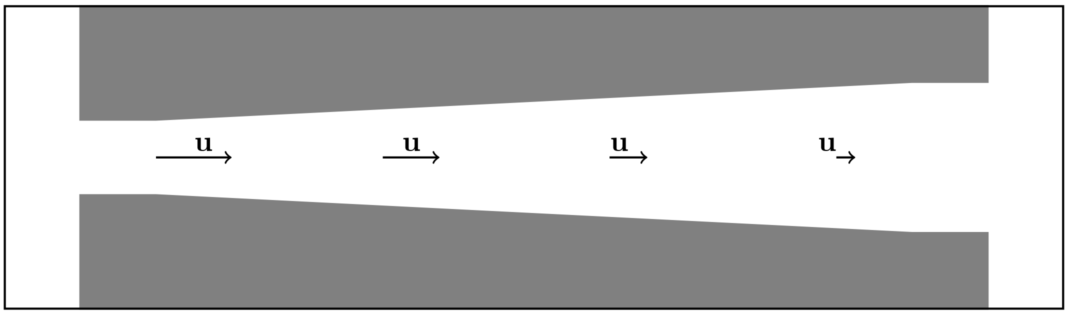


Figure 1.3: Illustration of a diverging open ended channel in which a steady flow of fluid will be fastest in the narrow end and slowest in the wide end.

This phenomenon is relatively easily understood if one considers mass conservation and continuity. Only the fluid which has passed through the narrow part of the channel is available to fill the channel down stream where the channel is wider. As the fluid must remain continuous the only choice available is to slow down as it moves down the channel.

The terms: $-\nabla p + \nabla \cdot \boldsymbol{\tau}$. Both terms represent effects of stresses in the fluid. ∇p is the gradient of the pressure and stems from the normal stresses in the fluid. $\nabla \cdot \boldsymbol{\tau}$ stems from the anisotropic part of the stresses and describes viscous forces. This can be understood as the friction forces between layers of fluid pulling on each other as they pass one another. In order to simplify the NS-equations for the solution of engineering problems a series of assumptions can be made for $\boldsymbol{\tau}$ but going into depth with this is outside the scope of this project. A simple assumption is that the fluid is incompressible which reduces $\boldsymbol{\tau}$ as described in the following.

Further explanations of the terms and assumptions which may be made can be found in [11, chapter 2].

The term: f. This term represents all exterior forcing on the fluid. Normally, this always includes gravity and may also include forcings on the system under consideration from e.g. electrical fields.

1.3.2 Incompressible Navier-Stokes Equation

For the problems of interest in this project the fluid under consideration is assumed incompressible, temperature independent and it is assumed that the viscosity is constant throughout the fluid. These assumptions simplify the NS-equation to the form presented in equation 1.3,

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{f}, \tag{1.3}$$

It is easily seen that the change from the general NS-equation is the reduction of $\nabla \cdot \boldsymbol{\tau}$ to $\mu \nabla^2 \mathbf{u}$ which is denoted the viscous diffusion term. This term may be interpreted as modelling the diffusion of momentum which occurs for an incompressible viscous fluid.

The form of the NS-equation presented in (1.3) may be used directly as a component in solving an incompressible fluid problem. It is however beneficial to non-dimensionalize the NS-equation as this will reduce the parameters in the equation to a single parameter denoted the Reynolds number.

The first step is to introduce a mean flow velocity U and characteristic length L. From these a set of dimensionless functions and operators are then introduced as,

- **Non-dimensional Velocity:** $\check{\mathbf{u}} = \frac{\mathbf{u}}{U}$.
- **Non-dimensional Pressure:** $\check{p} = \frac{p}{\rho U^2}$.
- **Non-dimensional Body Force:** $\check{\mathbf{f}} = \frac{L}{\rho U^2}\mathbf{f}$.
- **New Time Derivative:** $\frac{\partial}{\partial \check{t}} = \frac{L}{U}\frac{\partial}{\partial t}$.
- **New Spatial Derivative:** $\check{\nabla} = L\nabla$.

By multiplying (1.3) which is an equation in terms of body forces, with the quantity $\frac{L}{\rho U^2}$, which has the unit of $\left[\frac{\mathrm{m}^3}{\mathrm{N}}\right]$, one obtains the dimensionless equation,

$$\frac{L}{U^2}\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\frac{L}{\rho U^2}\nabla p + \frac{\mu L}{\rho U^2}\nabla^2\mathbf{u} + \frac{L}{\rho U^2}\mathbf{f}, \Leftrightarrow \tag{1.4}$$

$$\frac{L}{U}\frac{\partial}{\partial t}\frac{\mathbf{u}}{U} + \frac{\mathbf{u}}{U}\cdot\nabla\frac{\mathbf{u}}{U} = -L\nabla\frac{p}{\rho U^2} + \frac{\mu}{\rho U L}L^2\nabla^2\frac{\mathbf{u}}{U} + \frac{L}{\rho U^2}\mathbf{f}. \tag{1.5}$$

By substituting in the dimensionless functions and operators one obtains,

$$\frac{\partial}{\partial \check{t}}\check{\mathbf{u}} + \check{\mathbf{u}}\cdot\nabla\check{\mathbf{u}} = -\check{\nabla}\check{p} + \frac{\mu}{\rho U L}\check{\nabla}^2\check{\mathbf{u}} + \check{\mathbf{f}}. \tag{1.6}$$

Introducing the Reynolds number as, $Re = \frac{\rho U L}{\mu}$ and dropping the check marks to simplify the notation one arrives at the final non-dimensional NS-equation,

$$\frac{\partial}{\partial t}\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2\mathbf{u} + \mathbf{f}. \tag{1.7}$$

Note that $L = D$ for the model problems considered here.

1.3.3 Mass Conservation

Another requirement for a fluid problem is that of mass conservation. This principle simply states that mass cannot appear out of nothing and likewise can not disappear into nothing. Thus, we must require,

$$m = \rho V = \text{constant}, \tag{1.8}$$

where m is the mass and V is the volume. For an incompressible fluid any "box of fluid" must have a constant density in space and time. From this consideration and by taking the particle derivative of equation 1.8 one gets the condition,

$$\frac{Dm}{Dt} = \frac{D\rho}{Dt}V + \rho\frac{DV}{Dt} = 0 \Rightarrow \nabla \cdot \mathbf{u} = 0. \tag{1.9}$$

See [11, section 2-3] for a full derivation of this equation, i.e. for the missing steps hidden in the $\Rightarrow$.

(1.9) is easily non-dimensionalised by introducing,

- **Non-dimensional Velocity:** $\check{\mathbf{u}} = \frac{\mathbf{u}}{U}$.
- **New Spatial Derivative:** $\check{\nabla} = L\nabla$.

Multiplying (1.9) by $\frac{L}{U}$ and rewriting to obtain,

$$L\nabla \cdot \frac{\mathbf{u}}{U} = \check{\nabla} \cdot \check{\mathbf{u}} = 0. \tag{1.10}$$

Dropping the check mark we get the non-dimensional equivalent of (1.9), which happens to have the exact same form.

1.3.4 Model Equations

We have now considered all the components needed to formulate a model for the problem.

The two main model equations are thus the non-dimensional NS- and continuity-equations:

$$\frac{\partial}{\partial t}\mathbf{u} + \mathbf{u} \cdot \nabla\mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2\mathbf{u} + \mathbf{f}, \tag{1.11}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{1.12}$$

Additionally, we need to formulate boundary and initial conditions for the problem. Considering the problem geometry described in section 1.2 a set of boundary conditions may be formulated.

Cylinder Boundary Condition: The boundary constituted by the cylinder surface uses the usual FD no-slip condition. For the stationary cylinder this may be formulated as:

$$\mathbf{u}|_{(x,y)\in\mathbf{CS}} = (0, 0), \tag{1.13}$$

where **CS** is the set of points which lie on the surface of the cylinder.

Wall Boundary: The boundary constituted by the moving wall also uses the usual FD no-slip condition, This may be formulated as:

$$\mathbf{u}|_{(x,y)\in\mathbf{WS}} = (u_\infty, 0), \tag{1.14}$$

where **WS** is the set of points which lie on the wall.

Far Field Boundaries: As the method of choice for solving the problem is simulation, and as it is impossible to simulate an infinite domain, one needs to approximate the infinite half plane above the wall by a finite box. The concerns in the reader's mind for which this approximation gives rise will be addressed in section 3. The "box" approximation gives rise to two different far field BC's. The first is an inflow condition given by:

$$\mathbf{u}|_{(x,y)\in\mathbf{IF}} = (u_\infty, 0), \quad \nabla p|_{(x,y)\in\mathbf{IF}} = 0. \tag{1.15}$$

where **IF** is the set of points which lie on the inflow boundaries. These BCs represent a completely undisturbed flow as one would expect far away from the cylinder[5].

The second is an outflow condition given by;

$$\nabla\mathbf{u}|_{(x,y)\in\mathbf{OF}} \cdot \mathbf{n} = (0,0), \quad p|_{(x,y)\in\mathbf{OF}} = p_\infty = 0. \tag{1.16}$$

where **OF** is the set of points which lie on the outflow boundary, **n** is the outward pointing normal vector and p_∞ is a reference pressure which is set equal to zero. This BC simulates that the fluid simply flows freely out of the domain at the outflow boundary.

An illustration of the problem domain with the different BC's highlighted is provided in figure 1.4.

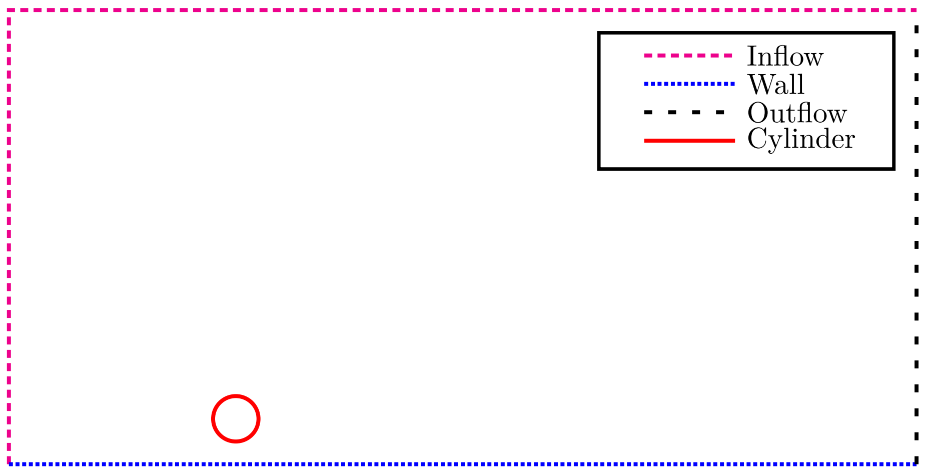


Figure 1.4: Illustration of the location of the different boundary conditions for the problem. (**CS**): Cylinder surface (full line), (**WS**): Wall surface (densely dotted), (**IF**): Inflow (densely dashed), **(OF): Outflow (loosely dashed)**.

[5]Due to numerical performance the pressure condition stated here is not used in the solver from the **Nektar++** framework. Instead a higher order pressure condition derived to provide better numerical accuracy is used. Details of this condition and its derivation may be found in [18] and/or [16, section 8.3]

Initial Condition: The initial condition is chosen to be that of a free flow without the cylinder, i.e.

$$\mathbf{u}|_{t=0} = (u_\infty, 0), \quad p|_{t=0} = p_\infty = 0. \tag{1.17}$$

Since the investigation is not concerned with the initial transient solution, any reasonable choice of velocity and pressure fields as initial condition could be used as long as it is chosen to be compatible with the numerical method used.

1.4 Physical Quantities

For general FD problems many different quantities and characteristics of the flow may be of interest depending on the intended application. Additionally, a number of auxiliary quantities are important for analytical purposes in order to obtain the desired quantities and characteristics. This section outlines the quantities and characteristics of interest to this work.

The quantities of interest here are the pressure induced drag on the cylinder, the non-dimensionalized frequency of vortex shedding called the Strouhal number, the non-dimensionalized pressure at the cylinder surface called the pressure coefficient and the vortex movement pattern behind the cylinder. A few examples of areas where these quantities are important follow here.

In aerospace, car, boat and wind turbine engineering the drag on an object subjected to a fluid flow is of high interest as the minimization or maximization of the quantities are important for the performance of the object. The pressure coefficient at an object's surface and Strouhal number may likewise be of interest in these areas of engineering. In fluid mixing, understanding the flow patterns has interest as this may determine the most efficient way to mix some reactant with a fluid to create a product. Understanding the flow patterns behind an object can also be of importance if other objects are to be placed in the wake of the first object.

An crucial quantity for the analysis of vortex patterns and structures which is of interest in this work is the vorticity as it may be used to uniquely define a vortex.

Vorticity: This quantity is defined in $\mathbb{R}^3$ as the curl of the velocity vector,

$$\boldsymbol{\omega} = \nabla \times \mathbf{u}, \tag{1.18}$$

which in 2D translates to a scalar quantity,

$$\omega = \frac{\partial}{\partial x} v - \frac{\partial}{\partial y} u, \tag{1.19}$$

since the x- and y-components of (1.18) are zero in 2D.

The vorticity at a given point in the fluid may be understood as a measure for how much of a rotating motion the fluid experiences near this point. A positive vorticity corresponds to counter-clockwise rotation of the fluid while a negative vorticity corresponds to a clockwise rotation.

The introduction of the vorticity allows for a strict definition of a vortex which is used in this work to identify individual vortices uniquely.

A Vortex: In order to perform a rigorous investigation of the creation, annihilation and movement of vortices in a flow one must first define a vortex. In this work the center of a vortex is understood as an extremum in vorticity. This allows vortices to be defined uniquely by their centres from the extrema in the vorticity which are given by the points (x_i, y_i) in the flow where,

$$\nabla\omega = \mathbf{0} \;\Leftrightarrow\; \omega_x = 0 \;\wedge\; \omega_y = 0, \tag{1.20}$$

and where the Hessian of the vorticity is either positive or negative definite corresponding to minima and maxima, respectively.

From this definition it is possible to locate and count the number of vortices in a flow at a given time. With the capability to uniquely identify vortices it also becomes possible to trace the movement, creation and annihilation of vortices over time.

Strouhal Number: The Stouhal number St may be thought of as a dimensionless frequency for the shedding of vortices from an object immersed in fluid. It is given by,

$$St = \frac{L}{U_\infty} f, \tag{1.21}$$

where f is the shedding frequency, L is a characteristic length scale in the problem under consideration and U_∞ is the far field velocity of the fluid. In this work $L = D$.

Pressure Coefficient: The pressure coefficient, C_p may be thought of as a dimensionless measure for the pressure relative to the far field pressure and is defined by,

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho U_\infty^2}, \tag{1.22}$$

where p_∞ is the far field pressure, U_∞ is the far field velocity and ρ is the density of the fluid.

For the cylinder problem the base pressure coefficient C_{bp} is defined as C_p at the cylinder surface 180 degrees from the front of the cylinder.

Pressure Drag Coefficient: The drag force, F_d, on an object is defined as the sum of the forces acting on the object parallel to the direction of movement for the objects through a fluid. That is, it is the force which works to slow down the object as it moves through the fluid. In an incompressible viscous fluid the drag is composed of a force due to the pressure, F_p, of the fluid on the object and a shear force, F_s, from the layers of fluid sliding over the surface of the object. Choosing the direction of positive drag in the direction opposite the movement of the object we get,

$$F_d = F_{p_d} + F_{s_d} = -\left(\int p \; \hat{\mathbf{n}} \cdot \hat{\mathbf{i}} \; d\mathbf{S} + \int \tau \; \hat{\mathbf{t}} \cdot \hat{\mathbf{i}} \; d\mathbf{S} \right), \tag{1.23}$$

where τ denotes the stress tensor component along the surface where $\hat{\mathbf{i}}$ is the unit vector in the direction of movement of the object. $\hat{\mathbf{t}}$ and $\hat{\mathbf{n}}$ are the normal and tangential vectors for the cylinder surface. In this work only the pressure-drag force is considered. From F_{p_d} a non-dimensional drag coefficient C_{D_p} is defined as,

$$C_{D_p} = \frac{F_{p_d}}{\frac{1}{2}\rho U_\infty^2}. \tag{1.24}$$

2 Theory

This chapter provides an overview of the most important theoretical concepts from the different areas of mathematics used throughout the thesis. Section 2.1 outlines results and concepts used from Dynamical Systems Theory (DST) and presents an algorithm used to identify critical points for the vorticity. Section 2.2 states definitions and results for orthogonal polynomials and approximation theory which provides the basis for UQ and SEM. Section 2.3 provides a brief overview of the strengths and weaknesses of the Spectral Element Method (SEM) used in the numerical simulations compared to the classical Finite Volume/Element Metohds (FVM/FEM). Here, an illustration of the main steps in the application of the SEM to a model problem is also provided to give the reader a basic understanding of how the model problems are solved numerically. An illustration of convergence results obtained using the method is also provided. Lastly, section 2.4 presents used concepts and results from Uncertainty Quantification (UQ).

2.1 Dynamical Systems Theory

The overarching aim of DST is to understand the structure of a dynamical system and how it changes depending on the parameters of the system. A two-dimensional dynamical system may be written generally as a system of first order ODE's as,

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} P(x,y,c) \\ Q(x,y,c) \end{pmatrix}, \tag{2.1}$$

where $\dot{x}$ and $\dot{y}$ may be interpreted as a velocity given by the vector field $(P(x,y,c), Q(x,y,c))$ and $c \in \mathbb{R}^k$ is a k-dimensional vector of parameters. For systems of this form fixpoints, also known as critical points, are of special interest since they mark points where the behaviour of the system may change with small variations in the parameters c.

Critical points are defined as points in space where the right hand side of (2.1) *is zero. Hence points where the velocity in the system is zero.*

In [2] Brøns presents theorems which state that regular points, i.e. points which are not critical points are structurally stable, meaning that small changes in the parameters c cannot change the structure of the dynamical system. In order to determine the type of a critical point the eigenvalues of the Jacobian of the governing dynamical system is considered[6].

Consider now a 2D flow problem where $\mathbf{u}(\mathbf{x}, t)$ is a known velocity field. Consider the time t as a parameter and look at the system at $t = t_0$. This gives the dynamical system,

$$\dot{\mathbf{x}} = \mathbf{u}(\mathbf{x}, t_0). \tag{2.2}$$

The solutions to this system are called streamlines and describe the integral curves of the velocity field at the instant in time t_0. That is, curves which a massless tracer particle would follow through the flow at a given instance in time. The critical points of this dynamical

[6]A well developed theory for identifying the critical points, their type, and the ways they may be created and/or annihilated when the parameters c are varied, exists, see e.g. [1].

system are in fluid dynamic terms denoted stagnation points, i.e. points where the fluid is at rest. For incompressible flow problems we may define a stream function Ψ, which fulfil $u = \frac{\partial \Psi}{\partial y}, v = -\frac{\partial \Psi}{\partial x}$, [2]. The system (2.2) now takes the form,

$$\dot{\mathbf{x}} = \left(\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t} \right) = \left(\frac{\partial \Psi}{\partial y}, \frac{-\partial \Psi}{\partial x} \right). \tag{2.3}$$

The definition of Ψ means that it is constant along streamlines for (2.2) as may easily be proven,

$$\frac{d\Psi}{dt} = \frac{\partial \Psi}{\partial x}\frac{dx}{dt} + \frac{\partial \Psi}{\partial x}\frac{dy}{dt} = \frac{\partial \Psi}{\partial x}\frac{\partial \Psi}{\partial y} - \frac{\partial \Psi}{\partial y}\frac{\partial \Psi}{\partial x} = 0, \tag{2.4}$$

that is, the streamlines for (2.2) are level curves for Ψ. This fact means that critical points for (2.2) are stationary points for Ψ since these are simply special cases of level curves.

In [2] it is also stated that for any regular critical point of (2.3), i.e. points where the Jacobian of the system is regular, are structurally stable. This means that in order to locate structural changes in the flow the degenerate critical points where the Jacobian is singular must be considered.

Analysis using DST: In [2], Brøns derives a number of theoretical results for the dynamical system (2.3) describing the streamlines of the two-dimensional velocity field (u,v) of incompressible fluid flow using the stream function Ψ. As will be shown below an equivalent system may be formulated for the vorticity ω. This means that all the results developed by Brøns may be used to analyse the system governed by ω instead of Ψ. That is, the theory may be used to analyse the creation, annihilation and movement patterns of vortices in a 2D flow problem.

As stated earlier a central goal of the present work is to implement and demonstrate a method for rigorously investigating the creation, annihilation and dynamics of vortices[7]. The first step on this path is to obtain the velocity field for the fluid and from this calculate the vorticity. This is done by solving the governing equations numerically, which will be discussed in detail later. In the following the velocity field and thus the vorticity is assumed known.

It turns out that the dynamics of the stationary points for vorticity and thus of vortices may be understood using DST. This is done by considering time as a parameter instead of a continuous variable and investigating the critical points of a family of dynamical systems of the form,

$$\begin{pmatrix} \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial s} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \omega_y(x, y, t_i, c) \\ -\omega_x(x, y, t_i, c) \end{pmatrix}, \quad i \in \{1, ..., N\}, \; c \in \mathbb{R}^k, \; k \in \mathbb{N}. \tag{2.5}$$

Here (x, y) are spatial coordinates, ω the vorticity and $c \in \mathbb{R}^k$ are a vector of parameters. s is an artificial time introduced to consider how the flow would move when the real time is

[7]Remember that a vortex has been defined from a vorticity extremum, i.e. points in space and time where $\omega_x(x, y, t) = \omega_y(x, y, t) = 0$ and where the Hessian is positive or negative definite, see section 1.4.

frozen. Notice that we have frozen time and that the N separate time steps are considered as individual dynamical systems. Consider now the vorticity field at one of these snapshots in time, $\omega(x, y, t_i, c)$. If this quantity is differentiated with respect to s, utilizing the chain rule one gets,

$$\frac{d}{ds}\omega(x, y, t_i) = \frac{\partial \omega}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial \omega}{\partial y}\frac{\partial y}{\partial s} = \omega_x\omega_y - \omega_y\omega_x = 0, \tag{2.6}$$

where we have used from (2.5) that $(\frac{\partial x}{\partial s}, \frac{\partial y}{\partial s}) = (\omega_y, -\omega_x)$. This is in complete analogy to (2.4).

Following the analogy, it is seen from (2.6) that the streamlines given by the solutions to (2.5) are level curves for $\omega(x, y, t_i, c)$ with respect to s and thus the extrema of $\omega(x, y, t_i, c)$ are special cases of level curves which have $\omega_x = \omega_y = 0$. From this it follows that a critical point of the dynamical system (2.5) corresponds to stationary points for ω. These may either be extrema, saddle points or degenerate points. As stated above a degenerate point is a point where the Jacobian matrix for the system (2.5) is singular and it is only at degenerate critical points that structural changes in (2.5) may occur, see [2, Theorem 3].

This means that if one can identify all stationary points of the vorticity one finds all the critical points of the system (2.5) and thus also all vortices present in the flow. The critical points and any possible structural changes they may undergo, corresponding to the creation or annihilation of vortices, can then be analysed using the machinery of DST. This will be elaborated on below as the saddle-center bifurcation is treated.

Comment on critical points for (2.5) **and extrema/saddles for vorticity:** By comparing the Hessian matrix for the vorticity and the Jacobian matrix for the system (2.5) it is immediately observed that extrema for the vorticity corresponds to centres for (2.5) and saddles for the vorticity corresponds to saddle points for (2.5). Hence identifying a center for (2.5) corresponds to identifying a vortex in the flow.

Nullclines: Consider (2.1). A nullcline is defined as a curve along which either $\dot{x}$ or $\dot{y}$ is zero. Thus a nullcline corresponds to the zero contours of either $\dot{x}$ or $\dot{y}$. In 2D one may use nullclines to identify critical points by looking for intercepts between the contours $\dot{x} = 0$ and $\dot{y} = 0$.

The technique of using the intercepts between nullclines is used for identifications of critical points for ω. This means that by identifying $\omega_x = 0$ one then only has to solve a one dimensional problem along the curves $\omega_x = 0$ to identify $\omega_y|_{(\omega_x=0)} = 0$ and thereby the critical points.

Parameter dependence: Any spatial physical instationary problem has by definition some dependence on space and time. Besides this dependence the problem most likely also depends on a range of parameters. These dependencies may be on boundary and initial conditions, the domain shape and for a fluid problem the Reynolds number. The task now becomes to identify at what parameter values the structure of the system (2.1) changes and what changes occur. These changes in structure as a function of parameter variations are denoted bifurcations.

2.1.1 Bifurcations

Meiss [1] offers a definition of a bifurcation as,

Bifurcation: a qualitative change in dynamics occurring upon a small change in a parameter.

Many different types of changes in dynamics can occur in (2.1) as a consequence of varying system parameters, see [1, Chapter 8] for a treatment of some of these. Some bifurcations occur as a single parameter is varied while others occur only as a consequence of varying multiple parameters at once. The minimum number of parameters which must be varied for a given bifurcation to occur is denoted the *Codimension* of the bifurcation.

For the systems investigated in the present work the codimension 1 saddle-center bifurcation was the only bifurcation observed away from domain boundaries in all but a single case[8]. In the single exceptional case a co-dimension 2 bifurcation was identified. Due to its frequent occurrence the saddle-center bifurcation is covered in detail here, while the co-dimension 2 bifurcation is treated specifically in chapter 8 when it is encountered.

Saddle-Center Bifurcation: This bifurcation corresponds to the creation or annihilation of two critical points as a single parameter is varied. One of the critical points is a saddle point and the other is a center.

In order to illustrate such a bifurcation consider a Taylor expansion of the vorticity given by,

$$\omega = \sum_{n,m=0}^{\infty} a_{nm} x^n y^m, \tag{2.7}$$

In [2] Brøns shows that if one applies transformations to the dynamical system in consideration such that the degenerate critical point undergoing a bifurcation is centred at the origin and the coordinate axis are twisted and stretched appropriately the following theorem holds[9],

Theorem 1: *Let $a_{10}, a_{01}, a_{11}, a_{20}$ of (2.7) and $\tilde{a}_{n0}$ for $n < N-1$ be small parameters, and assume that non-degeneracy conditions $a_{02} \neq 0$ and $\tilde{a}_{N0} \neq 0$. Then there is a coordinate transformation that brings ω into the normal form,*

$$\omega = \frac{\sigma}{2} y^2 + f(x) + O(N+1), \tag{2.8}$$

$$f(x) = \sum_{n=1}^{N} c_n x^n, \; c_{N-1} = 0, c_N = \frac{1}{N}, \tag{2.9}$$

$$\sigma = \begin{Bmatrix} -1 \textit{ for } N \textit{ even and } a_{02}/\tilde{a}_{N0} < 0, \\ 1 \textit{ for } N \textit{ even and } a_{02}/\tilde{a}_{N0} > 0 \textit{ or } N \textit{ odd,} \end{Bmatrix} \tag{2.10}$$

and $c_1, ..., c_{N-2}$ are small parameters.

Above the $\tilde{a_{n0}}$'s are given by,

[8]In [2, part III] Brøns shows that for a flow away from a boundary this is the only possible codimension 1 bifurcation.

[9]This is **Theorem 5** in [2] formulated for ω instead of Ψ.

$$\tilde{a}_{N0} = a_{N0} + \text{ a non-linear combination of } a_{nm} \text{ of lower order then } N. \tag{2.11}$$

Using this transformed expression for ω it is easy to see that all critical points near the degenerate point for the system (2.5) are situated along the transformed x-axis and that there are at most $N-1$ of them,

$$\dot{x} = \frac{\partial \omega}{\partial y} = \sigma y = 0 \Leftrightarrow y = 0, \quad \dot{y} = -\frac{\partial \omega}{\partial x} = -f_x(x) = 0. \tag{2.12}$$

The Jacobian of (2.5) at a critical point, determines its type and is given by,

$$J_{(\dot{x},\dot{y})} = \begin{bmatrix} 0 & \sigma \\ -f_{xx}(x) & 0 \end{bmatrix}. \tag{2.13}$$

From (2.13) one can determine the type of a critical point by considering the eigenvalues given by, $\lambda = \pm\sqrt{-\sigma f_{xx}(x)}$. This leaves three options. Either the critical point is degenerate allowing for a bifurcation to happen. The second option is $-\sigma f_{xx}(x) > 0$ which given two real eigenvalues of opposite sign, corresponding to a saddle as the critical point has one attracting eigendirection and one repelling eigendirection. The last option is $-\sigma f_{xx}(x) < 0$ which gives two purely imaginary eigenvalues of opposite sign which corresponds to a center, i.e. either a maximum or minimum in the vorticity.

For a co-dimension 1 bifurcation one has $N = 3$, i.e. the terms of order lower then four determine the behaviour of the system. From Theorem 1 this gives $\sigma = 1$ and reduces the expression for ω to,

$$\begin{aligned} \omega &= \frac{1}{2}y^2 + f(x) + O(4), \\ f(x) &= c_1 x + \frac{1}{3}x^3. \end{aligned} \tag{2.14}$$

Using (2.12) and (2.13) it can be seen that the occurrence of a bifurcation as c_1 is varied may be illustrated in one dimension around $x = 0$ by considering the derivative of $f(x)$,

$$f_x(x, c_1) = c_1 + x^2, \tag{2.15}$$

From (2.15) it is seen that the origin is a degenerate critical point for the system (2.5) when $c_1 = 0$. It is easy to see that in general critical points exist for (2.5) at,

$$x = \pm\sqrt{-c_1} \quad \Leftrightarrow f_x(x, c_1) = 0. \tag{2.16}$$

For $c_1 = 0$ we see that $f_x(0,0) = 0$ and $\frac{\partial}{\partial x} f_x(0,0) = f_{xx}(0,0) = 0$ while $\frac{\partial^2}{\partial x^2} f_x(0,0) = f_{xxx}(0,0) = 2$ and $\frac{\partial}{\partial c_1} f_x(0,0) = 1$. The first two conditions are known as singularity conditions, the next as a non-degeneracy condition and the last as a transversality condition. If these conditions are fulfilled then the bifurcation which occurs at $x = 0$ when c_1 is varied is a saddle-center bifurcation [1, Corollary 8.4]. For the 1D case the singularity conditions simply state that at $x = 0$, $f_x(x, 0)$ is zero valued, has a horizontal tangent and the non-degeneracy condition that $f_x(x, 0)$ has non-zero curvature. The transversality condition states that increasing/decreasing c_1 around zero increases/decreases the value of f_x at $x = 0$ which is necessary for a change in system behaviour to occur.

From (2.16) it may then be seen that for $c_1 < 0$ two critical points exist whereas for $c_1 = 0$ only a single (degenerate) critical point exists and for $c_1 > 0$ no critical points exist. I.e. the saddle-center bifurcation occurs just as c_1 passes zero.

The change in the number of critical points as c_1 is varied is illustrated on figure 2.1.

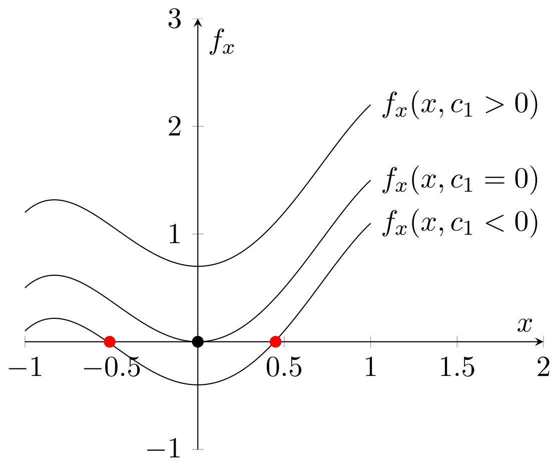


Figure 2.1: Illustration of the creation of two critical points as c_1 is varied across zero. The figure can also be used to illustrate how the search for extrema and saddles in vorticity can be performed using the nullclines. Consider the x-axis as an ω_x nullcline and searching along this to identify $\omega_y = 0$ corresponding to $f_x = 0$.

A bifurcation diagram for the situation showing the annihilation of a center (vortex in the flow) and a saddle as c_1 is varied across zero is presented on figure 2.2.

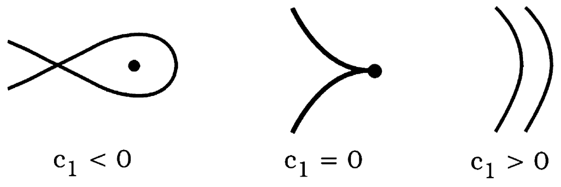


Figure 2.2: Illustration of the creation/annihilation of a vortex (center) and saddle point as the parameter c_1 is varied. The figure is borrowed from [2, Figure 3.2] courtesy of Brøns.

The first illustration in figure 2.2 corresponds to the situation of $f_x(x, c_1 < 0)$ shown in figure 2.1. Here a saddle and a center exist. The second illustration in figure 2.2 corresponds to the situation of $f_x(x, c_1 = 0)$ shown in figure 2.1, which corresponds to the bifurcation point where the critical point becomes degenerate. Lastly, the third illustration in figure 2.2 corresponds to the situation of $f_x(x, c_1 > 0)$ shown in figure 2.1 where no critical points exists.

Note: *A final and important remark on the parameter c_1 is that it is a mathematical parameter which depends on one or more of the physical parameters of the system. Thus for our model problem of the cylinder near the moving wall we have that c_1 may depend on time, Reynolds number and D/G-ratio. This means that $c_1(Re, D/G, t_0)$ may change as either of the three parameters are varied causing the co-dimension 1 bifurcation.*

2.1.2 Tools for Analysing Flow Topology

The challenge now becomes to identify the extrema/saddles for ω as they correspond to the critical points for the dynamical system (2.5). This means that given a fluid velocity field, $\mathbf{u}$, a numerical tool for calculating the position and type of extrema/saddles for ω must be developed. The idea of using a nullcline method for this identification as outlined below was suggested by Brøns and Bisgaard in [12]. Here they applied it to the stream function Ψ instead of the vorticity ω, however, as it was shown above these problems are equivalent.

Identifying the Stationary Point: First it is necessary to determine which extrema and saddles exist for ω at a given time $t = t_i$. As noted earlier these may be found by first determining the ω_x nullclines and then along these determining the points which fulfil $\omega_y|_{(x,y)|\omega_x=0} = 0$. This idea of searching along the $\omega_x = 0$ contour corresponds to identifying the points of intersection between the x-axis and f_x illustrated on figure 2.1. Here the x-axis corresponds to a curve in the 2D domain along which $\omega_x = 0$ and f_x corresponds to ω_y.

With a way to identify the stationary points for the vorticity at each time step having been formulated one needs to determine their type.

Identifying the Type: Whether the critical points are local maxima, local minima or saddle points in the vorticity must now be determined. As was mentioned above the maxima and minima in ω correspond to centres for our dynamical system (2.5) and in turn to vortices in the flow.

To determine the type of the points we consider the eigenvalues of the Hessian matrix for the vorticity. Assuming that the determinant of the Hessian matrix at the point is different from zero (i.e. the critical point is non-degenerate) the type can be determined directly from the eigenvalues. If all eigenvalues of the Hessian matrix evaluated at the point are positive the point must be a minimum in vorticity. Likewise if all the eigenvalues are negative the point is a maximum and if the eigenvalues vary in sign the point is a saddle.

As we work in 2D the following method may be used to determine the type. Name the two eigenvalues λ_1 and λ_2 respectively and denote the Hessian matrix by,

$$H(\omega) = \begin{bmatrix} \omega_{x,x} & \omega_{x,y} \\ \omega_{y,x} & \omega_{y,y} \end{bmatrix}, \tag{2.17}$$

In order to determine the eigenvalues we need to solve the system,

$$H(\omega) - I\lambda = 0, \tag{2.18}$$

for which simple algebra gives that the product of the eigenvalues must fulfil,

$$\lambda_1\lambda_2 = \omega_{x,x}\omega_{y,y} - \omega_{x,y}\omega_{y,x}. \tag{2.19}$$

This means that if the determinant of $H(\omega)$ is positive at a critical point for (2.5) it is either a maximum or a minimum (a center) and if the determinant is negative it is a saddle. If the determinant is positive and one then computes the value of one of the eigenvalues, say λ_1 it is possible to fully determine the types of the points. The formula for calculating the eigenvalues are given by,

$$\lambda = \frac{(\omega_{x,x} + \omega_{y,y}) \pm \sqrt{(\omega_{x,x} + \omega_{y,y})^2 - 4(\omega_{x,x}\omega_{y,y} - \omega_{y,x}\omega_{x,y}))}}{2}. \tag{2.20}$$

We may now construct the lookup table seen in table 2.1 for the type of non-degenerate critical points in 2D.

Condition \ Type	Saddle	Maximum	Minimum
$\lambda_1\lambda_2 < 0$	$\times$		
$\lambda_1\lambda_2 > 0 \wedge \lambda_1 < 0$		$\times$	
$\lambda_1\lambda_2 > 0 \wedge \lambda_1 > 0$			$\times$

Table 2.1: Reference table for determining the type of a non-degenerate critical point for a 2-dimensional quantity.

Based on the prior discussion a simple algorithm for calculating the critical points and their type for the vorticity may be formulated as shown in **Algorithm 1**.

Algorithm 1 Calculating extrema in the vorticity.

```
1: Obtain the velocity field, $\mathbf{u}$, at a sequence of times, $t \in \{t_0, t_1, ..., t_N\}$.
2: Calculate the contours $\omega_x = 0$ at each time step.
3: Calculate the points $(x, y)|_{\omega_y=0}$ along the $\omega_x$ nullclines at each time step.
4: Store the resulting points $(x, y)|_{(\omega_y=0,\omega_x=0)}$ along with the field value $u$ at these points.
5: Calculate the eigenvalues of the Hessian for the vorticity at each of the resulting points
   to determine their type and store this information.
```

2.2 Orthogonal Polynomials and Approximation Theory

This section states some definitions and results from [4, chapter 3] with regards to using orthogonal polynomials for high accuracy approximations to functions which lie in appropriate function spaces. The results are included as they are relevant both to the convergence properties of the SEM presented in section 2.3 and UQ presented in section 2.4. For further explanation, proofs and references to proofs see [4, chapter 3].

Orthogonality Relation: A set of polynomials $\{Q_n(x) \mid x \in I \subset \mathbb{R}\}$ of varying degree $n \in \mathcal{N} \subset \mathbb{N}_0$ is said to be an orthogonal set with respect to the weight function $w(x)$ if,

$$\int_I Q_n Q_m w dx = \gamma_n \delta_{m,n}, \quad \forall m, n \in \mathcal{N}. \tag{2.21}$$

Here $\delta_{m,n} = 1$ for $m = n$ and $\delta_{m,n} = 0$ for $m \neq n$ is the Kronecker delta, $\gamma_n = \int_I Q_n^2 w dx$ and $I \subset \mathbb{R}$ is the support of the polynomials.

Function Spaces, Inner Products and Norms: The following spaces are used in error measurements and results in the following,

- The weighted L^2_w space with support on I:

$$L^2_w(I) = \left\{ v : I \to \mathbb{R} \mid \int_I v^2 w dx < \infty \right\}. \tag{2.22}$$

- The weighted Sobolev space H^k_w with support on I:

$$H^k_w(I) = \left\{ v : I \to \mathbb{R} \mid \frac{d^m v}{dx^m} \in L^2_w(I), \ \forall\ 0 \leq m \leq k \right\}. \tag{2.23}$$

Each space is equipped with an inner product,

- L^2_w:

$$(u, v)_{L^2_w(I)} = \int_I uvw dx, \quad \forall u, v \in L^2_w(I). \tag{2.24}$$

- H^k_w:

$$(u, v)_{H^k_w(I)} = \sum_{m=0}^{k} \left(\frac{d^m u}{dx^m}, \frac{d^m v}{dx^m} \right)_{L^2_w(I)}, \quad \forall u, v \in H^k_w(I). \tag{2.25}$$

and from these one may define a norm in the spaces as,

- L^2_w:

$$\parallel u \parallel_{L^2_w(I)} = (u, u)^{\frac{1}{2}}_{L^2_w(I)}. \tag{2.26}$$

- H^k_w:

$$\parallel u \parallel_{H^k_w(I)} = (u, u)^{\frac{1}{2}}_{H^k_w(I)}. \tag{2.27}$$

Polynomial Approximations: First we define the finite dimensional orthogonal projection of a function $f \in L^2_w(I)$ onto a set of orthogonal polynomials $\{\phi_k(x)\}_{k=0}^N \subset \mathbb{P}_N$ with respect to a positive weight $w(x)$, where $\mathbb{P}_N$ is the space of polynomials of degree N or less:

$$P_N f = \sum_{k=0}^{N} \hat{f}_k \phi_k(x), \quad \hat{f}_k = \frac{1}{\parallel \phi_k \parallel_{L^2_w}} (f, \phi_k)_{L^2_w}, \quad 0 \leq k \leq N. \tag{2.28}$$

It has been shown that the orthogonal projection, $P_N f$ of f is the best approximation to f possible in the space of N'th degree polynomials, $\mathbb{P}_N$, measured in the L^2_w-norm, i.e.

$$\| f - P_N f \|_{L^2_w} = \inf_{\forall \psi \in \mathbb{P}_N} \| f - \psi \|_{L^2_w} \tag{2.29}$$

Below is stated the powerful result of convergence for the orthogonal projection,
For any $f \in L^2_w(I)$:

$$\lim_{N \to \infty} \| f - P_N f \|_{L^2_w(I)} = 0, \tag{2.30}$$

where the rate of the convergence depends on the smoothness of f and the choice of polynomials used for the projection.

Legendre Polynomials: The Legendre polynomials may be defined from a three term recursion relation as,

$$L_{n+1}(x) = \frac{2n+1}{n+1} x L_n(x) - \frac{n}{n+1} L_{n-1}(x), \quad n > 0, \quad L_0(x) = 1, \; L_1(x) = x. \tag{2.31}$$

The polynomials are orthogonal on $I = [-1, 1]$ with respect to $w(x) = 1$ and have the orthogonality relation,

$$\gamma_n = \frac{2}{2n+1}. \tag{2.32}$$

Spectral Convergence: Now follows the important result that all these definitions have been leading up to. Take $I = [-1, 1]$ with $w(x) = 1$ and the set of Legendre polynomials defined above $\{L_n(x)\}_{n=1}^{\infty}$.

Then for any function $f(x) \in H^p_w[-1, 1], p \geq 0$ there exists a constant $C \in \mathbb{R}$ independent of N, such that

$$\| f - L_N f \|_{L^2_w[-1,1]} \leq C N^{-p} \| f \|_{H^p_w[-1,1]} . \tag{2.33}$$

See [4, Page 33] for a proof. This result means that the convergence rate of $L_N f$ towards f in terms of the number of polynomials used in the expansion $L_N f$ is bounded by the smoothness of $f(x)$ i.e. by p. Thus, if orthogonal projection is used to approximate a sufficiently smooth function, very fast convergence, termed spectral convergence, may be expected. Similar results exist for other choices of polynomial sets and corresponding support I and weight functions $w(x)$.

This result of spectral convergence is one of the fundamental properties of both UQ and the SEM which gives them the potential of being strong tools.

Numerical Quadrature: For the application of both the SEM and UQ it is crucial to be able to perform numerical evaluation of integrals very accurately. That is, we seek to approximate the integral,

$$\mathrm{Int}[f] = \int_I f(x)w(x)dx, \tag{2.34}$$

accurately. By using orthogonal polynomials for a numerical quadrature a highly accurate approximation of Int$[f]$ may be obtained. Consider a sequence of orthogonal polynomials, $\{\phi_k\}_{k=1}^N$ and let $\{z_k^{(N+1)}\}_{k=1}^{N+1}$ be the zeros of ϕ_{N+1}. Then by interpolating $f(x)$ using the $N+1$, N'th degree Lagrange polynomials, $\{l_k^N\}_{k=1}^{N+1}$, through the nodes $\{z_k^{(N+1)}\}_{k=1}^{N+1}$, $f(x) \approx \sum_{k=1}^{N+1} f(z_k)l_k^N$, it is possible to approximate (2.34) by,

$$\int_I f(x)w(x)dx \approx \int_I \sum_{k=1}^{N+1} f(z_k)l_k^N w(x)dx = \sum_{k=1}^{N+1} f(z_k)\int_I l_k^N w(x)dx = \sum_{k=1}^{N+1} f(z_k)w_k, \tag{2.35}$$

where $w_k = \int_I l_k^N w(x)dx, k \in \{1, 2, ..., N+1\}$ may be evaluated explicitly if $w(x)$ is known. It turns out that this approximation of Int$[f]$ is exact if $f(x) \in \mathbb{P}_{2N-1}$ on I, see [4, Theorem 3.11]. This means that for functions $f(x)$ which are approximated very accurately by polynomials of order $2N-1$ the numerical quadrature given in (2.35) is a very accurate approximation of (2.34).

2.3 The Spectral Element Method

Historically, the approach to solving PDE problems numerically for complex geometries has been based on the Finite Volume Method (FVM) [13] and the standard Finite Element Method (FEM) [14].

These methods have the strength that they are relatively easy to program, have a well-developed theory behind them as well as being usable and robust for a large class of problems. Also (probably because of these facts) the methods are used extensively by the industry. They have the disadvantage, however, that the numerical solution obtained only converges towards the true solution at a rate of $O(h^2)$ where h is a measure of the element size used to discretize the domain. This means that if one is interested in high accuracy for the solution and/or its derivatives a possibly impractical large number of elements must be used. This makes solving large/complex 2D problems or moderately large 3D problems extremely costly in computations and therefore in time. This fact among others leads to an interest in methods which offer the choice of convergence at a faster rate, i.e. which converge to a given accuracy with fewer Degrees of Freedom (DOF). The Spectral Element Method (SEM) is such a method.

The SEM is a combination of the FEM and the Spectral Method[10]. It benefits from the FEM/FVM's ability to resolve complex geometry while allowing for the possibility of much faster rates of convergence for smooth solutions due to the usage of higher order basis functions like the Spectral Methods. The rate of convergence obtained with the SEM for an infinitely smooth solution is spectral, i.e. $O(h^{p+1})$, where p is the order of polynomials used

[10]See [15] for a good introduction to the Spectral Method.

as basis functions. The convergence rate $O(h^{p+1})$ also highlights the important fact that convergence may be obtained in three ways. Either a fixed polynomial order may be used and the number of elements increased (corresponding to a decrease in h). Alternatively a fixed mesh may be created and convergence obtained by increasing p, and lastly a combination of both.

The SEM is not yet a standard in the industry, however in recent years some companies, e.g. COMSOL, have begun to open their eyes to the method and have included support for it to some degree.

Below is listed some of the general pros and cons of the SEM.

- Pros:
  - For infinitely smooth solutions the SEM converges spectrally, see (2.33), when the polynomial order p used for the basis is increased. Thus the possible convergence rate is only limited by the smoothness of solution.
  - The elemental approach allows simulations on complex geometries with ease.
  - The method is flexible as both h and p convergence may be used to obtain accurate solutions.
  - The SEM allows direct and efficient differentiation of solutions to obtain approximations of higher order derivatives of the solution.
  - The SEM also allows for the use of adaptive methods for mesh refinement (both h and p wise) which in turn allows for an optimal use of resources in the solution process.
- Cons:
  - The upper bound on the stable time step used with explicit time stepping methods scale as $O(p^{-2})$ for the problem at hand, leading to short time steps for high basis order.
  - For solutions which are not infinitely smooth the convergence rate is limited by smoothness. E.g. for solutions which have discontinuous first derivatives the convergence rate is limited to the convergence rate of the FEM/FVM.

The problems being investigated in here have characteristics which make the classical second order accurate FVM/FEM methods seem like suboptimal choices and the SEM like a prime candidate. The characteristics in question are,

- The problems are time dependent and periodic with a transient initial period which must be allowed to disappear over a substantial period of time.
- The problems are open flow problems, which are treated by using a large domains to avoid artefacts introduced by boundary conditions.
- Multiple derivatives of the solution are desired with good accuracy.
- From the physics behind the problem there is no reason to believe that the solution should not be sufficiently smooth.

The first point on the list leads to a high number of iterations in the solution process. This means that if each iteration is very expensive, due to a large number of degrees of freedom (DOF) in the system, the total computational cost will become very high for a large number of iterations. However, as will be touched upon again in section 3.1, a downside to the SEM is that the scaling of the maximum eigenvalue for the weak advection operator[11] used in solving the NS-equation is bounded by $O(p^2)$, where p is the order of the basis used. This fact leads to a restriction on the time step which scales as $O(p^{-2})$, leading to very short stable time steps for high values of p. Thus a trade off between the longest possible time step allowed, i.e. the number of iterations needed, and the computational effort needed to solve the equations in each time step exists. This problem has not been investigated here.

The second point requires that the problem must be solved on a large domain to capture the physics correctly. This is due to the lack of knowledge about the correct boundary conditions in the directions where the open domain must be truncated. This problem is treated in more detail in section 3.2. It means that a large number of elements is needed to discretize the domain which in turn introduces a high number of DOF's.

The third point means that a very accurate solution is needed to allow for good accuracy in the derivatives of the solution. This in turn leads to a very high number of DOF's needed in areas where the solution changes rapidly unless the spatial convergence rate is fast.

The last point means that it may be profitable to use a SEM approach as it will require much fewer DOF's for the same accuracy. This fact is illustrated in figure 2.5 in the following section.

Based on these considerations one could ask the question of how to choose the discretization parameters to exploit high order while avoiding the severe time-stepping constraint. As mentioned this problem has not been treated extensively in this project, and thus a strict time step limit has simply been endured.

2.3.1 Overview of the Method

The technicalities of applying the SEM to the NS- and continuity-equations for general 2D/3D domains are quite extensive and would require far to lengthy an explanation to be included in detail here. Instead the interested reader is encouraged to consult [16] for a full presentation. In order to provide the reader with an overview of the main steps in applying a nodal continuous Galerkin version of the SEM to a PDE without a lot of technical details this section contains an outline of its application to a model problem. The outline is kept very condensed and is merely meant to provide the reader with the basic idea of the steps involved. For a full introduction to the FEM and SEM the reader is referred to [14] and/or [17].

The Goal: The goal of any numerical method for solving PDE's is to determine an approximate solution $\tilde{u}(\mathbf{r})$ to the true solution $u(\mathbf{r})$ for a given problem as well as possible. In the SEM this is done by considering the weak formulation of the PDE and based on this determine $\tilde{u}(\mathbf{r})$ as a linear combination of orthogonal polynomials, since this allows for the possibility of a high rate of convergence.

[11]See [16, chapter 6.]

Model Problem: As a model problem we use the stationary problem,

$$\begin{aligned} \nabla^2 u(\mathbf{r}) &= g(\mathbf{r}), \;\; \mathbf{r} \in (\Omega \subset \mathbb{R}^2), \\ u(\mathbf{r}) &= f(\mathbf{r}), \;\; \mathbf{r} \in \overline{\Omega}, \end{aligned} \tag{2.36}$$

where Ω is the domain, $\overline{\Omega}$ is the domain boundary, u the sought solution, f is a known Dirichlet boundary condition and g is some known forcing function.

Weak Formulation: The application of the SEM requires that the PDE is recast into its weak formulation as follows. Multiply (2.36) by a test function v, to be defined later, which fulfils that it is zero on all Dirichlet boundaries, here $v(\mathbf{r})|_{r\in\overline{\Omega}} = 0$. Now integrate over the domain which yields,

$$\int (\nabla^2 u) v \; d\Omega = \int gv \; d\Omega. \tag{2.37}$$

Utilizing integration by parts on the right hand side of the equation and the restriction of $v = 0$ on the boundary gives,

$$\int \nabla u \nabla v \; d\Omega = -\int gv \; d\Omega. \tag{2.38}$$

Equation (2.38) is called the weak formulation of the problem. It can be shown that a solution to the weak formulation will also be a solution to the strong formulation. Hence a function u solving (2.38) will solve the original problem (2.36).

Approximate Solution, Lifting Formulation and Dirichlet Conditions: The next step is to approximate the true solution u by an approximate solution given by a finite polynomial expansion $\tilde{u}$, presented in more detail in a following paragraph. One approach is then to decompose the approximate solution into a function $\tilde{u}^H$ which is zero on the Dirichlet boundary, and another **known** function $\tilde{u}^D$ that satisfies the Dirichlet condition,

$$\tilde{u} = \tilde{u}^H + \tilde{u}^D, \quad \tilde{u}^H(\overline{\Omega}) = 0, \quad \tilde{u}^D(\overline{\Omega}) = f(\overline{\Omega}). \tag{2.39}$$

By substituting the approximate solution (2.39) into (2.38) one obtains,

$$\begin{aligned} \int \nabla(\tilde{u}^H + \tilde{u}^D)\nabla v \; d\Omega = -\int gv \; d\Omega, \;\; \Leftrightarrow \\ \int \nabla \tilde{u}^H \nabla v \; d\Omega = -\int gv \; d\Omega - \int \nabla \tilde{u}^D \nabla v \; d\Omega. \end{aligned} \tag{2.40}$$

Given a known function, v, the right hand side of (2.40), can be calculated, leaving only the unknown $\tilde{u}^H$ to be determined.

Discretizing the Domain: Just as for the FEM/FVM the domain Ω is divided into a set of M non-overlapping elements covering all of Ω,

$$\Omega = \bigoplus_{n=1}^{M} \Omega_n, \quad \Omega_i \cap \Omega_j = \emptyset \; \forall \; i \neq j \tag{2.41}$$

The decomposition is illustrated in figure 2.3a where the domain Ω is divided into four elements.

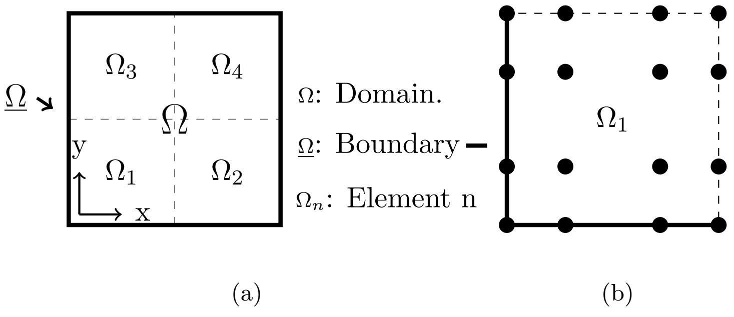


Figure 2.3: Illustration of domain decomposition for the application of the SEM.

For each element a set of local nodes, $\{\mathbf{r}_i\}_{i=1}^{N_n}$ is defined as illustrated on figure 2.3b. The position of these nodes depends on the choice of basis and test functions and are determined such that they maximize the accuracy of the method. For the nodal Galerkin approach using Lagrange polyonimals as basis and test functions the nodes in 2D for quadrilateral elements correspond to the tensor-product of the 1D Gauss-Lobatto quadrature points, [16, section 2.3.4.2].

The decomposition of the domain leads to N global nodes with N_i of them being interior nodes, i.e. the nodes, $\{(x_i, y_i) | (x_i, y_i) \in \Omega \backslash \overline{\Omega}\}$ and N_b being boundary nodes on $\overline{\Omega}$. All the global nodes can also be viewed in terms of N_n local nodes on each of the n elements.

The Sought Solution: As mentioned the approximate solution to (2.38), $\tilde{u} \approx u$ is a finite polynomial expansion of orthogonal polynomials,

$$\tilde{u} = \tilde{u}^H + \tilde{u}^D = \sum_{k=1}^{N_i} \hat{u}_k^H \phi_k(\mathbf{r}) + \sum_{k=N_i+1}^{N} \hat{u}_k^D \phi_k(\mathbf{r}) = \sum_{k=1}^{N} \hat{u}_k \phi_k(\mathbf{r}), \tag{2.42}$$

where $\{\hat{u}_k^H\}_{k=1}^{N_i}$ is a set of undetermined coefficients, $\{\hat{u}_k^D\}_{k=1}^{N_b}$ a set of known coefficients determined by the Dirichlet boundary condition and $\{\phi_k(\mathbf{r})\}_{k=1}^{N}$ a set of global continuous piecewise polynomial spatial basis functions belonging to an appropriate function space. For the model problem the appropriate function space is that of all continuous piecewise polynomials of order at most p.

The use of polynomials of order p is the crucial difference between the FEM and the SEM. In the classical Finite Element Method only first order polynomials, i.e. $p = 1$, are used for all local basis functions which leads to the second order convergence, $O(h^2)$.

The set $\{\phi_k(\mathbf{r})\}_{k=1}^{N}$ is chosen such that it only has a non-zero value on a single element plus at most the neighbours of this element. This choice means that the global basis functions may be represented by the direct sum of a set of local basis functions on each element, $\{\phi_i^{(n)}(\mathbf{r})\}_{i=1}^{N_n}$ as,

$$\phi_k(\mathbf{r}) = \bigoplus_{n=1}^{M} \sum_{i=1}^{N_n} \phi_i^{(n)}(\mathbf{r}_k)\phi_i^{(n)}(\mathbf{r}). \tag{2.43}$$

Each of these local basis functions are polynomials of order up to p. The fact that the global basis functions may be decomposed into local basis functions on each element allows for great flexibility in the shape of the domain as each element may be treated individually. This means that integration etc. may be done on each element independently of the other elements and the final resulting system can be constructed as a direct sum of the smaller elemental systems.

Building a System: The next step is to create a discrete system from which the unknown coefficients $\{\hat{u}_k\}_{k=1}^{N_i}$ may be determined. Replace $\tilde{u}^H$ in (2.40) by its series expression. For each interior node, $j \in \{1, 2, ..., N_i\}$, replace v by the test function $v_j = \phi_j$ to obtain N_i equations of the form,

$$\int \nabla \left(\sum_{k=1}^{N_i} \hat{u}_k \phi_k(\mathbf{r}) \right) \nabla \phi_j(\mathbf{r}) d\Omega = -\int g(\mathbf{r})\phi_j(\mathbf{r}) d\Omega - \int \nabla \tilde{u}^D \nabla \phi_j(\mathbf{r})\ d\Omega, \quad j \in \{1, 2, ..., N_i\}, \tag{2.44}$$

where the right hand side only consists of known functions which may be evaluated analytically or numerically. Rewriting by utilizing that $\hat{u}_k$ are constants we obtain,

$$\sum_{k=1}^{N_i} \hat{u}_k \int \nabla \phi_k(\mathbf{r}) \nabla \phi_j(\mathbf{r}) d\Omega = -\int g(\mathbf{r})\phi_j(\mathbf{r}) d\Omega - \int \nabla \tilde{u}^D \nabla \phi_j(\mathbf{r})\ d\Omega, \quad j \in \{1, 2, ..., N_i\}. \tag{2.45}$$

The system of equations given in 2.45 now consists of N_i equations with N_i unknowns and may be written as a linear system $\mathbf{S}\hat{\mathbf{u}}^H = \hat{\mathbf{g}} + \hat{\mathbf{u}}^D$.

Solving the System: Now the problem becomes solving a linear matrix-vector system of the form,

$$\mathbf{A}\hat{\mathbf{u}} = \hat{\mathbf{b}}, \tag{2.46}$$

to determine the unknown coefficients $\hat{\mathbf{u}}$. This may be done using ones favourite linear solver which should be configured to exploit any properties of the system matrix, e.g. sparsity or symmetry. The solution process may be performed by directly solving the system of equations or using an iterative method to obtain an approximate solution[12].

[12]For parallel solution of the system an iterative method is the only efficient approach.

Accuracy of Solution: The error analysis is very involved but the key point is that the error, $\epsilon = u - \tilde{u}$, in the solution is bounded by expression of the form,

$$\| \epsilon \| \leq C h^{\mu-1} \| u \|, \tag{2.47}$$

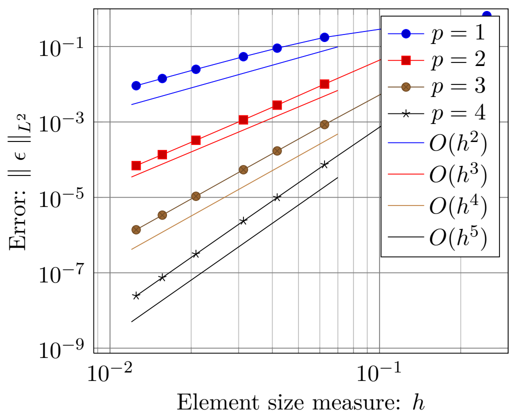
where, $\mu = \min(k, p+1)$, k the smoothness of the solution and p the order of the polynomials used for the expansion. Thus if the solution is sufficiently smooth the convergence rate is spectral in p, see (2.33). This expression again shows that the convergence may be controlled by the element size h, the polynomial order of the basis functions p or both.

Illustrating Convergence: In order to illustrate the rate of convergence of the SEM stated in (2.47), consider the model problem (2.36) on the domain $\Omega = (x, y) \in [0,1] \times [0,1]$, with,

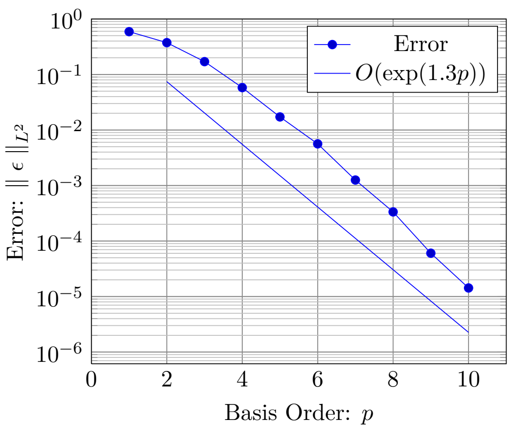
$$\begin{aligned} g(x,y) &= -50\pi^2 \sin(5\pi x)\sin(5\pi y), \;\; f(x,y) = u_{\text{true}}(x,y), \quad (x,y) \in \overline{\Omega}, \\ u_{\text{true}}(x,y) &= \sin(5\pi x)\sin(5\pi y). \end{aligned} \tag{2.48}$$

where u_{true} is the true solution which can easily be verified by insertion. The problem (2.36) has been solved with the SEM framework **FEniCS**, see [17], using the nodal Galerkin approach outlined above. A minimal python script for the **FEniCS** solver is provided in appendix A.5.4.

First, the problem was solved for an increasing number of elements, i.e. decreasing value of mesh size h, with different fixed polynomial orders. Secondly, it was solved for a fixed number of elements while increasing the polynomial order. The error between the true solution and the solution obtained using the SEM measured in L^2-norm was then calculated and the results presented in figure 2.4.



(a) Convergence rate for different polynomial order along with tendency lines.



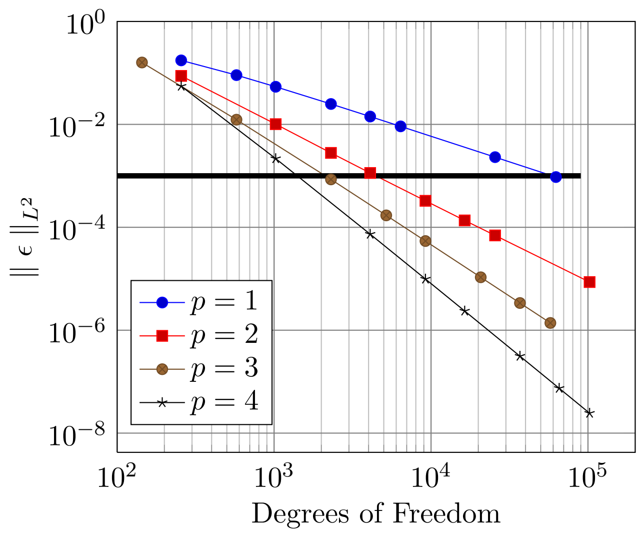
(b) Convergence plot for increasing polynomial order and fixed mesh

Figure 2.4: Illustration of convergence rates for the SEM solution to the problem (2.36) with the functions g and f given in (2.48).

Figure 2.4a shows the error versus element side length. From here it can be seen that the rate of convergence is bounded as $O(Ch^{p+1})$ for each fixed p as it is stated in (2.47). From figure 2.4b the spectral convergence is observed as h is kept fixed and p allowed to increase.

A crucial point is now whether it is more cost efficient to solve the problem using many elements, i.e. low h or high basis order, i.e. high p.

The degrees of freedom in the system (2.46) needed to obtain a desired accuracy in the solution is one important measure for this. Based on this measure an illustration of the clear advantage of increasing p for the problem at hand is provided in figure 2.5.



(a)

Figure 2.5: Illustration of degrees of freedom versus the error measured in L^2-norm, $\| \epsilon \|_{L^2} = \| u_{\text{true}} - \tilde{u} \|_{L^2}$. Each graph shows how the error obtained using a given polynomial order on each element while increasing the number of elements. A black line has been introduced to mark the error $\| \epsilon \|_{L^2} = 10^{-3}$.

Figure 2.5 shows a plot of the error versus the degrees of freedom along with a black line denoting an error of $\| \epsilon \|_{L^2} = 10^{-3}$. From here it is seen that increasing p decreases the degrees of freedom needed to obtain a certain accuracy significantly, supporting utilizing the SEM over the classical FEM for the problem at hand.

2.4 Uncertainty Quantification

In most areas of science throughout history the modelling approach has to a large extend been deterministic in nature. The world however is full of uncertainties and thus the reality being modelled is far more stochastic then deterministic. When developing new technology there are uncertainties in experiments. When manufacturing products there are uncertainties in every production step. When using the products for their real world purposes the environment is far from as perfect as the idealization used in models. These observations all point to the introduction of uncertainties in the modelling process.

One way of handling uncertainties in the modelling process is to identify the parameters of the model which contain uncertainties. Then perform a large number of realizations using e.g. the Monte Carlo approach as outlined in section 2.4.2. Although very simple to understand and execute this approach may be very expensive to use. As a very relevant example we may choose the problem of the cylinder near a moving wall. This problem is expensive to solve as roughly ten hours of calculation and post processing time is needed per simulation. Thus if good accuracy on the statistics is desired the Monte Carlo approach becomes impractical with the resources available for this work. Therefore it is interesting to investigate the possibility of a much more efficient method for smooth problems called the stochastic collocation method explained in section 2.4.4. This method is based on generalized Polynomial Chaos (gPC) as explained in section 2.4.3.

2.4.1 Statistical Quantities

Given some quantity that can be described by the function g, which among other depends on a random variable X. It is often of interest to obtain some information about different statistical quantities of g which depend on X. The statistical quantities often considered in application, and hence also considered in the present work, are the mean and variance. These are also known as mean,

$$\mu_{g(X)} = \int_{I_X} g(X) f_X(x) dx, \tag{2.49}$$

and variance,

$$\sigma^2_{g(X)} = \int_{I_X} (g(X) - \mu_X)^2 f_X(x) dx. \tag{2.50}$$

Here X is the random variable, which for the purpose of this work is taken to be the Reynolds number, Re. Examples of quantities $g(X)$ considered here are the Strouhal number, St and Base pressure coefficient C_{bp}. $f_X(x)$ is the probability density function and I_X is the domain over which X has support.

For the reader unfamiliar with probability theory the basic concepts are outlined in chapter 1 of [4].

2.4.2 The Monte Carlo Approach

A widely used way of handling uncertainty in a model is by sampling using the Monte Carlo method as outlined below, see also [4, page 53],

1. A set of M independent identically distributed random numbers, $x^{(i)},\ i \in \{1, 2, ..., M\}$, are generated according to a chosen probability distribution.
2. For each of the M numbers the model equations are solved to obtain the quantity of interest.
3. The required solution statistics are estimated.

A clear strength of the Monte Carlo method is that it is non-intrusive, meaning that it may be used with existing deterministic codes without any modification. Unfortunately the Monte Carlo method suffers from the problem of slow convergence mentioned earlier. As may be shown by using the Central Limit Theorem [4, Theorem 2.26] the error convergence rate of the method is $O(M^{-\frac{1}{2}})$. This means that in order to reduce the error by one order of magnitude roughly one hundred times as many simulations must be performed. For a model with long execution time, as is the case for the problems considered in the present work, this fact quickly becomes a severe bottleneck. This makes the Monte Carlo approach nigh impossible to apply for certain problems.

It should be mentioned, however, that the convergence rate of the Monte Carlo method is independent of the number of uncertain parameters. Hence whether one or one hundred parameters in a system contain uncertainty the convergence rate is the same. To the authors knowledge this is a property not shared by any other method. This means that for systems with a large number of uncertain parameters the Monte Carlo approach is very good despite its slow convergence.

2.4.3 generalized Polynomial Chaos

This section is an extract from chapter 5 of [4] modified to the needs for the present work. For the work done here only a single parameter is allowed to contain uncertainty. This means that only a univariate problem is considered and thus all multivariate theory has been left out. Let X be a continuous random variable with a known probability density function $f_X(x)$ and finite moments,

$$\mathbb{E}(|X|^{2m}) = \int |x|^{2m} f_X(x)dx < \infty, \quad m \in \{\mathcal{N}\}, \tag{2.51}$$

with $\mathcal{N} = \{0, 1, ..., N\}$. One can then define the gPC basis functions as the orthogonal polynomials which satisfy,

$$\mathbb{E}(\phi_n(X)\phi_m(X)) = \int \phi_m(x)\phi_n(x)f_X(x)dx = \gamma_n\delta_{mn}, \quad m, n \in \{\mathcal{N}\}, \tag{2.52}$$

where δ_{nm} is the Kronecker delta, with,

$$\gamma_n = \mathbb{E}[\phi_n^2(X)], \quad n \in \{\mathcal{N}\}. \tag{2.53}$$

This construction results in $\phi_m(x)$ being orthogonal polynomials for $x \in \mathbb{R}$ with respect to the weight function $f_X(x)$. This means that the choice of polynomials depends on the distribution of the random variable X.

The distributions considered in the project are the Gaussian and uniform distributions[13]. For these distributions the polynomial bases becomes the Hermite and Legendre polynomials respectively. The distributions as well as the Rodriguez formulas and recursion relations for the corresponding polynomials along with the polynomial orthogonalization constants are given by,

[13]The reasoning behind choosing these distributions is given in section 8.4.

- **Gaussian PDF $\mathcal{N}(0,1)$, Hermite polynomials:**

$$f_{X,G}(x) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{x^2}{2}}, \tag{2.54}$$

$$H_n(x) = (-1)^n \mathrm{e}^{\frac{x^2}{2}} \frac{d^n}{dx^n} \mathrm{e}^{-\frac{x^2}{2}}, \tag{2.55}$$

$$H_{n+1}(x) = xH_n(x) - nH_{n-1}(x), \tag{2.56}$$

$$\int_{-\infty}^{\infty} H_n(x)H_m(x)f_{X,G}(x)dx = \gamma_n\delta_{n,m} = n!\delta_{n,m}. \tag{2.57}$$

- **Uniform PDF $\mathcal{U}(-1,1)$, Legendre polynomials:**

$$f_{X,u}(x) = \frac{1}{2}, \tag{2.58}$$

$$L_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}\left[(x^2-1)^n\right], \tag{2.59}$$

$$(n+1)L_{n+1}(x) = (2n+1)xL_n(x) - nL_{n-1}(x), \tag{2.60}$$

$$\int_{-\infty}^{\infty} L_n(x)L_m(x)f_{X,u}(x)dx = \gamma_n\delta_{n,m} = \frac{2}{2n+1}\delta_{n,m}. \tag{2.61}$$

The orthogonality properties of the gPC with respect to the PDF of X ensure that the polynomials may be used as a basis for functions g in terms of X. The following definition [4, Definition 5.4] is what is understood as a Strong gPC approximation of a function g of the random variable X with a known PDF.

Strong gPC approximation: *Let $g(X)$ be a function of a random variable X whose PDF is $f_X(x)$ and support is I_X. A general polynomial chaos approximation in a strong sense is $g_N(X) \in \mathbb{P}_N(X)$, where $\mathbb{P}_N(X)$ is the space of polynomials of X of degree up to $N \geq 0$, such that $\parallel g(X) - g_N(X) \parallel \to 0$ as $N \to \infty$, in a proper norm defined on I_X.*

The N'th order orthogonal projection of $g(X) \in L^2_{f_X(x)}(I_X)$ onto $\{\phi_k\}_{k=1}^N \in L^2_{f_X(x)}(I_X)$ given in (2.28) is a strong gPC approximation, and may be written as,

$$P_N[g] = \sum_{k=0}^{N} \hat{g}_k\phi_k(X), \quad \hat{g}_k = \frac{1}{\gamma_k}\mathbb{E}(g(X)\phi_k(X)), \quad \phi_k \in L^2_{f_X(x)} \forall k. \tag{2.62}$$

This means that the projection has the properties (2.29) and (2.30) which leads to $P_N[g]$ converging spectrally to $g(X)$. One can further conclude that the error introduced by the finite continuous orthogonal projection of g is due to the truncation of the modes with $n > N$.

Using (2.62) directly to approximate $g(X)$ does not make sense since evaluating the $\hat{g}_k$'s requires full information about $g(X)$ which is what we seek to approximate. Therefore in order for this construction to be usable for determining an accurate approximation of the functional relationships and statistics for g with only a limited knowledge of $g(X)$ the $\hat{g}_k$'s must be evaluated based only on this limited knowledge. Here the stochastic collocation method (SCM), which makes use of numerical quadrature or interpolation, may be applied. This is described in the next section.

The statistics of interest, i.e. mean and variance, may be approximated from very cheaply from $P_N[g]$ utilizing the orthogonality of the basis functions as follows,

Mean:

$$\mu_g = \mathbb{E}(g(X)) \approx \mathbb{E}(P_N[g](X)) = \int_{I_X} \left(\sum_{k=0}^{N} \hat{g}_k \phi_k(X) \right) f_X(x) dx = \hat{g}_0, \tag{2.63}$$

Variance:

$$\sigma_g^2 \approx \mathbb{E}((P_N[g](X) - \mu_g)^2) = \int_{I_X} \left(\sum_{k=1}^{N} \hat{g}_k^2 \phi_k(X)^2 \right) f_X(x) dx = \sum_{k=1}^{N} \hat{g}_k^2 \gamma_k. \tag{2.64}$$

It is noted that for these results to hold in the form written above the polynomial basis should not be normalized.

2.4.4 Stochastic Collocation Approach

Based on the gPC theory outlined above there are now two general ways of utilizing the methods of UQ. The first is to build the uncertainty of the parameters directly into the model equations. This is referenced as the stochastic Galerkin method presented in [4, Chapter 6]. The second is to utilize a deterministic model and sample the parameters which contain uncertainties cleverly. This method is referred to as the stocastic collocation method (SCM), see [4, Chapter 7].

The stochastic Galerkin method requires the construction of a new model problem and thus also a new solution procedure. This can be a very cumbersome and daunting task for complex problems and luckily it is not the only way to utilize UQ. Instead one may use the stochastic collocation method, as this is a completely non-intrusive method. This moves the UQ from the solution phase to the pre and post processing phases and the solution of the model equations may therefore be performed with existing deterministic methods and codes.

Consider a model problem consisting of a PDE system with appropriate IC's and BC's depending on a parameter X suffering from uncertainty.

$$\frac{d}{dt} u(\mathbf{r}, t, X) = \mathcal{L}(u), \quad (\mathbf{r}, t, X) \in \ \Omega \times [0, T] \times I_X, \tag{2.65}$$

$$\mathcal{B}(u) = 0, \quad (\mathbf{r}, t, X) \in \ \overline{\Omega} \times [0, T] \times I_X, \tag{2.66}$$

$$u = u_0, \quad (\mathbf{r}, t, X) \in \ \Omega \times [t = 0] \times I_X, \tag{2.67}$$

where $\mathcal{L}$ and $\mathcal{B}$ are spatial operators, Ω the spatial domain, T a final time and I_X the domain of support for X. The problem (2.65)-(2.67) now depends on the stochastic variable X which is problematic as this changes the nature of the problem. But instead of having to account for the dependency on the variable X over all of I_X the SCM utilizes knowledge about the distribution of X to choose a set of collocation nodes $\Theta_M = \{X_j\}_{j=1}^{M}$ at which the model problem must be solved. The nodes should be chosen in accordance with the basis $\{\phi_n\}$ determined from the distribution of X to provide the basis of an accurate evaluation of the expansion coefficients $\hat{g}_k$.

This reduces the problem (2.65)-(2.67) to a set of M problems of the form,

$$\frac{d}{dt}u(\mathbf{r},t,X_j) = \mathcal{L}(u), \quad (\mathbf{r},t) \in \ \Omega \times [0,T], \quad j \in \{1,2,...,M\} \tag{2.68}$$

$$\mathcal{B}(u) = 0, \quad (\mathbf{r},t) \in \ \overline{\Omega} \times [0,T], \tag{2.69}$$

$$u = u_0, \quad (\mathbf{r},t) \in \ \Omega \times [t=0]. \tag{2.70}$$

Thus the stochastic problem (2.65)-(2.67) has been reduced to M deterministic problems of the form (2.68)-(2.70). These problems may then be solved using already existing deterministic solvers and the UQ process has thus been moved to,

- **Pre Processing:**
  - Determine the distribution of X.
  - Chose the correct basis ϕ_n based on the distribution of X.
  - Determine the set of nodes $\Theta_M = \{X_j\}_{j=1}^M$ from the choice of basis $\{\phi_n\}$. (One should use the zeros of $\{\phi_{M+1}\}$ for high accuracy quadratures).
- **Post Processing:**
  - Calculate the value of the quantity of interest at each node, $g(X_j)$.
  - Use $g(X_j)$ to determine the coefficients $\hat{g}_k, k \in \{1,2,...,M\}$ using an appropriate method.
  - Determine $P_N[g]$ based on the $\hat{g}_k$'s to obtain a functional relationship of X and calculate any desired statistics.

From the process described above a few problems remain to be addressed. The first is how to determine the distribution of X. This requires either experimental knowledge of the problem being modelled or a well founded guess and thus there is no universal answer. The next is the choice of Θ_M which allows accurate determination of the $\hat{g}_k$'s. There are two ways to do this, one is interpolation and the other discrete projection, see [4, page 79]. In this work interpolation is chosen which leads to the following problem,

Interpolation: The idea is to approximate the continuous projection of g by calculating $\tilde{g}_k \approx \hat{g}_k$ based solely on the knowledge of $g(X)$ at the nodes $X_j, j \in \{0,1,..,N\}$ which yields a "discrete" projection. Hereby one obtains,

$$P_{N,D}[g] = \sum_{k=0}^{N} \tilde{g}_k \phi_k(X) \approx P_N[g]. \tag{2.71}$$

Using interpolation to obtain the $\tilde{g}_k$'s means requiring that $P_{N,D}[g](X_j) = g(X_j) \ \forall \ j$. This leads to a system of M equations which may be written as,

$$\mathbf{A}^T \tilde{\mathbf{g}} = \mathbf{g}, \tag{2.72}$$

with $A_{k,j} = \phi_k(X_j),\ 0 \leq k \leq N,\ 1 \leq j \leq M$, being a Vandermonde-like matrix, $\tilde{\mathbf{g}} = (\tilde{g}_0, \tilde{g}_1, ..., \tilde{g}_N)^T$ the vector of expansion coefficients and $\mathbf{g} = (g(X_0), g(X_1), ..., g(X_M))^T$. In order to be able to solve the problem uniquely it must be well posed i.e. one must require that $M = N$. For high accuracy in our univariate case the nodes Θ_M should be chosen as the zeros of ϕ_{N+1}, which at the same time fulfils the demand that $M = N$.

Solving this system yields the expansion coefficients and thus $P_{N,D}[g]$ may be constructed and statistics calculated.

The difference between the continuous orthogonal projection and the interpolation based "discrete" projection is termed the aliasing error and is caused by the finite discrete sampling of the continuous function g.

3 Discretizing the Problem

This chapter provides an overview of the temporal and domain discretizations used for the model problems to enable the generation of numerical solutions. Section 3.1 presents the time stepping scheme used for solving the model equations (1.11)-(1.12) with **Nektar++**. Section 3.2 covers the discretization of the domain.

3.1 The Time Stepping Scheme

The model problems are solved using the incompressible Naiver-Stokes solver **IncNavierStokesSolver** from the **Nektar++** framework. Here the Spectral Element Method is used for the spatial discretization, see [16] for a thorough derivation and explanation of the SEM applied to the NS-equation including numerous optimization procedures for increased performance. The temporal discritization is done using the velocity correction scheme presented below.

The Velocity Correction Scheme: The incompressible Navier-Stokes solver implemented in **Nektar++** uses the velocity correction scheme[14]. It is outside the scope of this thesis to go into details with the derivation of the scheme as it is not the focus of the work presented here. For these details the reader is referred to the work of Karniadakis et. al. in [18] and to [16, section 8.3].

The scheme is a splitting scheme which means that the pressure and velocity is handled separately. The scheme leads to second order accuracy in time for both pressure and velocity, see [16, section 8.3.4].

As seen below the non-linear advection term $\mathbf{u} \cdot \nabla \mathbf{u}$ is handled explicitly in time while the diffusion term, $\nabla^2 \mathbf{u}$ is handled implicitly. This means that the stability of the scheme with regards to the size of the time step is limited by the way the advection step is treated. The solution at time step n is denoted $\mathbf{u}^n$ and p^n for the velocity field and pressure field respectively.

The scheme is as follows:

1. **Explicit treatment of first intermediate velocity and advection term:**

$$\tilde{\mathbf{u}} - \sum_{q=0}^{J-1} \frac{\alpha_q}{\gamma_0} \mathbf{u}^{n-q} = \Delta t \sum_{q=0}^{J-1} \frac{\beta_q}{\gamma_0} [-\mathbf{u} \cdot \nabla \mathbf{u}]^{n-q},$$

2. **Poisson Equation is solved to time step the Pressure:**

$$\nabla^2 p^{n+1} = (\frac{\gamma_0}{\Delta t}) \nabla \cdot \tilde{\mathbf{u}},$$

with a higher order pressure boundary condition,

$$\frac{\partial p^{n+1}}{\partial n} = -\left[\frac{\partial \mathbf{u}^{n+1}}{\partial t} + \nu \sum_{q=0}^{J-1} \beta_q (\nabla \times \nabla \times \mathbf{u})^{n-q} + \sum_{q=u}^{J-1} \beta_q [\mathbf{u} \cdot \nabla \mathbf{u}]^{n-q} \right] \cdot n$$

[14]This version of the scheme is taken from `http://www.nektar.info/wiki/3.3/Tutorial/IncNavierStokesSolver/VCS`

3. **Explicit evaluation of second intermediate velocity field:**

$$\tilde{\tilde{\mathbf{u}}} = \tilde{\mathbf{u}} - \frac{\Delta t}{\gamma_0}\nabla p^{n+1}.$$

4. **Determining the velocity at new time step implicitly from the diffusion step:**

$$\left(\nabla^2 - \frac{\gamma_0}{\Delta t\nu}\right)\mathbf{u}^{n+1} = -\left(\frac{\gamma_0}{\Delta t\nu}\right)\tilde{\tilde{\mathbf{u}}}.$$

Above J is the integration order used, $\hat{u}$ and $\hat{\hat{u}}$ are two intermediate states for the velocity, $\nu = \frac{\mu}{\rho}$ is the kinematic viscosity, Δt is the time step and γ_0, α_q and β_q are sets of constant which depend on J.

For all simulations second order integration where used. The coefficients γ_0, α_q and β_q for the second order scheme used are given in table 7.1.

Name	γ_0	α_0	α_1	β_0	β_1
Value	3/2	2	- 1/2	2	-1

Table 3.1: Table displaying the γ_0, α_q and β_q values for the second order stiffly stable time integration scheme.

Choosing the time step: As seen above the diffusion step is handled implicitly which makes the advection step the limiting factor for the time step size. Here a CFL-like condition, see [19, section 10.7], must be met in order to assure the stability of the method. It is outside the scope of this work to go into details about the stability analysis of time stepping schemes but the following results from [16, section 6.3.1] have been used to estimate a ***largest stable time step***.

The stability condition for time stepping the advection operator may found by considering the maximum eigenvalue of the weak advection operator, as explained in [16, section 6.3.1]. The requirement for a stable time step is that the time step multiplied by this eigenvalue remains in the stability region Ω_{stable} of the method, i.e. $\Delta t \cdot \lambda_{max} \in \Omega_{\text{stable}}$. This leads to the following limit,

$$\Delta t \leq \frac{\alpha_{im}}{C(\mathbf{V}, g)P^2}, \tag{3.1}$$

where P is the polynomial order of the basis used for the SEM spatial discretization, C is a function depending on $\mathbf{V}$ which is the advection velocity, and g which is a geometric factor depending on the mesh. Finally α_{im} is the intersection of the stability region for the method used with the imaginary axis[15]. The P^2 dependence of the eigenvalue is shown experimentally in [16].

$C(\mathbf{V}, h)$ is highly problem dependent [16, page 317] and it is suggested in [16] to estimate $C(\mathbf{V}, g)$ by an expression which may be formulated as $C(\mathbf{V}, g) \approx \frac{0.2 \max|\mathbf{V}|}{g}$[16]. This provides the estimate for the maximum time step as,

[15]The reader may consult [19, chapter 7] for a discussion of stability regions for numerical time stepping methods.

[16]This is the suggested value for a 1D problem but since no value is suggested for 2D it is the value used in the present calculations.

$$\Delta t \leq \frac{\alpha_{im}}{\frac{0.2 \max |\mathbf{V}| P^2}{g}}. \tag{3.2}$$

Equation (3.2) has been used as a guideline for identifying a stable time step. Using the second order method in the velocity correction scheme presented above one finds, $\alpha_{im} \approx 0.15$. The order of basis functions used are $P = 10$ and the smallest element side length is around $g = 0.1$. Doing some initial simulations at stable time step it was found that the highest speed of the fluid was below $\max |V| = 2$. Plugging these values into 3.2 yields a time step value of,

$$\Delta t \approx 0.00035. \tag{3.3}$$

Based on this estimate a set of experiments were then performed to investigate the stable time step for the highest resolution mesh, closest to the wall and at Reynolds number, $Re = 300$. Through this investigation it was found that the time step of $\Delta t = 0.0005$ was the largest stable time step.

As a side note this discussion of choosing a stable time step highlights a crucial weakness of the SEM once again. For the classical FEM, $P = 1$, which means that the allowed time step for a given mesh is much larger. However the number of elements needed when using the classical FEM to obtain the same accuracy as for the SEM can be impossibly big which makes solving the equation at each time step extremely expensive.

3.2 The Discrete Domain

Another necessary step before simulations can be performed is to determine a suitable size of the domain of interest and how to discretize it. As the model is of an open flow problem a large domain is needed to assure that blockage is not introduced by the far-field boundary conditions[17].

Domain Size: In order to avoid blockage the domain for the cylinder in free flow has been chosen to extend over the range: $x \times y = [-10D, 30D] \times [-15D, 15D]$ where D is the cylinder diameter. This choice was made based on an investigation performed by Huang and Sung [8] which showed that no significant change in the solution could be observed if the domain was increased further.

For similar domain parameters Huang and Sung found that both the Strouhal number, St, and base pressure coefficient, C_{bp}, agree well with previous simulation results presented in [10] and [7]. A series of simulations validating the choice of domain size have been performed, and the results are presented in chapter 7.2.

Based on the choice of domain for the cylinder in free flow, the domain for the cylinder near a moving wall is chosen to extend over $x \times y = [-10D, 30D] \times [0, 20D]$ to assure that the half open flow is simulated accurately.

[17]Blockage is a term for the fact that the far-field boundary conditions introduce artificial changes in the flow which a real world open flow would not. Another approach is to construct BC's which allow a smaller domain, however this has need been investigated here.

Mesh Construction: The last step is to create a mesh which captures the features of the chosen domain on which to solve the discretized model problems.

As stated above the open flow property dictates either sophisticated boundary conditions or a large domain which may lead to a huge number of DOF's. This makes the problem very expensive to solve. However initial investigations showed that in most of the domain upstream of the cylinder both the pressure and velocity are almost constant in time. For the free stream problem this is also true both above and below the cylinder. When introducing the wall it remains true above the cylinder away from the wall. The areas of almost constant pressure and velocity are illustrated on figure 3.1,

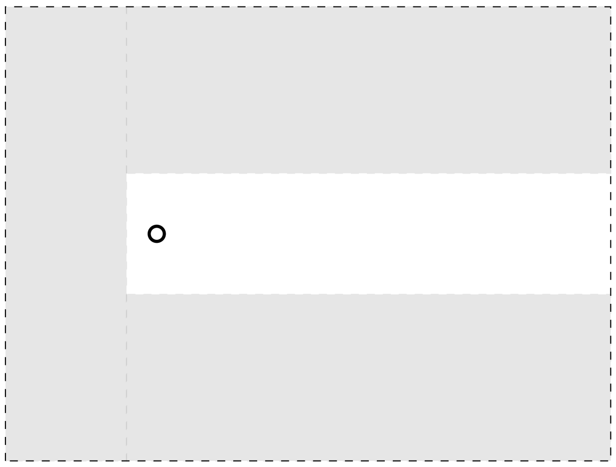


Figure 3.1: Domain for open flow problem with areas of almost constant velocity and pressure highlighted.

The fact that the pressure and velocity are almost constant in these areas means that far fewer DOF's are needed to capture the solution with good accuracy. This fact lead the author to construct hybrid meshes consisting of both quadrilateral and triangular elements in order to save as many DOF's as possible in areas where the solutions only vary slightly. The choice of using hybrid meshes makes it easy to vary the DOF's throughout the domain to get a good distribution which captures the solution without wasting calculation time. Illustrations of hybrid meshes used in the simulations are presented in figure 3.2.

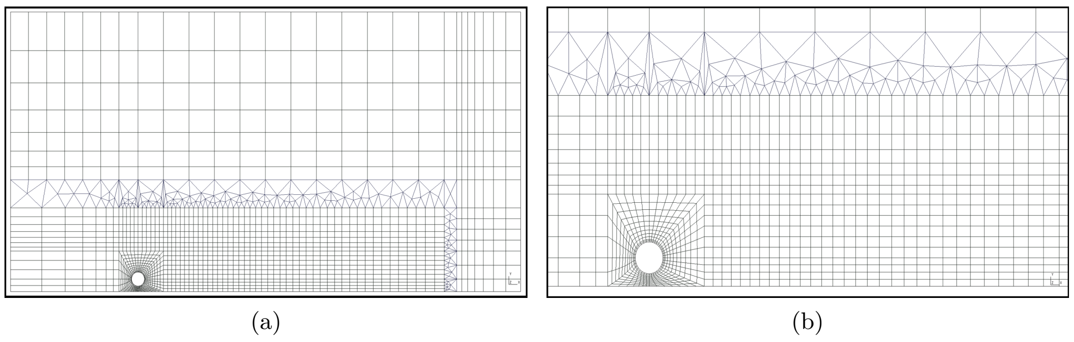
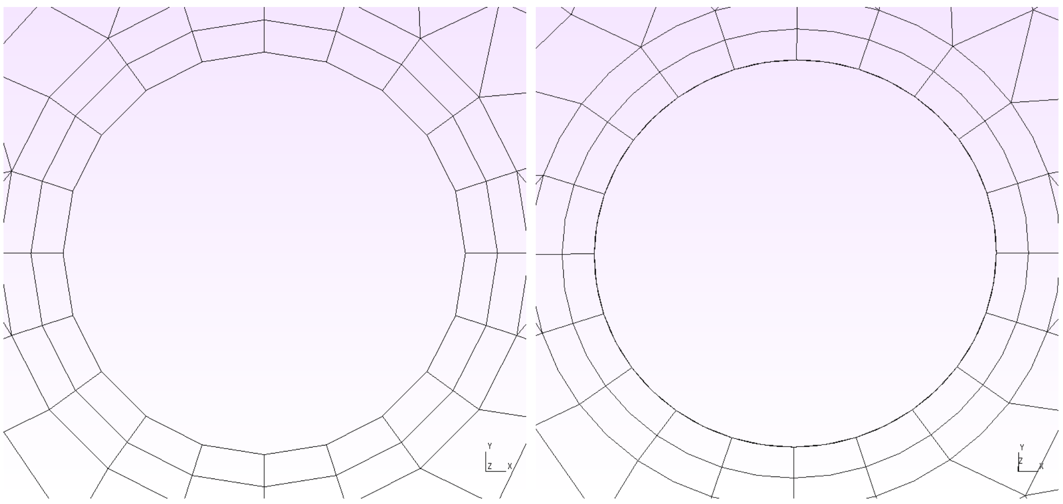


Figure 3.2: Examples of the hybrid meshes used. (a) full domain. (b) zoom close to cylinder and the interesting part of the domain.

From figure 3.2 it is seen that the area around and behind the cylinder is densely meshed while the area away from the cylinder contains only few elements. This design choice saves DOF's while assuring high resolution in areas where the solution changes rapidly.

Curvilinear Elements: The **Nektar++** framework supports curvilinear elements (i.e. elements with curved edges) to allow for better representations of curved geometries with fewer elements. This is important for maintaining the spectral accuracy of the methods along curved geometry. If only linear polynomials were used to resolve the cylinder a massive amount of elements would be needed to approximate the shape of the cylinder accurately. However introducing a lot of elements defeats the purpose of using the SEM. Thus in order to represent the geometry of the cylinder with high accuracy without a large number of elements, curvilinear elements are used. All meshes generated have therefore been modified to use curvilinear elements of appropriate order along the cylinder.

Figure 3.3 illustrates the improvement in resolving the cylinder shape obtained from using third order polynomials compared to using first order polynomials, with the same number of elements.



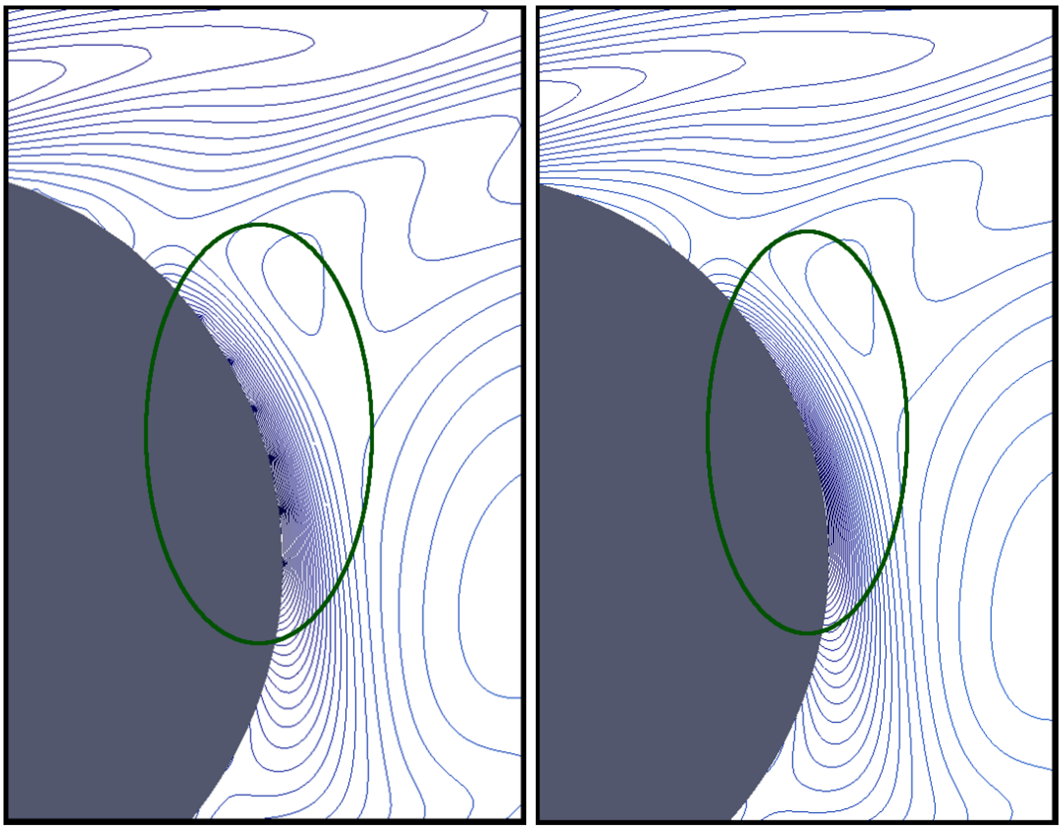
(a) First order polynomials. (b) Third order polynomials.

Figure 3.3: Illustration of advantage of using higher order polynomials to mesh a circular object.

From the figure it is easy to see the improvement in the circular shape as the polynomial order is increased.

Based on the considerations stated above *all simulations have been performed using tenth order curvilinear elements around the cylinder* to assure a highly accurate resolution of the cylinder.

Example of Underresolved Geometry: A small illustration of the problem with underresolving the geometry is presented here. Figure 3.4 illustrates how underresolving the cylinder by using low order polynomials to mesh it causes problems. The figure shows a close up of the cylinder surface downstream of the flow. In figure 3.4a third order polynomials have been used to approximate the cylinder. Here small vortices are seen to have formed along the cylinder. These vortices are formed at the intersection between elements along the cylinder. This is due to the errors in resolving the problem domain to sufficient accuracy introduced by the low order polynomial approximation of the cylinder.



(a) Cylinder resolved using third order polynomials. Small vortices are clearly seen to have formed on the upper half of the cylinders backside.

(b) Cylinder resolved using tenth order polynomials. No small vortices are observed.

Figure 3.4: Illustration of vorticity contours at an instant in time for flow on a zoom on the domain behind the cylinder. The parameters used for both simulations are $Re = 280$, $D/G = 5/3$.

Figure 3.4b shows the same extract from the domain at approximately the same point in time. For this simulation tenth order polynomials have been used to mesh the cylinder. Here it is seen that no small vortices are observed as the error of using polynomials to approximate the cylinder surface has become insignificant.

4 Software

This chapter provides an overview of the software used for the pre processing, simulations and post processing of simulation data preparing it for final data analysis. This includes both software written by the author and open source software. The chapter is included to provide the reader with enough information to understand and use the developed software and can be skipped without loss with respect to understanding the results presented in the following sections.

Section 4.1 provides an overview of the software used in the pre processing phase. Section 4.2 gives a brief description of the framework **Nektar++** used to perform the CFD calculations, as well as to calculate the needed derived quantities and norms. Lastly section 4.3 describes the different pieces of software written for post processing the data and performing the UQ analysis.

4.1 Pre Processing

As described in section 3.2 a mesh which provides the basis for correct and sufficiently accurate results is needed in order to perform CFD simulations using SEM based software. For any none-trivial domains the requirement of a mesh automatically creates the need for meshing software. All mesh generation for the project have been performed using the recognised open source mesh generator **Gmsh** for which thorough documentation is available in the ***Gmsh*** *reference manual* [20]. **Gmsh** uses the file type **.geo** for the geometry describing the problem domain and the type **.msh** for the final mesh of the domain. The user creates a geometry in a **.geo** file either directly in a text editor or using the **Gmsh** GUI[18]. One then uses **Gmsh** to convert the geometry to a **.msh** file intended for use in simulation software. A **.geo** file containing a the geometry for a mesh of the domain with a cylinder near a wall with instructions on how to generate the mesh file is provide in appendix A.1.

In order to reconstruct the meshes the reader simply have to:

1. Download and install **Gmsh**.
2. Load the **.geo** files into the program.
3. Set the order of the elements to allow for curved elements along non-straight domain boundaries.
4. Create the mesh and save it as a **.msh** file as explained in the appendix.

For more information about **Gmsh** and how to accomplish the above mentioned steps the reader is referred to the ***Gmsh*** *reference manual* [20].

Converting Gmsh to Nektar++: In order to use the mesh generated with **Gmsh** it must be converted to the format used by **Nektar++**. This is done using the build in mesh converter in **Nektar++** called **MeshConvert**. In a Linux terminal the converter is executed with the following command,

```
"/path/to/nektar++"/builds/utilities/PreProcessing/MeshConvert/MeshConvert InputName.msh OutputName.xml
```

[18]GUI is an abbreviation for Graphical User Interface.

This produces an **.xml** file with the name **OutputName.xml**. The file contains the mesh in a format **Nektar++** understands. The last step before executing the solver is to add problem related information as well as solver related information to the **.xml** file.

This information includes, what fields are to be calculated (i.e. velocity in 1D, 2D or 3D and pressure), the order of the expansion used for each field, general solver information, simulation and solver parameters, boundary conditions and initial conditions. An example of an input **.xml** used for the simulations in this project with the geometry removed may be found in appendix A.2.

4.2 Simulation

As mentioned multiple times already all simulations have been performed using the SEM based framework **Nektar++** [5] written in **C++**. Appendix A.3 contains an overview of version 3.3 of the Nektar++ software framework. From the **Nektar++** framework the solver used is the unsteady incompressible Navier-Stokes solver called **IncNavierStokesSolver** for which tutorials, examples and full lists of solver choices may be found on **http://www.nektar.info/wiki/3.3/UserGuide/Tutorial/IncNavierStokesSolver**.

Solver Specifics: Below is a list of the solver specific choices used for all simulations for easy reference.

- **Basis functions:** Choice = MODIFIED.
  This choice corresponds to a modal basis instead of a nodal basis.
- **Projection Type:** Choice = Continuous.
  The Continuous Galerkin approach is used for the spatial discretization.
- **Expansion Order:** Choice = 10.
  Tenth order polynomials were used for the local polynomial expansion.
- **Equation Type:** Choice = UnsteadyNavierStokes.
  This choice corresponds to solving the full unsteady incompressible Navier-Stokes equation.
- **Solver Type:** Choice = VelocityCorrectionScheme.
  Choosing the velocity correction scheme as the method for solving the equation.
- **Evolution Operator:** Choice = Nonlinear.
  The way the non-linear convection/advection term is handled.
- **Time Integration:** Choice = IMEXOrder2.
  The second order implicit-explicit scheme for time stepping.

Using IncNavierStokesSolver: All **Nektar++** solvers are constructed to take a single **.xml** file as the only input argument. This is the file described in the pre processing step containing all information about the simulation. To execute the solver on a single processor in a Linux terminal use the following command,

```
"/path/to/nektar++"/builds/solvers/IncNavierStokesSolver/IncNavierStokesSolver Input.xml
```

To execute the solver in parallel on X processors using MPI use the following command,

```
mpirun -n X "/path/to/nektar++"/builds/solvers/IncNavierStokesSolver/IncNavierStokesSolver Input.xml
```

4.3 Post Processing

Software to create the mesh and perform the CFD-simulations are now in place and the velocity field, in the x- and y-direction and pressure, (u, v, p), on a domain of interest may be obtained. The next step is, given the fields (u, v, p), to perform the necessary calculations to obtain desired quantities of interest. This requires a number of post processing steps which must be performed for all simulations. In order to perform these steps efficiently a series of programs and scripts have been written in **Shell Script**, **C++** and **Python** utilizing the open source program **Paraview**'s[19] python module. In addition several post processing programs provided with **Nektar++** have been put to good use. The final analysis of the postprocessed data are performed in **Paraview**'s GUI environment.

Here follows an explanation of the programs and scripts written for post processing. The full scripts, including details on their usage, are included in appendix A.5.

Shell Scripts

A series of Linux based shell scripts[20] were written to speed up and automatize data processing. Below, each script is listed and a short description is provided. The full scripts are provided in appendix A.5.1.

- **FullDataTreatment.sh**: This shell script executes all other shell scripts in the right order to perform all post processing calculations and organise the output.
- **NektarPP.sh**: Processes a series of **.chk** data files from a simulation to compute the following:
  - **Vorticity:** The shell scrip calls the **Nektar++** program **CalcVorticity** to calculate the vorticity of the velocity field $\mathbf{u} = (u, v)$.
  - **Vorticity Gradient:** The shell scrip calls the **Nektar++** program **CalcGrad** to calculate the gradient of the vorticity, i.e. ω_x and ω_y.
  - **Vorticity Hessian:** The shell scrip calls the **Nektar++** program **CalcGrad** twice to calculate the Hessian of the vorticity, i.e. $\omega_{x,x}$, $\omega_{x,y}$, $\omega_{y,x}$, and $\omega_{y,y}$.

  Finally the **Nektar++** function **FldToVtk** is called to convert the resulting **.fld** files to **.vtu** (XML) files for post processing with **Paraview**.
- **DomainCropping.sh**: The script calls the **C++** program **DomainCropper** for each **.vtu** data file. As the name suggests **DomainCropper** crops the domain to only include the area which contains data of interest to the analysis.
- **ZeroContour_CriticalPoints.sh**: Processes a series of **.vtu** data files. For each file the python script **ContourAndCriticalPoints.py** is called to calculate the nullclines of ω_x and the critical points for the vorticity, $\omega_x = \omega_y = 0$.
- **VorticityExtremaType.sh**: Processes a series of **.vtu** data files. For each file the python script **CriticalPointIdentification.py** is called to calculate the type of each of the critical points.

[19] A reasonable manual for **Paraview** may be found at [6].

[20] Shell Scripts are programs which may be executed directly in a Linux terminal.

- **CriticalPointTrace.sh**: Processes a series of **.vtu** data files. For a batch of **.vtu** files the python script **VTUFormatPointDataCombiner.py** is called to combine all critical points in one file to create a trace of the critical points position in time.

Python Scripts

The python scripts utilizes the **Paraview** python library with access to much of the programs GUI functionality. This means that much of the data manipulation can be performed efficiently by utilizing **Paraview**s code base. All scripts along with descriptions are provided in appendix A.5.2.

All python scripts assume that the user has installed the ***argparse*** *package and* ***Paraview*** *and has enabled access to the associated python libraries*[21]*.*

- **ContourAndCriticalPoints.py**: This script processes a **.vtu** file and extracts the coordinates and field information along the $\omega_x = 0$ contour and the critical points for vorticity i.e. $\{(x, y) \mid \omega_x = 0, \omega_y = 0\}$. This is done using the following steps,
  - Identify the $\omega_x = 0$ contour using **Paraview**'s **Contour** function.
  - Mark the part of the contours with a vorticity above the tolerance: $Q(\mathbf{r}|_{\omega_x=0}) > tol$ using **Paraview**'s **ExtractSelection** function.
  - Save the contour and marking information to a **.vtp** file.
  - Identify critical points for the vorticity along the $\omega_x = 0$ contour with $Q(\mathbf{r}|_{\omega_x=0}) > tol$. This is done using the **Contour** function this time locating $\omega_y = 0$.
  - Save the critical point information to a **.vtp** file.
- **CriticalPointIdentification.py**: The script processes a **.vtp** file containing the critical points for vorticity to determine their type. This is done by utilizing the eigenvalue approach described in section 2.1.2, in the following steps,
  - Calculate one eigenvalue λ_1 with **Paraview**'s **Calculator** function.
  - Calculate the eigenvalue product $\lambda_1\lambda_2$ with **Paraview**'s **Calculator** function.
  - Identify extrema by identifying $\lambda_1\lambda_2 > 0$ using **Paraview**'s **ExtractSelection** function.
  - Extract minima from the extrema by identifying $\lambda_1 > 0$ using **Paraview**'s **ExtractSelection** function.
  - Save the field data and coordinates of the minima to a **.vtu** file.
  - Extract maxima from the extrema by identifying $\lambda_1 < 0$ using **Paraview**'s **ExtractSelection** function.
  - Save the field data and coordinates of the maxima to a **.vtu** file.
  - Identify saddles by identifying $\lambda_1\lambda_2 < 0$ using **Paraview**'s **ExtractSelection** function.
  - Save the field data at and coordinates of the saddle points to a **.vtu** file.

[21] Explanation of the setup process may be found at: `http://paraview.org/Wiki/ParaView/Python_Scripting`.

- **VTUFormatPointDataCombiner.py**: This script processes a batch of **.vtu** files by combining all field information to a single **.vtu** file. This is used to create a trace of the critical points identified with **CriticalPointIdentification.py**. The script makes use of the **argparse**, **xml**, **copy** and **array** python packages.

C++ Program

A **C++** program has been written to crop the full domain of the simulation to reduce its size. As explained in section 3.2 the open flow simulations demand a huge domain to assure that far field BC's do not introduce significant errors in the calculations. For the post processing however the huge domain introduces a lot of unnecessary calculations. Therefore this program has been written to remove the uninteresting part of the domain thus saving significant computational resources, storage space and time.

The **C++** program is called **DomainCropper** and the source code and compilation instructions may be found in A.5.3.

The program takes seven input parameters. These are,

- Six floating point numbers specifying lower and upper bounds on the (x, y, z) coordinates of the domain.
- the name of the file to be cropped. The file must be in **.vtu** format.

The program loops over all elements of the mesh given in the **.vtu** file. If none of the points in the element lie within the bounds specified by the first six input parameters the element is eliminated. If just a single point lies within the bounds the element is kept. This ensures that no data within the area of interest are lost.

By utilizing the software described in the previous sections the post processing may be performed in a single step by organising the simulation data and post processing files as described in the following section.

4.3.1 The Finished Simulation Package

In order to utilize as much automation in the post processing steps as possible each simulation is organised in folders as illustrated on figure 4.1 and explained below.

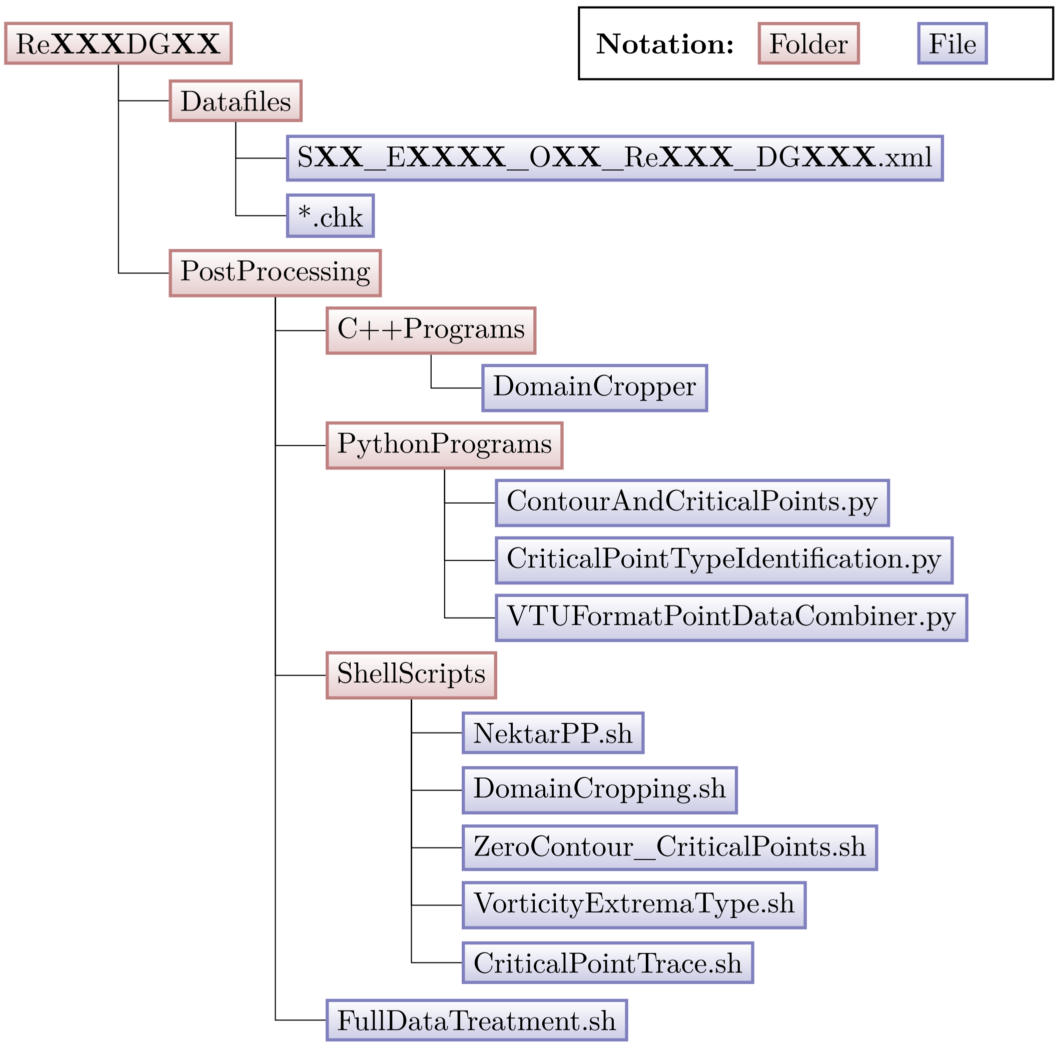


Figure 4.1: Illustration of the folder and file structure for the post processing after a sequence of output files: ***.chk** has been obtained.

Top Folder: Named by the Reynolds number and D/G ratio in the simulation: e.g. Re300DG10.

- **Datafiles:** Containing a **Nektar++ .xml** simulation file and a sequence of corresponding ***.chk** files with the output from **IncNavierStokesSolver** at a sequence of time steps.
- **PostProcessing:** All post processing material presented in this section positioned to allow the execution of *FullDataThreatment.sh* from a Linux terminal to perform the full process. In order perform the post processing enter this folder in a Linux shell and execute: `./FullDataTreatment.sh`

4.3.2 UQ Software

The stochastic collocation method for uncertainty quantification described in section 2.4.4 has been implemented using a number of MATLAB functions. The implementation have been done for applications using either a single uniformly or normally distributed underlying random variable. A short description of the function written for the implementation is provided below while all code with comments is included in appendix A.5.5. In addition a single example script where the method is applied is described briefly below and also included in the appendix.

Auxiliary Functions: All functions written to apply the SCM.

- **PnG:** This function evaluates and returns a gPC expansion based on a desired type of polynomials at a set of points x using pre calculated expansion coefficients, *tilde_g*.
- **PnG_Coefs:** The function evaluates and returns the expansion coefficients, *tilde_g* for a gPC expansion using the interpolation method based on a set of collocation nodes and solution data at the nodes.
- **LegendreP_UnNormalized:** Function which evaluates and returns the Nth Legendre polynomial at the points x along with the associated normalization factor g_N.
- **HermiteP_UnNormalized:** This function evaluates and returns the Nth Hermite polynomial at the points x along with the associated normalization factor g_N.
- **JacobiGQ:** Function which calculates the Gauss quadrature points x and associated weights w of the Nth Jacobi polynomial $J_N^{(\alpha,\beta)}$ for use in performing numerical quadrature. This function is taken directly from [14]. The function is used for calculating quadrature points when the underlying random variable is assumed uniformly distributed.
- **GeneralGaussianQuadrature:** Function implementing the Golub-Welsch algorithm, [21, Section 4.6.2], for calculating Gaussian quadrature points x and associated weights w for the class of orthogonal polynomials satisfying the three term recursion relation $p_{n+1}(x) + (B_n - x)p_n(x) + A_n p_{n-1} = 0, n = \{1, 2, ...\}$. The function is used for calculating quadrature points when the underlying random variable is assumed normally distributed.

Script: A simple sample script where the method is applied.

- **SCMexecutionNormal**: The SCM based UQ method is applied assuming a uniformly distributed random variable, X, using a fourth order expansion. The collocation points have been calculated using **JacobiGQ** and a coordinate transform. The desired quantity gX have been calculated from a simulation performed at the transformed collocation points. The expansion coefficients are calculated. Based on the coefficients the mean and variance are calculated. Finally the resulting gPC expansion, $P_{N,D}[g](X)$ is plotted along with mean plus and minus one standard deviation.

4.4 Hardware

Two systems were used to perform the simulations. The first system is the authors own laptop which has the specifications listed here,

- Memory: 2 x 8GB DDR3-1600 RAM DualChannel
- Processor: 8 x Intel Core i7-3610QM CPU @ 2.30GHz
- Operating System: Ubuntu 12.04.1 LTS 64-bit

On this system it is possible to perform a full simulation (initial transient and one full periodic simulation) in between 8 hours and 24 hours (depending on duration for the transient to disappear). The post processing for a full period is performed in less then 1 hour.

The second system used is the HPC cluster at the Technical University of Denmark (DTU) [22]. The specifications of this system are listed below,

- Number of nodes: 64 x HP ProLiant SL2x170z G6 nodes
- Memory: 24 GB memory per node
- Processor: 2 x Intel Xeon Processor X5550 (quad-core, 2.66 GHz, 8MB L3 Cache)
- Memory Bandwidth: QDR Inifiniband interconnect

The time scale for a full simulation on this system is roughly equivalent to that of the laptop, however this system allows for many simulations to run in parallel.

5 Parallel Execution with MPI

Nektar++ supports parallel execution for its solvers using the MPI[22] standard [23]. This section presents a series of tests of **Nektar++**'s MPI capabilities with regards to its incompressible Navier-Stokes solver **IncNavierStokesSolver**. The tests were performed in order to determine whether or not parallel processing would be favourable.

Before any testing is performed there are some general points to consider. One important factor to consider is that the system on which the simulations are performed, see section 4.4, have a limited number of nodes and a significant number of users. A number of important points which should be considered before executing in parallel using MPI are listed below.

- **Pro:**
  - Possible to speed up calculations by a factor up to the number of processors used.
- **Con's:**
  - Problematic to get time on the machines when requesting multiple processors due to queueing system.
  - Fewer independent simulations with different parameters can run at the same time due to limited processing resources.
  - Output from MPI execution may require additional post processing due to a segmented output process.

Based on the above considerations an MPI-based approach seems of limited interest already from the start. Thus using MPI for parallel processing would only be considered if the temporal scaling of the solution process with the number of processors was found to be very good.

Testing: Two different test series were run to investigate the scaling when using MPI for parallel solutions of the problem. The first was solving the incompressible NS equation on a simple square domain using an unstructured mesh with an inflow condition on one side, no-slip wall conditions on two sides and an outflow boundary condition on the last side. This test was designed to be as friendly to the solver as possible since partitioning a square domain across multiple processors should intuitively be straightforward.

The second test was solving the incompressible NS equation on a domain for the problem of the cylinder near a moving wall, see section 1.2, on the hybrid mesh presented in section 3.2. Here the introduction of the cylinder in the geometry and usage of different types of elements could have the potential of creating a problem for any mesh partitioning algorithm used by **Nektar++**.

Both tests are strong scaling tests which mean that the problem size is kept fixed while the number of processors used to solve the problem is increased. This choice is made since the problem of interest has a fixed size and thus the goal of parallelization is to obtain the solution to the problem faster and not to be able to solve bigger problems.

[22]MPI is an abbreviation for Message Passing Interface, and is a standard for parallel processing across multiple nodes/machines.

General Findings: To the authors surprise it was found that only limited decreases in solution time could be obtained when executing using MPI compared to standard serial execution. In fact, as will be illustrated later in this chapter, it was necessary to use more then eight processors to obtain better timings than when using a single processor.

It turned out that this behaviour has a good explanation however. During a talk with the lead developer of **Nektar++**, *Professor Spencer Sherwin*, the author learned that **Nektar++** by default uses a direct solver for time stepping the NS-equation. However when solving the system in parallel it becomes necessary to use an iterative method. According to *Professor S. Sherwin* the implementation of the iterative solution method in **Nektar++**[23] suffers from inefficient pre conditioners. This may atleast in part be the reason for the poorer performance of the iterative solver compared to the direct solver. Additionally the convergence criterion in **Nektar++** for the iterative solver is by default set very strict at $\| u_n - u_{n-1} \| \leq 10^{-10}$. This strict demand on convergence also leads to longer running times for the iterative method.

In summary it was found that the use of parallel execution actually slowed the solution process down considerably due to the difference in efficiency between the direct solver and iterative method implemented in **Nektar++**. However as is also shown below it was found that the parallel execution scales very well if one does not compare to the direct solver but instead use the iterative solver for a single processor.

As a side note it is mentioned that all solutions obtained using parallel execution were compared to the solution obtained using the direct solver on a single processor. The solutions were found to be identical and thus no problem with the correctness of the solution were introduced when solving in parallel. Information and test results for of the two cases are presented in the following.

Unstructured Square Mesh: An unstructured mesh with roughly 3500 triangular elements was created using **gmsh**. The incompressible NS-equation was solved using **Nektar++**'s solver with tenth order basis functions on each element.

The domain specifications are presented in figure 5.1

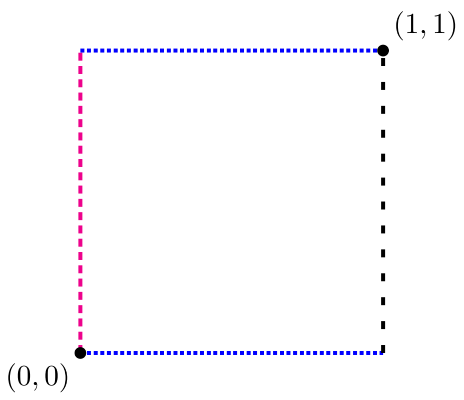


Figure 5.1: Illustration of the square mesh flow problem with BC's and domain size specified. (**WS**) Wall surface (densely dotted) no slip BC: (u,v) = (0,0). (**IF**) Inflow (densely dashed): (u,v) = (1,0). **(OF) Outflow (loosely dashed)** $\frac{d}{dx}(u,v) = \mathbf{0}$.

[23]The version of **nektar++** used here is version 3.3.0.

The system was solved for a given number of time steps and the wall clock time recorded.

Figure 5.2 shows how the solution time scales compared to the solution on a single processor as a function of the number of processors used. The scaling factor, F_{scale}, is calculated as,

$$F_{scale}(N) = \frac{T_1}{T_N}, \tag{5.1}$$

where T_1 is the wall clock execution time on a single processor while T_N is the wall clock execution time on N processors. For the present case the direct solver was used for the single processor solution.

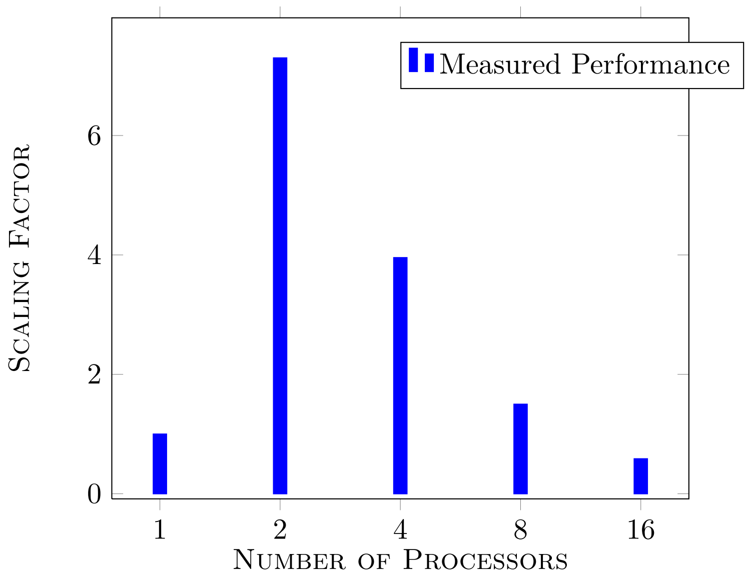


Figure 5.2: The bar graph shows the scaling factor normalized by the solution time on a single processor. The solution time for one processor is $t_{\text{sol}} = 189$ minutes. The equations were solved using a time step of $t_{\text{step}} = 0.005$ with 25000 time steps.

From figure 5.2 it is seen that between eight and sixteen processors are needed in order for the parallel iterative solution to break even with the serial direct solution. This makes the parallel solution very inefficient in terms of the computational resources used to obtain the solution.

Figure 5.3 also shows how the solution time scales compared to a single processor as a function of the number of processors used. Here the iterative solver was used for the single processor as well as for the multiple processors. The blue bar shows the measured results while the red bar illustrates perfect scaling compared to a single processor.

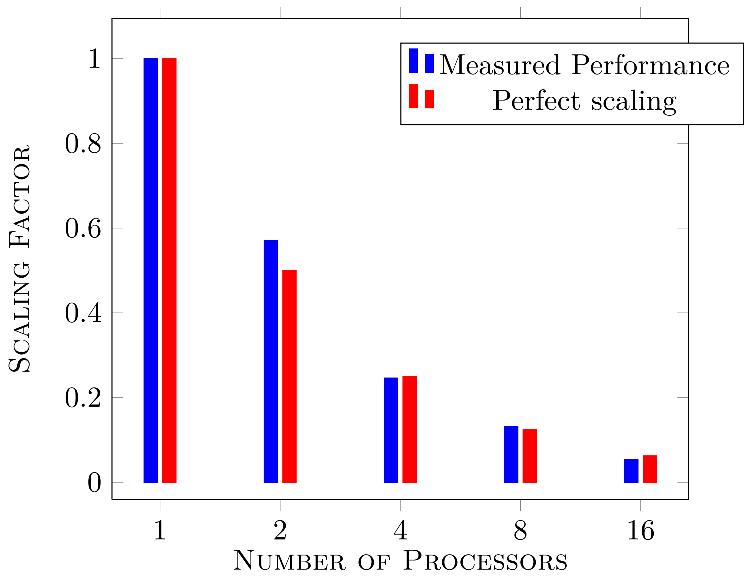


Figure 5.3: The bar graph shows the scaling factor normalized by the solution time on a single processor. The solution time for one processor is $t_{\text{sol}} = 516$ minutes. The equations were solved using a time step of $t_{\text{step}} = 0.005$ with 5000 time steps.

From figure 5.3 it is seen that the parallel solution scales very well from one to sixteen processors. If only the iterative solver was as efficient as the direct solver this would be a strong argument for utilizing parallelization in the solution process. The perfect scaling stated on figure 5.3 corresponds to the time used on n processors is $1/n$ times the time used on a single processor.

Hybrid Mesh for Cylinder/Wall Problem: For the second set of tests the hybrid mesh used for the model problem of the cylinder near the moving wall, as illustrated in section 3.2 with roughly 2500 elements has been used. The incompressible NS-equation was solved using twelfth order basis functions on each element over for a given number of time steps.

The domain and boundary condition information are presented in section 1.3.4.

The first test for the hybrid mesh was performed with the direct solver for the single processor and iterative solver for multiple processors. The results are presented in figure 5.4.

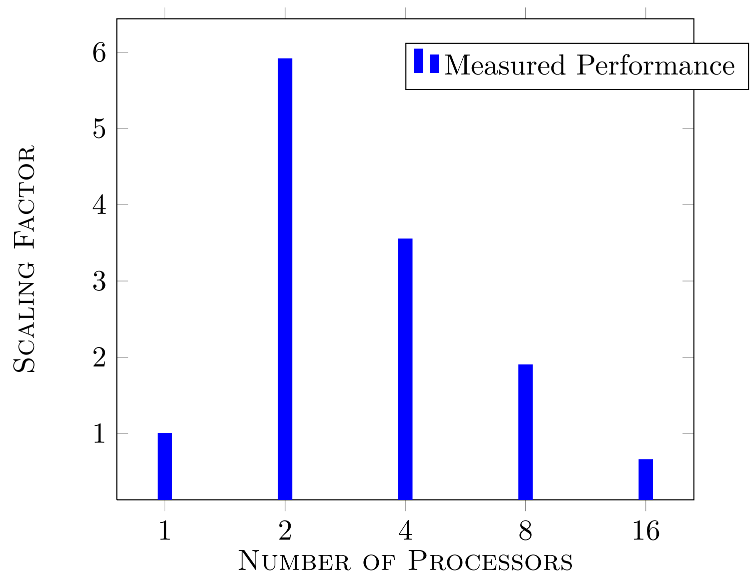


Figure 5.4: The bar graph shows the scaling factor normalized by the solution time on a single processor. The solution time for one CPU is $t_{\text{sol}} = 218$ minutes. The equation was solved over 20000 time steps of length $t_{\text{step}} = 0.0005$.

This test shows the same results as for the square mesh. I.e. for this case no performance decrease was found to be introduced by the hybrid mesh and more complicated geometry. At the same time the test still shows that the direct solver is faster then the iterative solver up until somewhere between eight and sixteen processors.

The second test for the hybrid mesh was as for the unstructured square mesh performed with the iterative solver for both the single and multiple processors. The results are presented in figure 5.5.

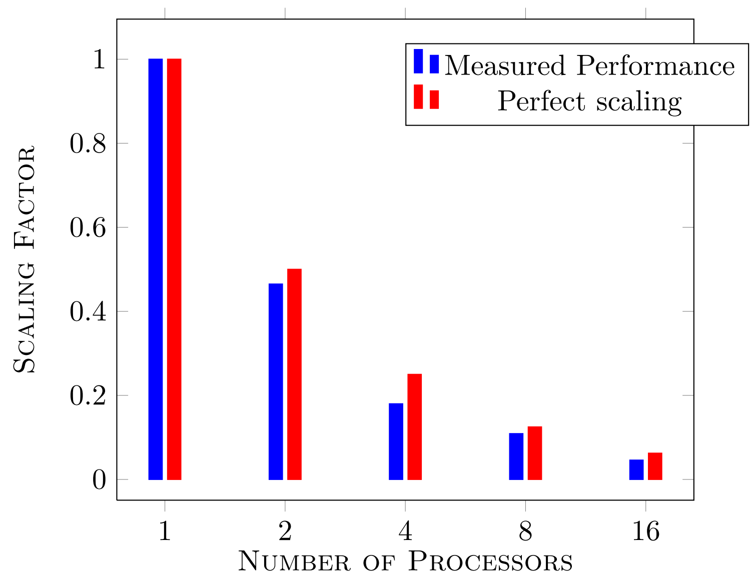


Figure 5.5: The bar graph shows the scaling factor normalized by the solution time on a single processor. The solution time for one CPU is $t_{\text{sol}} = 770$ minutes. The equation was solved over 5000 time steps of length $t_{\text{step}} = 0.0005$.

Again the test results presented in figure 5.5 show that using the iterative solver the parallel solutions scale very well. In fact it is observed that the scaling is better than perfect. Why this is the author is not entirely sure, however it may have to do with overhead associated with utilizing MPI for parallelization.

As a final test the tolerance used for the iterative solver to assure convergence was modified from 10^{-10} to 10^{-6}. This was done in the hope of achieving better scaling results for the iterative method compared to the direct solver. It was investigated and found that this change in tolerance did not impact the solution quality for this particular problem.

The results of this test are presented in figure 5.6.

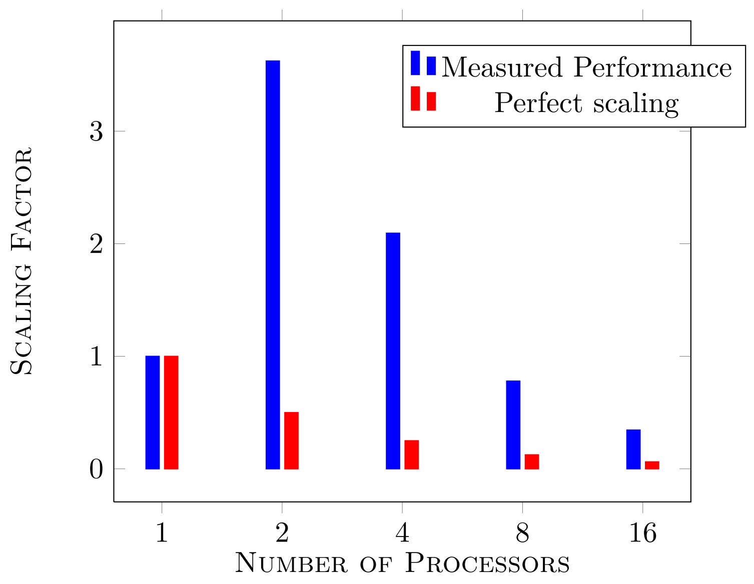


Figure 5.6: The solution time for one CPU is $t_{\text{sol}} = 77$ minutes. The equation was solved over 7500 time steps of length $t_{\text{step}} = 0.0005$.

The results presented in figure 5.6 show that changing the tolerance leads to better performance for the iterative solver. Now only between four and eight processors are needed for the parallel solver to outperform the serial solver. However the performance of the iterative solver is still poor compared to the direct solver.

Conclusion: Based on the MPI-testing presented above it was decided to run a large number of serial simulations for different parameter values simultaneously instead of using MPI for parallelizing the individual simulations. This manual parallelization assured that the largest number of simulations could be performed in a given period of time.

6 Simulations

This chapter provides a brief overview of the simulations performed throughout the project, an illustration of how the post processing to identify critical points for the vorticity is performed and how the data visualization is done. The analysis and results for all simulations listed here are presented in the following chapters.

Transient and Periodic Stages: For all parameter values of interest in this project it was found that the solutions to the model problems are either stationary or periodic in nature after any initial transient solution has been allowed to die out.

As a consequence of this, all simulations of the cylinder near the wall are performed in two stages. The first stage is a long simulation which begins from a set of artificial initial conditions. It ends when the initial transient solution has died out and either periodic shedding or a stationary flow has been reached. The artificial initial conditions are given by,

$$(u, v, p) = (1, 0, 0) \quad \forall \mathbf{r} \in \Omega, \tag{6.1}$$

and were chosen to fit the boundary conditions on the domain as well as possible. See section 1.3.4 for a presentation of initial and boundary conditions.

After the first stage of the simulation is done the field data for u, v and p at the final time step is stored and used as initial conditions for the second stage. The second stage is performed to obtain a large number[24] of datasets covering more then one period of vortex shedding. These datasets are then used for post processing and analysis.

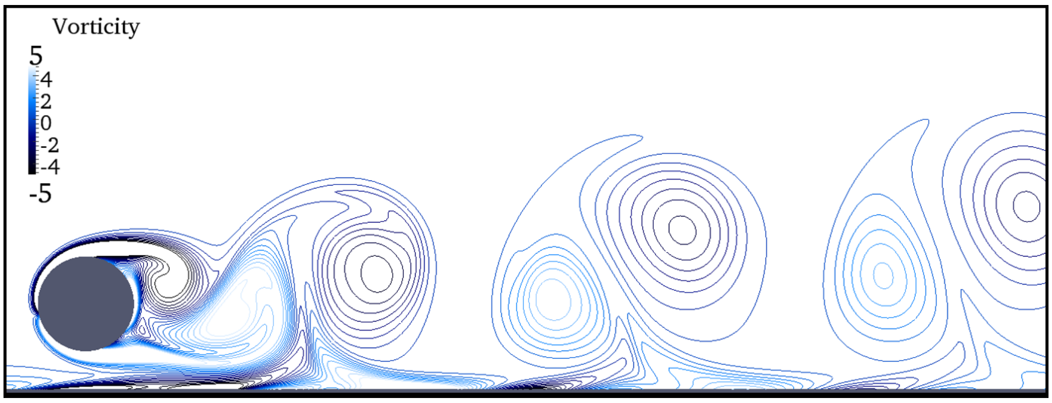
All simulations have been performed using the stable time step $\Delta t = 5 \cdot 10^{-4}$, identified in section 3.1. The time needed to kill the initial transient solution and reach perfect periodic shedding was found to be between 40 time units and 160 time units for almost all simulations. A few exceptions were found which will be discussed later.

Dimensions: For all simulations for the cylinder near the moving wall the domain dimensions simulated are given by $(x, y) \in [-10, 30] \times [0, 20]$. For the cylinder in free flow the domain dimensions used are $(x, y) \in [-10, 30] \times [-15, 15]$. The diameter of the cylinder for all simulations have been chosen as $D = 1$.

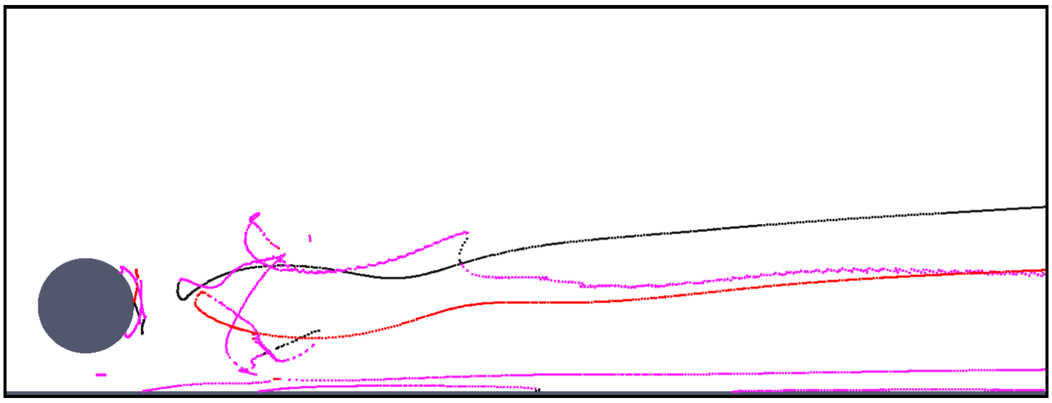
[24]The number of datasets are between 200 and 300 per simulation to ensure good temporal resolution.

6.1 Visualization

In order to ease the understanding of the data visualization presented in later chapters a short guide is given here. In order to visualize the creation and movement patterns of vortices in a consistent and understandable manner the following general choices have been made.



(a)



(b)

Figure 6.1: Example of visualization of vortex flow structure using $Re = 240$ and $D/G = 5/2$. **(a)** Vorticity contours with dark blue and black contours showing clockwise rotating vortices and light blue and white contours showing counter clockwise rotating vortices. **(b)** The path followed by the stationary points in vorticity over time. Here magenta marks **(Saddles)**, black marks **(Minima)** or clockwise rotating vortices, and orange marks **(Maxima)** or counter clockwise rotating vortices.

As explained in section 3.2 the simulations were performed over a large domain to minimize blockage introduced by the far field BC's. The visualizations however are only done for a limited part of the domain as this is where interesting flow patters are observed. For all plots of the cylinder/wall system white corresponds to the domain and dark gray corresponds to the wall and cylinder.

In general two types of plots of the domain are presented.

The first type illustrates the vortices by a contour plot at a fixed point in time along with a color bar showing the magnitude and rotational direction of the vortices. For this type of illustration a black-blue-white scheme has been chosen to color the contours which illustrate strength and direction of rotation for the vortices. An example of the first type of plot is provided in figure 6.1a.

The second type illustrates a trace over time of the critical points for vorticity which for minima and maxima corresponds to vortices rotating clockwise and counter clockwise respectively. For the second type of plots the position of the vorticity extrema as a function of time is illustrated by traces of small dots. To distinguish the different type of extrema the vorticity maxima (counter clockwise rotating vortices) are coloured orange, the minima (clockwise rotation vortices) black and the saddles magenta. An example of the second type of plot is provided in figure 6.1b.

For some purposes the two types of visualizations are plotted together and for most illustrations extrema and saddle point paths irrelevant to the current analysis have been edited away for clarity.

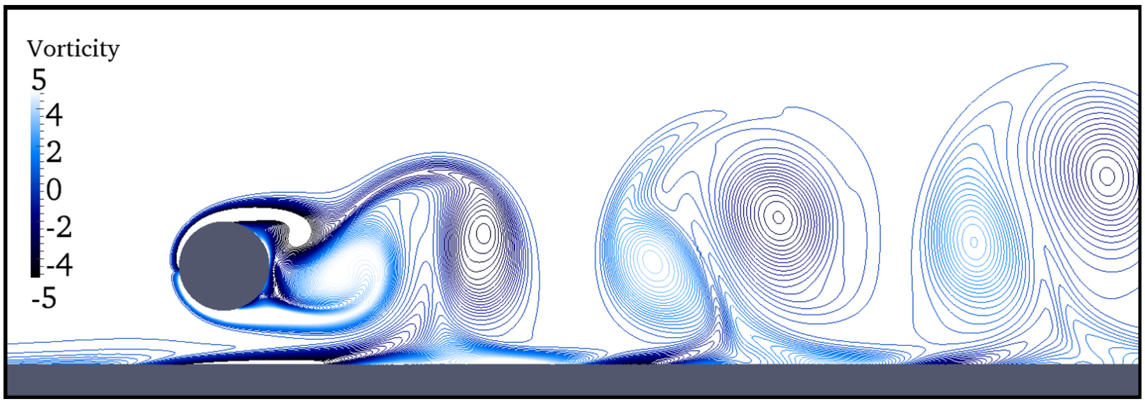
6.2 Critical Point Identification

For the investigation of the vortex creation, annihilation and movement patterns the critical points of the dynamical system given in (2.5) must be identified and categorized. This is done during post processing and the process is explained below.

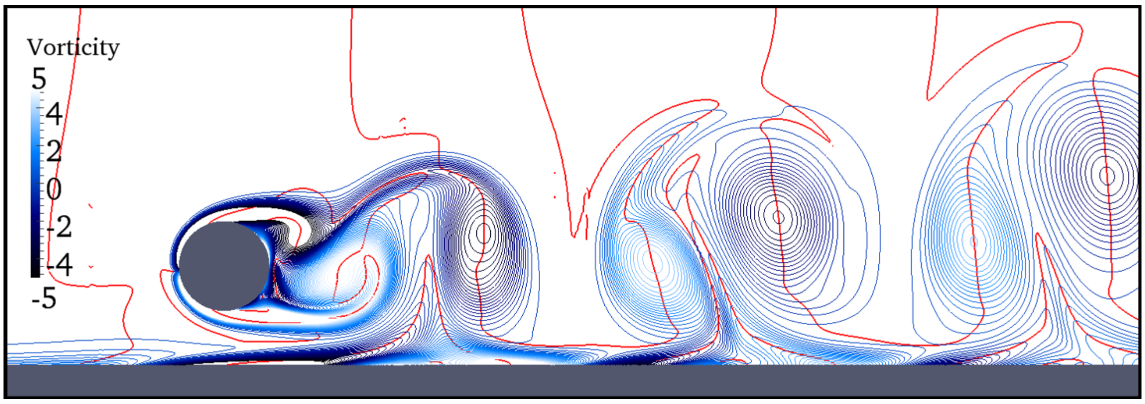
The figures 6.2 and 6.3 visualises the steps in the automated identification and classification of the extrema and saddles of the vorticity. This is based on the procedure given in Algorithm 1. The procedure is explained in six steps below.

- **Step 0:** Data at every time step for a simulation for which all needed fields have been calculated is chosen. As an example contours for the vorticity at a given time step is shown in figure 6.2a.
- **Step 1:** For each time step the ω_x nullclines are calculated, as visualised on figure 6.2b by the red contours.
- **Step 2:** In areas of very low vorticity, $\omega \ll 1$, small numerical errors in the simulations and calculation of ω_x introduce erroneous data in the $\omega_x = 0$ contours. In order to avoid this problem a filter has been introduced which removes parts of the ω_x nullclines that fulfil the relationship: $\omega < 10^{-5} \cdot \max |\omega|$. This may be done safely as vortices which are five decades smaller then the strongest vortices in the system are of no interest to the further analysis. The step is illustrated in figure 6.2c.
- **Step 3:** After the filtering, the points along the remaining ω_x nullclines which fulfil $\omega_y = 0$ are calculated. This identification of $\{(x_i, y_i)|\omega_y = \omega_x = 0\}$ is illustrated in figure 6.3a by showing the full ω_y nullclines in green[25].
- **Step 4:** Each of the points $\{(x_i, y_i)|\omega_x = \omega_y = 0\}$ are now investigated to identify whether they are **(maxima)**, **(minima)** or **(saddles)**. See figure 6.3b.
- **Step 5:** The procedure is repeated for each time step and the data is stored for analysis.

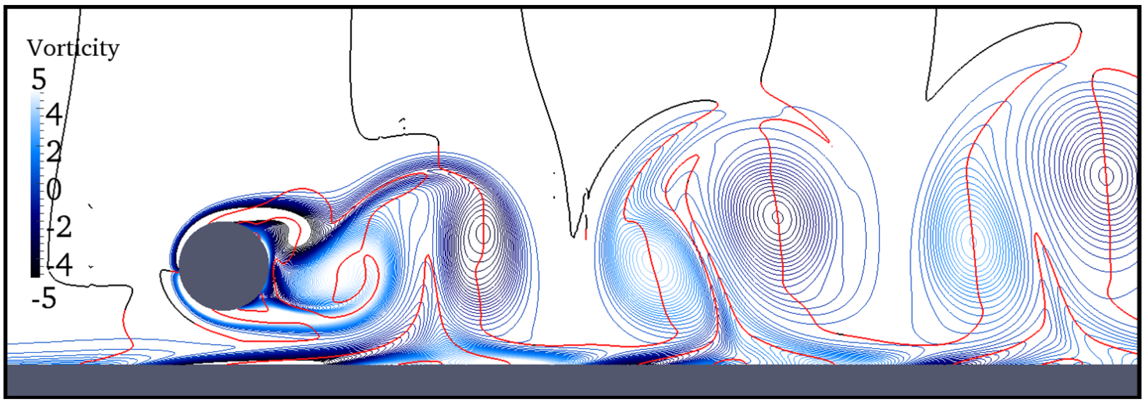
[25]This way of illustration makes it easier to see the intersections where $\omega_x = \omega_y = 0$.



(a)

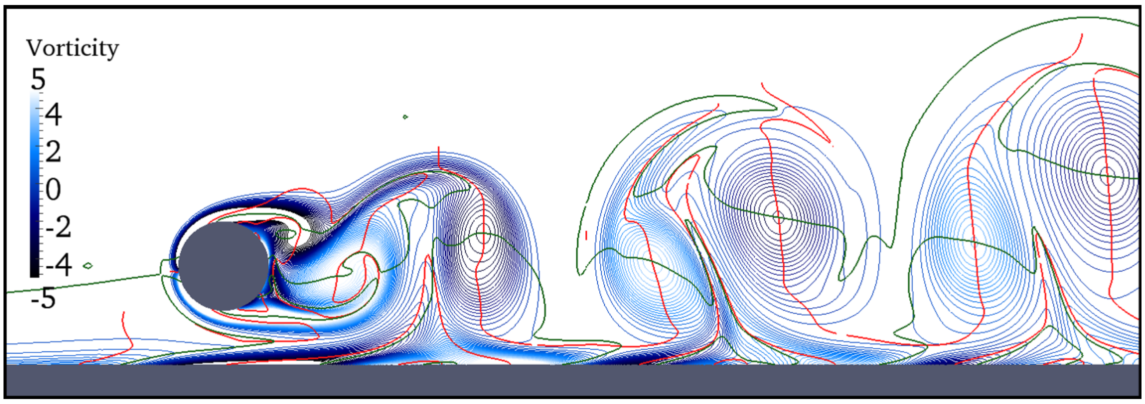


(b)

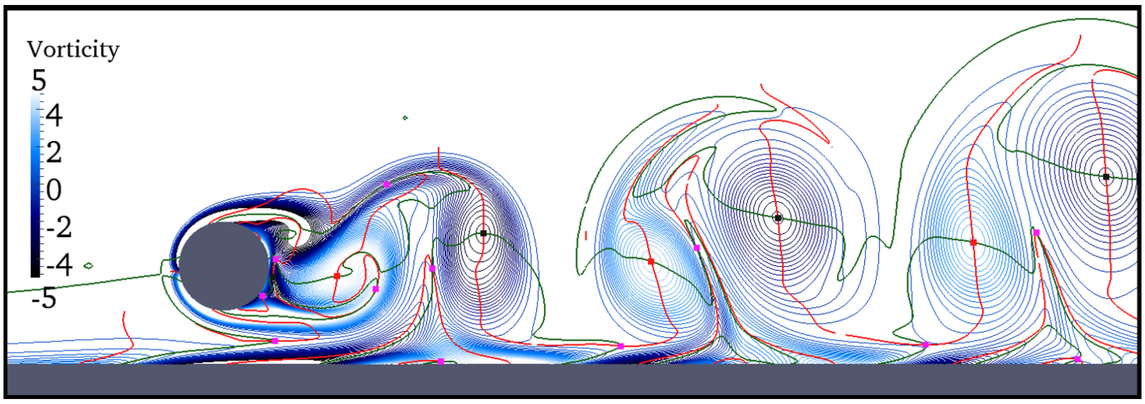


(c)

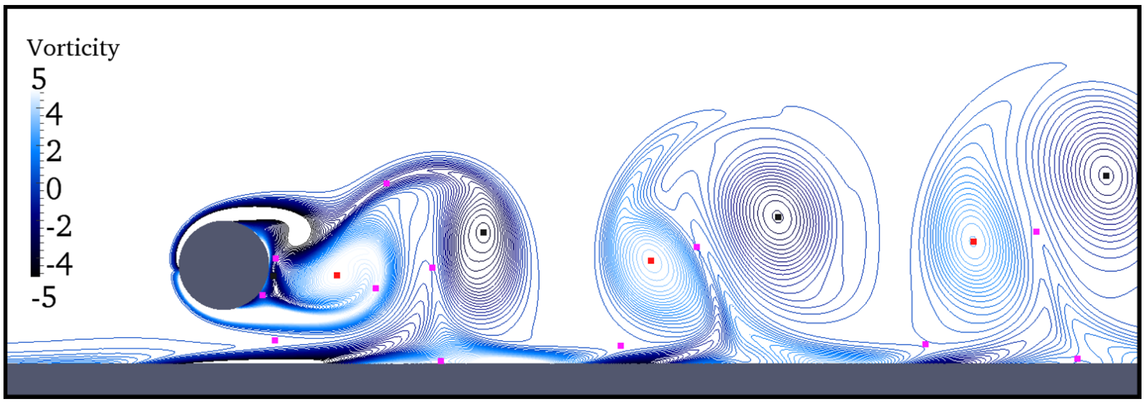
Figure 6.2: Illustration of (a) the vorticity using black-blue-white contours, (b) the $\omega_x = 0$ contour, coloured **(red)**, (c) filtering of the $\omega_x = 0$ contour by the requirement: Remove any part of a contour for which $\omega < 10^{-5} \cdot \max|\omega|$, i.e. the part of the contours coloured black in the figure are removed.



(a)



(b)



(c)

Figure 6.3: Illustrating (a) the identification of the points $\omega_y|_{\omega_x=0} = 0$, (b) The type of critical points, **(maxima)**, **(minima)** or **(saddles)**, (c) Only the critical points without the contours.

6.3 Validation Simulations

The first series of simulations performed were made in order to validate that results obtained using the **Nektar++** framework are trustworthy. These simulations may be divided into four parts.

- Validating the **Nektar++** framework. See appendix A.4 for results.
- Identifying a sufficient domain resolution in terms of element number and polynomial expansion order to assure that the solution has converged. See section 7.1.
- Validating simulation data against published results. See section 7.2.
- Assuring that the stationary/periodic regime is reached for a given simulation. See section 7.3.

6.4 Cylinder near the Moving Wall

After the validation process was completed a large number of simulations were performed for the cylinder near the moving wall. As detailed in section 1.2 this problem has three parameters of interest. These are time, t, the Reynolds number, Re, and the relation between the cylinder diameter, D, and the distance between the cylinder and wall, G, i.e. D/G. Obviously time is not a parameters which is varied like the Reynolds number or D/G. Instead after periodic shedding has been achieved, time is allowed to run over more then one full period of shedding and the period noted for each simulation. Then for each simulation, data about the flow pressure, velocity, vorticity and the first and second derivatives of the vorticity are stored at a sequence of time steps in order to allow investigation of the flow in time.

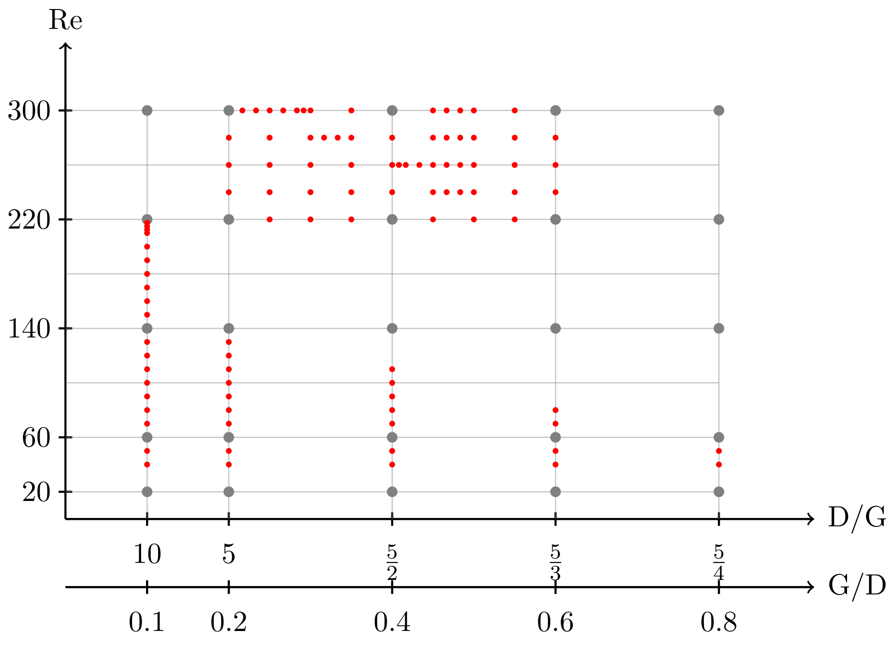


Figure 6.4: Illustration of the 2 parameter space $(Re, D/G)$ with parameter values at which simulations were performed marked by grey and red dots. Note that the scaling of the D/G axis is linear if we consider the quantity G/D i.e. the gab distance for a cylinder with $D = 1$.

Parameter Domain: The investigation of the structure of the vortices etc, is done over the range $(Re \times D/G) \in (\{20, 300\} \times \{10, \frac{5}{4}\})$. This forms a 2 dimensional parameters space which is illustrated in figure 6.4.

The illustration in figure 6.4 shows the full parameter space along with all points at which a simulation has been performed. The gray large circular points mark the initial simulations performed to gain an overview of the flow structures in the domain of interest. The smaller red points are all the simulations performed to investigate phenomena identified during the initial simulations.

6.5 Uncertainty Quantification

Three separate UQ investigations have been performed. In each case the Reynolds number was taken to be the uncertain parameter, $X = Re$.

Cylinder in Free flow: The first two investigations were performed for a cylinder in free flow with focus on comparing results obtained using UQ to known published results and to results obtained during simulations for this work. The quantities of interest here are St and C_{D_p}. One investigation was performed assuming that the Re was uniformly distributed in the interval $Re \in \mathcal{U}[50, 600]$. The other was performed assuming a normal distribution for Re, $\mathcal{N}(300, 50)$.

Note: *Technically it is wrong to use normal distribution for the Re as, $Re \geq 0$ is dictated by the physics. However using $\mathcal{N}(300, 50)$ for Re yields a probability of obtaining an unphysical* *value, i.e. $Re < 0$, given by $\int_{-\infty}^{0} f_{Re,G}(x)dx \approx 10^{-9}$, which is deemed acceptable for the present test.*

For both investigations three different basis orders have been used for the gPC expansions. In both cases these are a first, a second and a fourth order expansion.

Cylinder Near Moving Wall: The third investigation was performed for the cylinder near the moving wall at the diameter to gap ratio $D/G = 10$. Here the application of UQ was used to perform an investigation of the variation in the path travelled by a vortex downstream of the cylinder depending on Re. Here the Reynolds number was again assumed to be uniformly distributed, this time on the interval $Re \in \mathcal{U}[150, 240]$. For this investigation three different basis orders were also used. In this case these are a third, a fifth and a seventh order basis.

7 Validation

For any investigation of a physical system using computer simulations it is very important to assure the results of the simulations may be trusted. The best way of doing this is of course to validate the simulations against real world experiments. It is however outside the scope of this thesis to make experiments which may validate the flow structures and physical quantities investigated. Therefore other methods of validations have been used. These are,

- *Validation of the **Nektar++** framework and incompressible Navier-Stokes solver by application to problems with known solutions.*
- *Validation of solution accuracy through convergence testing measured in L_2-norm.*
- *Validation against published work.*

Section 7.1 provides the results of a convergence test for the quantities used in the analysis process. Section 7.2 presents comparisons of results obtained in this project and published results. Finally appendix A.4 contains a set of tests of the **Nektar++** framework and its incompressible Navier-Stokes solver **IncNavierStokesSolver** validating it against known solutions. These tests were performed by the author prior to this project are may therefore be found in the appendix.

7.1 Convergence

Given that the problem to be solved is well posed the SEM promises that the numerical solution converge to a unique solution for the 2D-Incompressible Navier-Stokes problem as the number of elements and/or the polynomial expansion order are increased.

That the solution converge to a unique solution makes is possible to investigate when the domain is well enough resolved by observing changes in L_2 norm for each of the field quantities of interest. When the L_2 norm has converged to a fixed value this indicates that the solution has converged. In order to assure sufficient resolution a convergence tests with increasing polynomial order on the SEM basis functions and an increasing number of elements has been performed[26].

The tests were performed by simulating the flow using the boundary and initial conditions presented in section 1.3 from the initial time $t = 0$ to the final time $t = 40$ using a time step length of $\Delta t = 5 \cdot 10^{-4}$.

Table 7.1 contains the parameters used for polynomial order and number of elements in the mesh and the corresponding approximate number of DOF's for the system.

[26]The test has been performed for all fields of interest using the build in L_2-norm calculator in **Nektar++**.

Simulation number	Number of elements	Polynomial order	$\approx$ DOF
1	1500	6	54000
2	1500	8	96000
3	1500	10	150000
4	1800	6	64800
5	1800	8	115200
6	1800	10	180000
7	2400	6	86400
8	2400	8	153600
9	2400	10	240000
10	3000	6	108000
11	3000	8	192000
12	3000	10	300000

Table 7.1: Table displaying the number of elements and polynomial order used for each simulation in the convergence test. The Reynolds number used: $Re = 300$ and the gap to diameter ratio $D/G = 5$.

Tables 7.2 and 7.3 show the results of the tests.

	1	2	3	4	5	6
u:	**28.541**	28.540	28.540	28.541	28.540	28.540
v:	3.6269	3.6285	3.6282	**3.6282**	3.6282	3.6282
p:	2.8721	2.8726	2.8738	2.8756	**2.8730**	2.8733
ω:	**13.926**	13.925	13.925	13.924	13.925	13.925
ω_x:	119.09	118.88	118.75	118.80	**118.70**	118.71
ω_y:	191.47	**191.50**	191.51	191.62	191.51	191.51
$\omega_{x,x}$:	1990.1	1997.1	1997.1	1984.4	1985.1	1979.4
$\omega_{x,y}$:	1661.6	1657.0	1653.5	1658.2	1654.0	**1652.4**
$\omega_{y,x}$:	1661.6	1657.0	1653.5	1658.2	1654.0	**1652.4**
$\omega_{y,y}$:	4442.4	4335.3	**4329.6**	4370.4	4333.4	4329.7

Table 7.2: The L_2-norm calculations for each of the 10 fields of interest obtained from the first 6 simulations, for which parameters are specified in table 7.1. Bold entries highlight that the field has converged to four significant digits.

	7	8	9	10	11	12
u:	28.541	28.540	28.540	28.541	28.540	28.540
v:	3.6284	3.6282	3.6282	3.6282	3.6282	3.6282
p:	2.8726	2.8737	2.8732	2.8744	2.8732	2.8732
ω:	13.925	13.925	13.925	13.925	13.925	13.925
ω_x:	118.83	118.70	118.72	118.73	118.71	118.71
ω_y:	191.52	191.51	191.51	191.52	191.51	191.51
$\omega_{x,x}$:	1988.2	1985.6	1981.2	1983.7	**1981.2**	1981.2
$\omega_{x,y}$:	1657.5	1653.1	1652.4	1656.5	1652.8	1652.4
$\omega_{y,x}$:	1657.5	1653.1	1652.4	1656.5	1652.8	1652.4
$\omega_{y,y}$:	4363.4	4329.8	4329.9	4351.8	4329.9	4329.9

Table 7.3: The L_2-norm calculations for each of the 10 fields of interest obtained from the last 6 simulations, for which parameters are specified in table 7.1. Bold entries highlight that the field has converged to four significant digits.

The bold face values in the tables denote the lowest number of DOF's for which a given quantity has converged to within four significant digits when compared to simulations using more DOF's.

From table 7.2 it may be seen that already at simulation 2 (1800 elements, eights order basis) the velocity field and vorticity has converged to four significant digits. Thus for all simulations where these quantities are the only ones desired, no higher resolution seems to be needed.

It may be seen that the last of the fields of interest does not converge to four significant digits until a domain of 3000 elements with eight order basis functions are used, as seen in table 7.3.

To clarify, it was found that all fields of interest have converged to a stable value measured to the fourth significant digit for a mesh with 3000 elements using eight order basis functions with $Re = 300$ and $D/G = 5$. This result have been used as a basis for the choice of the minimum number of elements and polynomial order to be used in the following simulations.

Note, *an important finding made in the process of using the isocline method*[27] *for calculating vortex traces was that a better resolved domain was needed to obtain accurate traces of the vortices and saddle points.*

The finding during the application of the isocline method made it necessary to use a mesh with 2400 elements and 10th order basis functions to obtain accurate traces of the vortex paths.

7.2 Comparing Results

The next step is to validate that solutions obtained using the number of elements and order of basis functions found to be sufficient in the previous section actually produce accurate results. This has been validated through a series of tests comparing results obtained using 3000 elements and 8th order basis functions with previously published results.

In [8] Huang and Sung presents a series of results validating their simulations for a cylinder in free flow. The results are based on the comparison of Strouhal number, St, and base pressure

[27]See section 2.1.2 for the method and section 6.2 for a visualization.

coefficient, C_{bp}, with the result published by Henderson in [7] and [10]. In [8] a range of results with the moving wall near the cylinder are also presented. From these results the St number and C_{bp} coefficient have been extracted for comparison. In addition the pressure drag coefficient has been calculated and the result compared to results presented in [10].

It is important to mention that Henderson, [10], [7] compares his data to experimental data. Here he found that at $Re \approx 190$ the two dimensionality of the flow breaks down and the two dimensional simulations no longer agree with the experimental data for the cylinder in free flow. This fact does not prevent the investigation of agreement between simulation data presented in the articles and the data obtained in the present simulations. If agreement is seen this supports that the solutions to the model problem obtained with **Nektar++** can indeed be trusted.

7.2.1 Free Flow

For the cylinder in free flow simulations where performed using a set of Reynolds numbers listed in table 7.4.

Simulation number	Reynolds Number
1	100
2	200
3	300
4	400
5	500
6	600

Table 7.4: Table displaying the Reynolds number for each simulation.

For each simulation the Strouhal number, time averaged drag coefficient for pressure and time averaged base pressure coefficient were calculated.

Strouhal Number: The Strouhal number for each simulation along with results published in [10] are presented in table 7.5. The accuracy of the temporal resolution in the present simulations and accuracy of reading off results from a figure in [10] allows comparison to at most the third decimal point.

Re	St Present	St Article
100	0.168	0.168
200	0.198	0.197
300	0.213	0.212
400	0.221	0.220
500	0.225	0.226
600	0.229	0.229

Table 7.5: The St number obtain for a cylinder in free flow at varying Reynolds number in this work and in [10]. The simulation data is determined by measuring the shedding period with a time step of $t_{step} = 0.04$ between each frame. The article data is read of [10, figure 3]. The uncertainty in reading of the data from [10, figure 3] is estimated to be $\pm 0.5 \cdot 10^{-3}$.

By considering the data presented in the table it is seen that down to measurement uncertainty determined by the temporal coarseness of the simulations performed and the ability to read data from the figure presented in [8], there is excellent agreement between the previous published work and the present work.

Average Drag Pressure Coefficient: The time average of the pressure component of the drag coefficient over a period were calculated and compared to the result obtained by Henderson [7, figure 1]. Henderson provided a four parameter fit for his simulation data given by,

$$f(x) = a_0 - a_1 x^{a_2} \exp(a_3 x), \quad (a_0, a_1, a_2, a_3) = (1.4114, 0.2668, 0.1648, -3.375 \cdot 10^{-3}). \quad (7.1)$$

Figure 7.1 shows the fit along with the data obtained in the present work.

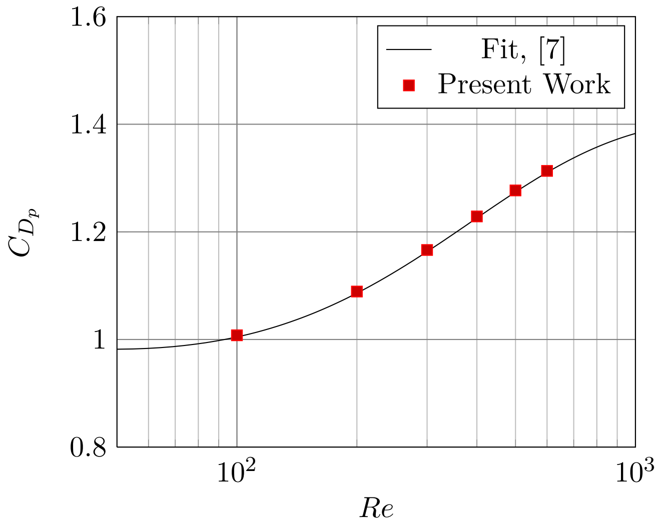


Figure 7.1: Illustration of fit for pressure drag coefficient $C_{D_p} = \frac{F_{D_p}}{\frac{1}{2}\rho U_\infty^2}$ averaged over a shedding period as a function of Reynolds number, [7]. The marks are the values obtained in the present work.

Table 7.6 shows the values obtained in the simulations along with the values obtained from the fit [7, figure 1].

Re	$\overline{C_{D_p}}$ Present	$\overline{C_{D_p}}$ Article
100	0.998	1.008
200	1.080	1.089
300	1.169	1.166
400	1.230	1.229
500	1.278	1.277
600	1.318	1.313

Table 7.6: The drag pressure coefficient calculated from the present simulations and obtained using (7.1). The uncertainty in the calculated value from the present simulations are estimated to lie on the last digit.

Just as for the Strouhal number the agreement between the two sets of results is seen to be excellent.

Average Base Pressure Coefficient: The average base pressure coefficient for each simulation were calculated and the results along with a fit calculated by Henderson [7] are presented in figure 7.2.

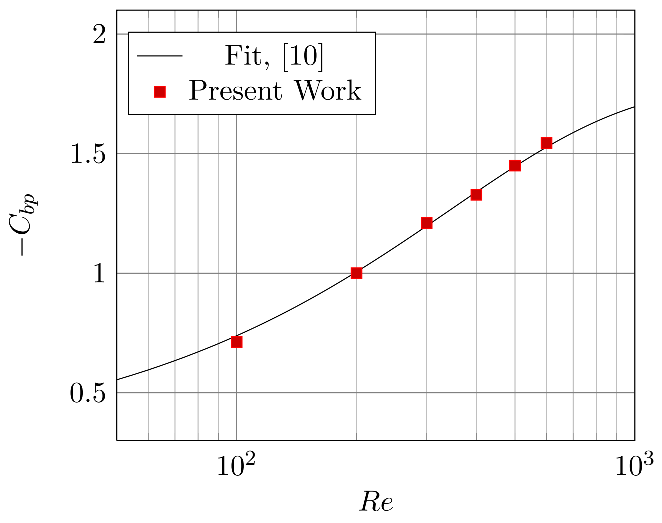


Figure 7.2: Illustration of fit for the base pressure coefficient $C_{bp} = \frac{p_b - p_\infty}{\frac{1}{2}\rho U_\infty^2}$ averaged over a single period as a function of Reynolds number, [7]. The marks are the values obtained in the present work.

The fit is based on simulation data obtained by Henderson and is given by,

$$f(x) = a_0 - a_1 x^{a_2} \exp(a_3 x), \quad (a_0, a_1, a_2, a_3) = (1.7826, 1.6575, -0.0427, -2.660 \cdot 10^{-3}). \tag{7.2}$$

Table 7.7 shows the values obtained in the simulations along with the values obtained from the fit [7, figure 1].

Re	$\overline{C_{bp}}$ Present	$\overline{C_{bp}}$ Article
100	-0.71 ± 0.02	-0.739
200	-1.00 ± 0.02	-1.006
300	-1.21 ± 0.02	-1.198
400	-1.33 ± 0.02	-1.339
500	-1.45 ± 0.02	-1.446
600	-1.54 ± 0.02	-1.527

Table 7.7: The calculated time averaged base pressure coefficient calculated from the fit (7.2) along with the one obtained in the present work.

The results obtained from the present simulations are in reasonable agreement with the previous results from [7] however they are seen to vary slightly more then those for St and

C_{D_p}. No satisfying reason for this variation has been identified but the variations are deemed small enough that they are not pursued further.

7.2.2 Cylinder Near Wall

A new set of tests were performed as the moving wall was introduced. Here the St and C_{bp} were calculated at four different values of D/G and compared to the same quantities derived from data presented in [8]. The way St was derived from [8] was by measuring the non-dimensional shedding frequency from the plot of drag and lift in [8, Figures 5-8]. C_{bp} was obtained from [8, Figure 14].

The D/G-values are $D/G \in \{0, 5/3, 5, 10\}$ corresponding to cylinder positions shown in figure 7.3.

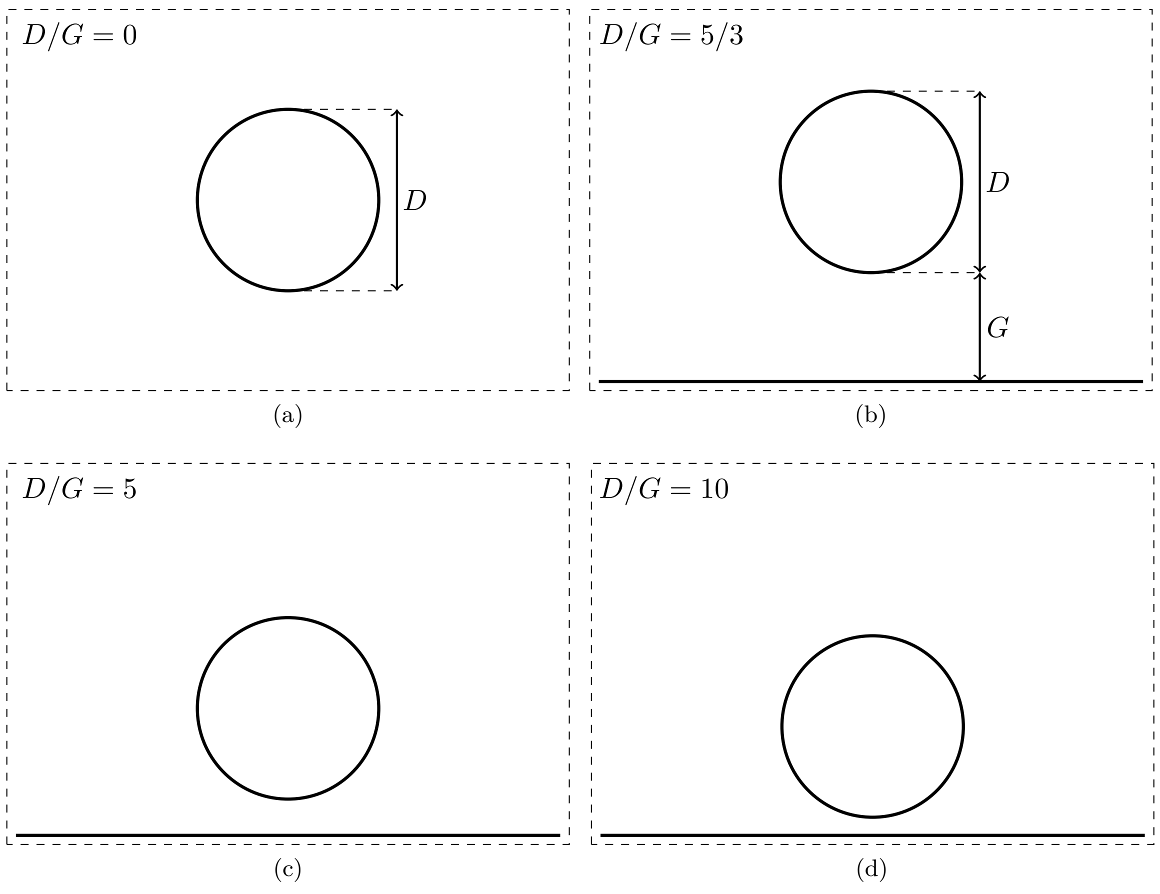


Figure 7.3: Schematic of the cylinder position compared to the wall for the four D/G-values.

Strouhal Number: The resulting Strouhal numbers from [8] and the set of tests are presented in table 7.8.

D/G-ratio	Article: St	Present: St
$D/G = 0$	0.213	0.213
$D/G = 5/3$	0.229	0.229
$D/G = 5$	0.174	0.173
$D/G = 10$	0.129	0.128

Table 7.8: Comparison of Strouhal numbers for four different positions of the cylinder at $Re = 300$ between results presented in [8] and results obtained in this work. The uncertainty in the data is estimated to be of the order $\pm 1 \cdot 10^{-3}$.

From table 7.8 it is seen that the Strouhal number found in the present simulations agrees extremely well with those calculated from [8].

Base Pressure Coefficient: The C_{bp} numbers from [8] along with the values from the test set are presented in table 7.9.

D/G-ratio	Article: C_{bp}	Present: C_{bp}
$D/G = 0$	-1.2 ± 0.02	-1.20
$D/G = 5/3$	-1.55 ± 0.02	-1.54
$D/G = 5$	-1.25 ± 0.02	-1.28
$D/G = 10$	-1.35 ± 0.02	-1.34

Table 7.9: Comparison of base pressure coefficient for four different positions of the cylinder at $Re = 300$ between results presented in [8] and results obtained in this work.

From table 7.9 it is seen that the base pressure coefficient found in the present simulations agrees with the values presented in figure 14 in [8] to within errors in reading the values from the article.

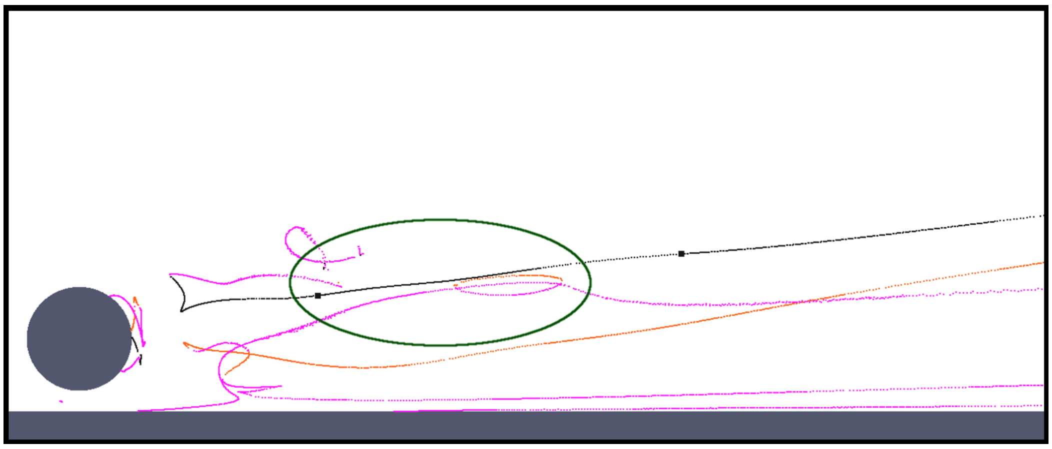
7.3 Death of the Transient Solution

The validity of the solutions obtained using the **Nektar++** framework has now been investigated and no evidence of them being wrong has been found. The next step is to start performing simulations using sets of parameters for which no data has been published. For the investigations of the vortex creation and their movement patterns as well as measuring quantities of interest we are only interested in the periodic shedding regime. This presents the challenge of determining when the initial transient solution has disappeared. The method used for determining that the transient solution has died out is presented below.

Stationary Flows: For parameter values resulting in stationary flows a measurement of the pressure induced drag was performed over a period of time comparable to a period of shedding for instationary flows. If this was found to be constant to within numerical errors for all time steps the initial transient was deemed to have disappeared. In addition the flows were inspected visually to assure that no changes could be seen between any of the snapshots in time.

Periodic Shedding: The death of the transient initial solution has been ensured by observing a timespan longer then one shedding period and during this span follow the critical points for the vorticity. The data for the position of the critical points is already available as this is what is used to study the creation and movement of vortices. This means that no new calculations are needed. If the critical points follow the exact same path after a shedding period has completed as they did during this period the transient solution has disappeared. This is true as any remaining transient would perturb the path of the critical points between two shedding cycles.

An example of the process of validating the disappearance of the transient is provided in figure 7.4. The figure shows part of the domain for a simulation with $Re = 240$ and $D/G = 5/1$.


(a)

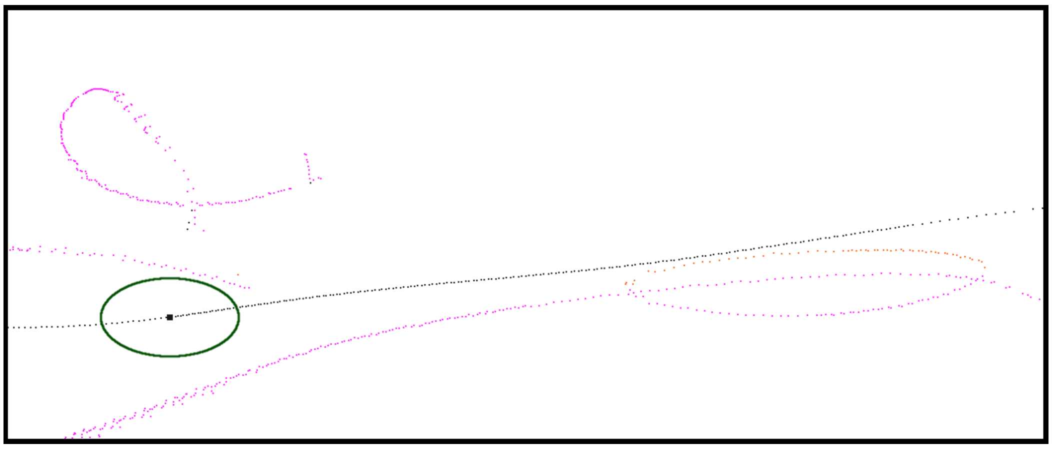

(b)

Figure 7.4: Illustration of the perfect overlap of the path of a critical point over longer then a period of vortex shedding used for validation that the transient solution has indeed disappeared. (a) a large set of the domain. The green circle highlights a part of one extrema path where the second shedding period overlaps with the first. (b) a zoom around the green circle on (a). The point data on the figures show the paths of the critical points as a function of time. Orange and black points are extrema in the vorticity while magenta points are saddle points for the vorticity.

From the area circled on figure 7.4 (a) it is seen that the density of data points for the vorticity extrema is twice as high over a part of the black curve marking the path of an extrema in vorticity over time. This is due to the fact that the simulation has run for more then a period and the additional points are thus marking the next vortex extrema tracing the exact same path as the previous.

Increased Transient Time: As one might expect from intuition it was discovered that the time it takes to make the transient solution disappear from the flow depends on the Reynolds number. It was discovered that this time likewise depends on the gab size to cylinder radius ratio D/G. For $Re = 300$ it is enough to simulate 40 normalised units of time before the transient disappears for all investigated D/G values, while already at $Re = 220$ it was found that for $D/G = 10$ the transient was not gone at 40 normalised units of times. And for $Re = 140$ both $D/G = 10$ and $D/G = 5$ still have some transient solution left at 40 normalised units of time. For Reynolds numbers around fifty[28] a full 160 normalized units of time was needed to achieve periodic shedding at $D/G = 5/4$. At $Re = 160$ and $D/G = 10$ 160 normalized time units was not enough to kill the transient solution completely.

Almost all simulations were allowed to run until the initial transient was completely gone. However for two simulations at $Re \in \{150, 160\}$ and $D/G = 10$ due to problems the simulations were not allowed to run long enough to remove the transient completely. The data from these simulations where only used in a single analysis presented in the following section and no new conclusions were drawn on basis of these simulations.

[28]The lowest Reynolds number at which periodic shedding was observed.

8 Analysis

This chapter presents all results obtained by using the methods presented in the earlier chapters for the simulations performed of the model problems. Section 8.1 presents the application of the vortex trace method for identifying vortex creation and annihilation points, their movement patterns and their change in magnitude downstream. The sections 8.2 and 8.3 presents two examples of how the method may be used to perform detailed studies of a specific structural change in the flow as a function of $(Re, D/G, t)$. Section 8.4 treats the application of uncertainty quantification to the test problems of a cylinder in free flow and a cylinder near the moving wall with Re being the input parameter containing uncertainty. Finally section 8.5 presents a limitation of the method in its current form.

8.1 Initial Investigation for the Cylinder and Wall

Saddle-Center Bifurcations Dominate the Model Problem: For all but a single case all vortex creation and annihilation, observed away from domain boundaries happened in codimension 1 saddle-center bifurcations. This is true over the entire $(Re, D/G)$-parameter range.

8.1.1 Stabilizing Effect of the Wall

During the investigation of the vortices creation and movement patterns it was discovered that the wall had a significant stabilizing effect on the flow. This stabilizing effect was investigated in order to identify at what parameter values $(Re, D/G)$ the flow changed character from being stationary to time dependent. Figure 8.1 presents a chart of different types of flows observed in the 2D-parameter space $(Re, D/G)$.

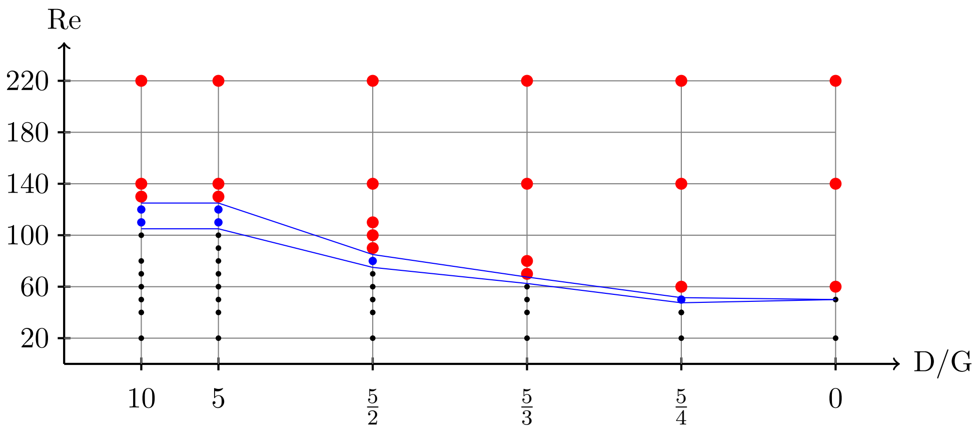
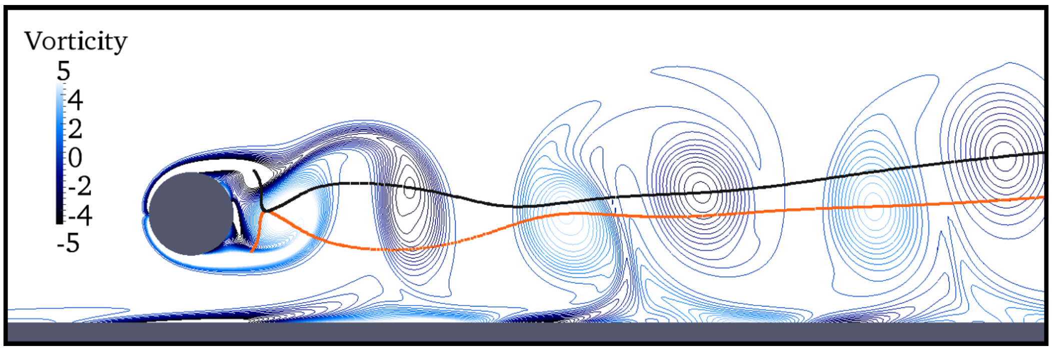
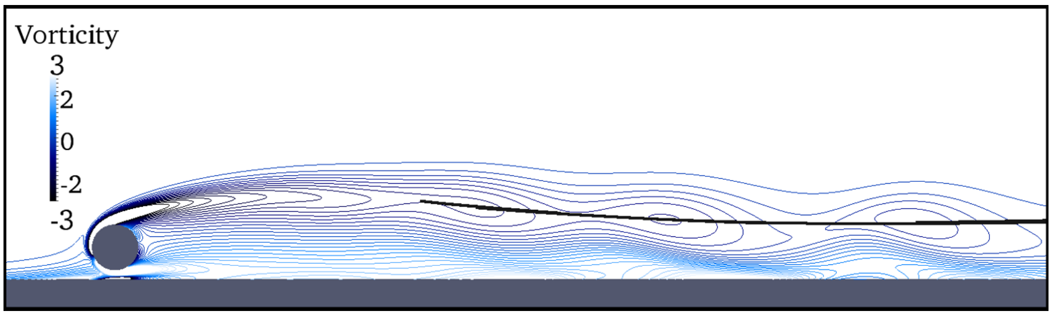


Figure 8.1: Map of stationary versus instationary flow past the cylinder near the moving wall in the 2 parameter space $(Re, D/G)$. Large (red) dots represent points in the parameter space where periodic shedding from both top and bottom of the cylinder has been observed. Medium (blue) dots represent selected points where only shedding from the top side of the cylinder was observed or where the flow has become instationary but no vortices has been observed within the domain. small (black) dots represent selected points where stationary flow was observed.

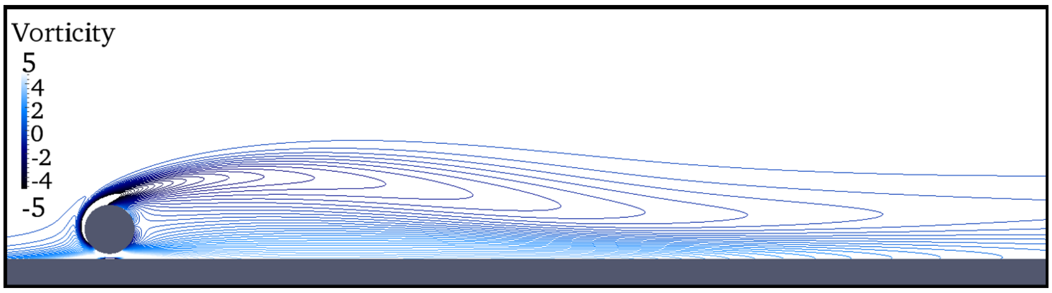
This map illustrates three parts of the $(Re, D/G)$ space where distinctly different behaviours were observed in the flow. All large red dots on figure 8.1 corresponds to parameter values where periodic shedding from both top and bottom of the cylinder was observed, see figure 8.2a for an example. All medium blue dots corresponds to parameter values where unsteady flow is observed but periodic shedding from both top and bottom of the cylinder is **not** observed, see figure 8.2b. This behaviour may either be due to no vortex creation or because the creation point have moved so far downstream that it is not captured by the simulation. Finally all black dots are points for which steady flow is observed, see figure 8.2c. All data presented in figure 8.1 are based observing the domain $(x, y) \in [-2, 20] \times [0, 6]$.



(a) $Re = 300, D/G = \frac{5}{4}$. At these parameter values two vortices are observed travelling downstream nearly symmetrically. The (orange) trace is the path followed by the clockwise rotating vortices shed from the lower side of the cylinder. The (black) trace is the path followed by the counter clockwise rotating vortices shed from the upper side of the cylinder.



(b) $Re = 120, D/G = 10$. Only shedding from the top of the cylinder is observed within the domain. The (black) trace is the path followed by the counter clockwise rotating vortices shed from the upper side of the cylinder.

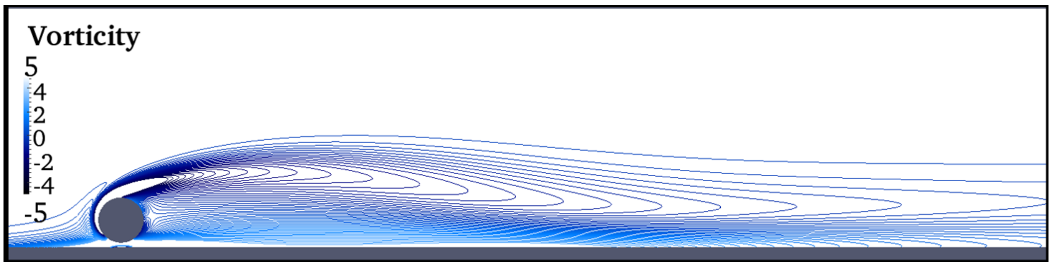


(c) $Re = 100, D/G = 10$. Purely stationary flow is observed.

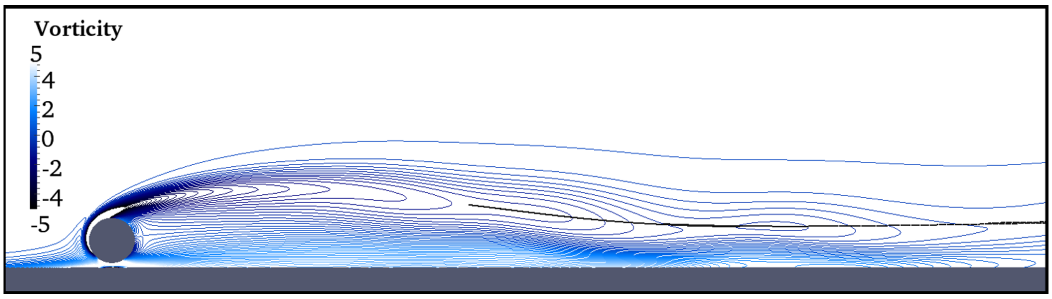
Figure 8.2: The center of the cylinders is positioned at $(x_c, y_c) = (0, \frac{D}{2} + G)$. The domain shown is given by: (a) $(x, y) \in [-2, 10] \times [0, 4]$. (b),(c) $(x, y) \in [-2, 20] \times [0, 6]$.

As may be seen from figure 8.1 the wall has a significant stabilizing effect of the flow. At $D/G = 10$ the flow is stationary at least up to the Reynolds number $Re_{\text{crit}} = 100$ in contrast to a cylinder in free flow where the critical Reynolds number has been found to be $Re_{\text{crit}} \approx 46 \pm 1$, [7]. Also it is seen that as the cylinder is moved closer to the wall a regime appears in which periodic shedding from both sides of the cylinder is no longer observed. In this regime either only shedding from the top side of the cylinder is observed, or only an unsteady flow without any vortex formation observed within the part of the domain downstream of the cylinder under consideration.

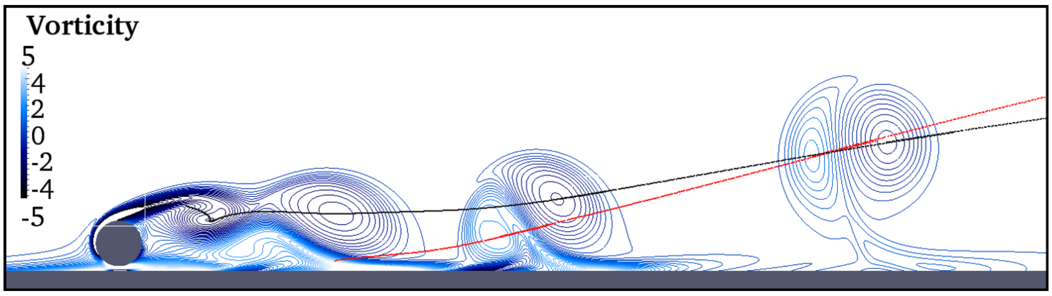
An illustration of the transition from stationary flow, across the parameter values without periodic shedding from both sides of the cylinder, to parameter values with periodic shedding is given in figure 8.3 for $D/G = 10$ and $Re \in [100, 180]$.



(a) $Re = 100, D/G = 10$. Purely stationary flow is observed.



(b) $Re = 120, D/G = 10$. Only shedding from the top of the cylinder is observed within the domain. The (black) trace is the path followed by the counter clockwise rotating vortices shed from the upper side of the cylinder.



(c) $Re = 180, D/G = 10$. At these parameter values two vortices are observed travelling downstream nearly symmetrically. The (orange) trace is the path followed by the clockwise rotating vortices shed from the lower side of the cylinder. The (black) trace is the path followed by the counter clockwise rotating vortices shed from the upper side of the cylinder.

Figure 8.3: The cylinders center is positioned at $(x_c, y_c) = (0, \frac{D}{2} + G)$. The domain shown is given by: $(x, y) \in [-2, 20] \times [0, 6]$.

By comparing figures 8.2a and 8.3c an interesting effect of moving the cylinder close to the moving wall may be observed. In figure 8.2a it is seen that the clockwise and counter clockwise rotating vortices are shed and travels downstream with almost equal spacing between vortices. In contrast it is seen in figure 8.3c that the clockwise rotating vortex appears to pull the counter clockwise rotating vortex off the wall and afterwards they continue downstream as a pair.

8.1.2 Different Critical Point Structures

During a period of shedding seven qualitatively different center and saddle point patterns have been identified in the flow for the cylinder near the moving wall. Five of them exists for $(Re, D/G, t) = (240, 5/3) \times [t_0, t_0 + T]$, where T is a period. These five are illustrated on figure 8.4. The final two structures are treated in detail in section 8.2 and 8.3.

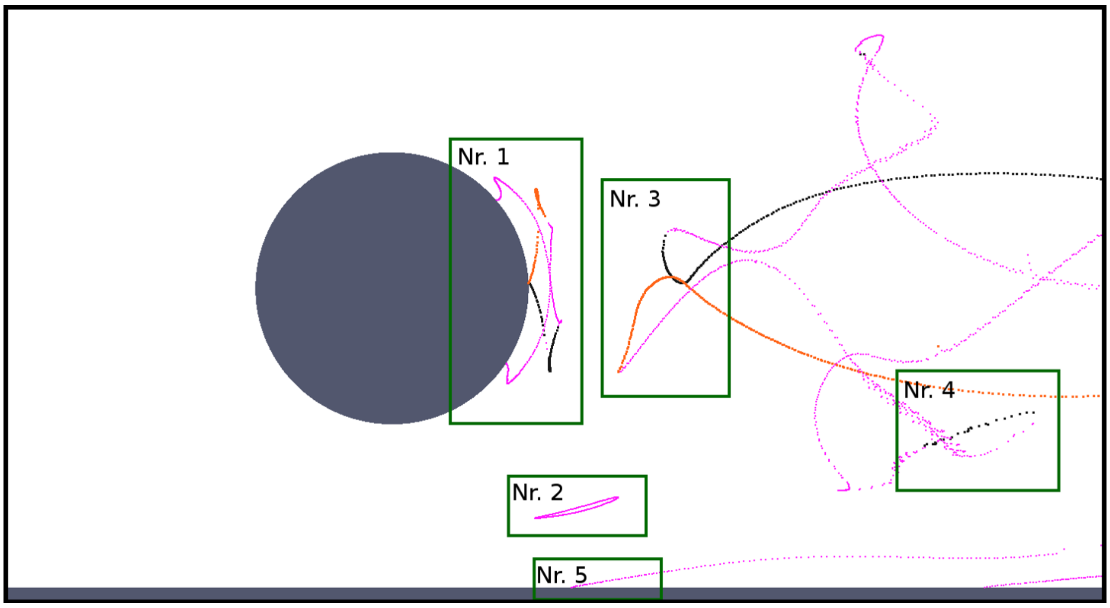


Figure 8.4: Plot of the traces of extrema and saddle points over a period of shedding, parameter values $Re = 240$, $D/G = 5/3$. This plot illustrates five different structures observed in the vorticity field over a period in the periodic shedding regime. **Nr. 1**: Creation of vortices and saddles from the backside of the cylinder surface and following annihilation in saddle-center bifurcations. **Nr. 2** Closed saddle-point cycle between cylinder and wall. **Nr. 3** Saddle-Center bifurcations creating vortices surviving downstream. **Nr. 4** Saddle-Center bifurcation leading to short lived extrema in vorticity. **Nr. 5** Saddle point creation at moving wall. Domain shown have the dimensions $(x, y) \in [-1.5, 2.5] \times [0, 2]$.

The five structure types shown on figure 8.2 are discussed in short below,

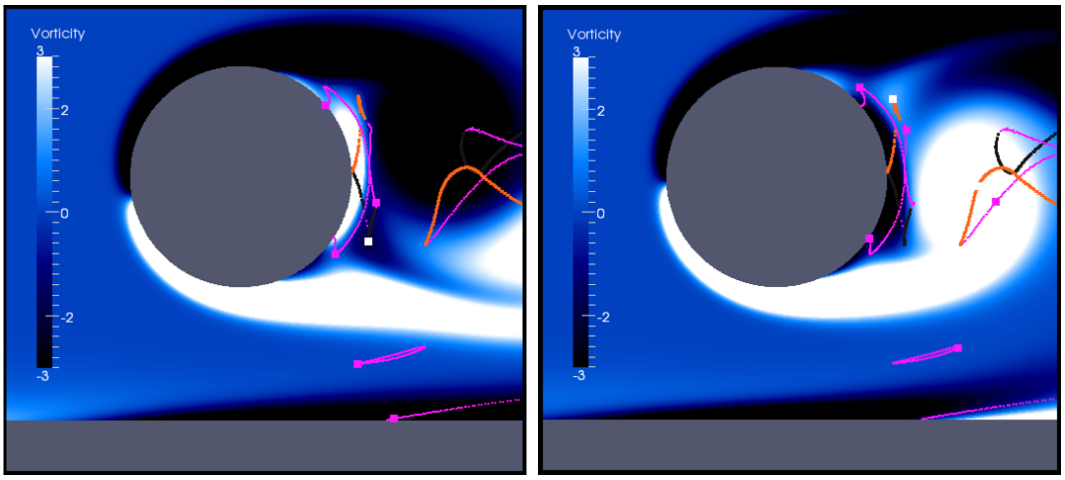
- **Nr. 1. Structure on cylinder downstream side:** Increasing the Reynolds number beyond the critical value where the flow goes from being stationary to periodic a structure consisting of the creation of two vortices and two saddles are observed on the downstream side of the cylinder over the duration of a period. Each vortex leave the cylinder and clash with one of the saddle-points in a saddle-center bifurcation. Although their movement patterns change slightly with varying $(Re, D/G)$ the structure persists for all $Re > Re_{\text{crit}}$. The vortices are illustrated in figure 8.5 which show a vorticity plot at two different instances in time.
- **Nr. 2. Saddle cycle:** For all $(Re, D/G)$ values a saddle cycle exists in the area between the cylinder and wall. It consists of a saddle point in vorticity which travels in a cycle over a period of shedding. This saddle-point mark the point in the flow

between the wall and cylinder where the fluid rotates in opposite directions. Near the wall fluid is ripped off the wall creating a clockwise rotation in the flow. Near the cylinder fluid is ripped off the cylinder resulting in fluid rotating counter clockwise. This effect happening on either side of the saddle point is also illustrated in figure 8.5.

- **Nr. 3. Saddle-Center bifurcation, surviving vortices:** For all Re-values above the critical value for periodic shedding one or two vortices which survive downstream are created downstream of the cylinder. This always happen in saddle-center bifurcations, and will be treated in more detail later.

- **Nr. 4. Weak Center appearance/disappearance:** For some $(Re, D/G)$-values in the periodic shedding regime at various places downstream of the cylinder small vorticity extrema appears and disappears in saddle-center bifurcations over the course of a period. These extrema were all found to appear near the center of a vortex travelling downstream. The extrema only exist over a short time span during a period and therefore they are not investigated further.

- **Nr. 5. Saddle creation at wall:** Along the moving wall alternating patches of clockwise and counter clockwise rotating fluid exists. Between these patches travel saddle points which appear from the wall. No vortices were found to appear directly from the wall. A saddle point created at the wall is also illustrated in figure 8.5.



(a) Illustration of a counter clockwise rotating vortex near the backside of the cylinder, marked by a white square. Also an illustration of the creation of a saddle-point at the wall.

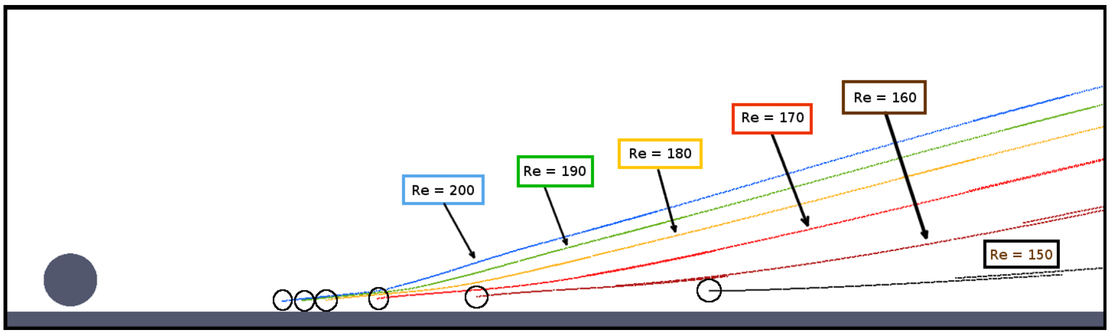
(b) Illustration of a clockwise rotating vortex near the backside of the cylinder, marked by a white square.

Figure 8.5: Parameter values: $Re = 240$, $D/G = 5/3$. Two temporal snapshots during a period of shedding. Among others the figures illustrate the cyclic travelling saddle point between the cylinder and wall. Saddles are marked by purple squares.

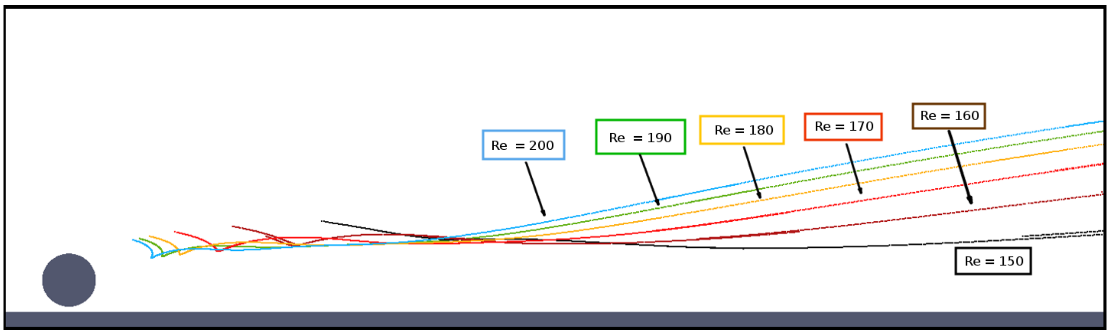
8.1.3 Downstream Surviving Vortices, Creation Points and Movement Patterns

An investigation of the effect of varying the Reynolds number and D/G-ratio on the creation point and movement path of the vortex pair which survive downstream has been performed. Detailed results for $D/G = 10$ are presented in this section along with conclusions on findings for other D/G-values.

Figure 8.6 provides a map of the creation point and pathway for the clockwise and counter clockwise rotating vortices surviving downstream at $D/G = 10$ and $Re \in [150, 200]$. Results presented here are based on simulations which were allowed to run for at most $t = 160$ time units. As may be seen from figure 8.6 this simulation time was not enough to obtain perfect periodic shedding for $Re \in \{150, 160\}$.



(a) Illustration of the creation point and pathway of the counter clockwise rotating vortex created near the wall at different Reynolds numbers.



(b) Illustration of the creation point and pathway of the clockwise rotating vortex created away from the wall at different Reynolds numbers.

Figure 8.6: Parameter values: $Re \in [150, 200]$, $D/G = 10$. Shown domain, $(x, y) \in [-1, 20] \times [0, 6]$. Cylinder center $(x_c, y_c) = (0, \frac{D}{2} + G)$.

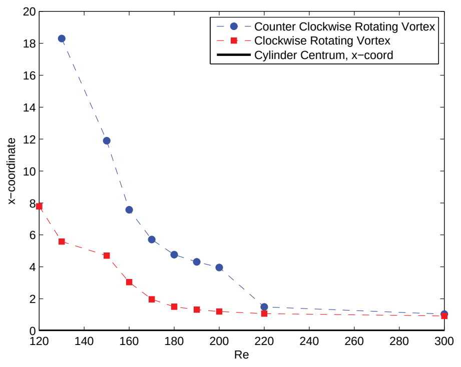
Figure 8.6 illustrate multiple findings, listed below.

- Creation points for both the clockwise and counter clockwise rotating vortices moves downstream as the Reynolds number is reduced.
- Both vortices are pushed further and further away from the wall with increasing Reynolds number when the cylinder is close to the wall.
- When the Reynolds number nears the critical value for periodic shedding the transient solution is not gone after a simulation over four times the length of a simulation where

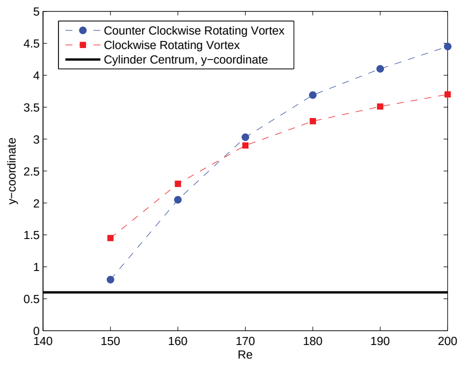
Re is further from the critical value. Thus in order to obtain a solution which has reached periodic shedding, for which no effect of the transient solution is left, very long simulation times are needed at Re close to Re_{crit}.

- At a given Reynolds number the counter clockwise rotating vortex created near the wall moves away from the wall at a steeper angle than the clockwise rotating vortex.

The x-coordinate of the creation point for the vortex and its distance to the wall at $x = 20$ for both the clockwise and counter clockwise rotating vortices has been recorded at a number of Re values at $D/G = 10$. The results are presented in figure 8.7.



(a) The x-coordinate of the creation point for both surviving vortices as a function of Reynolds number.



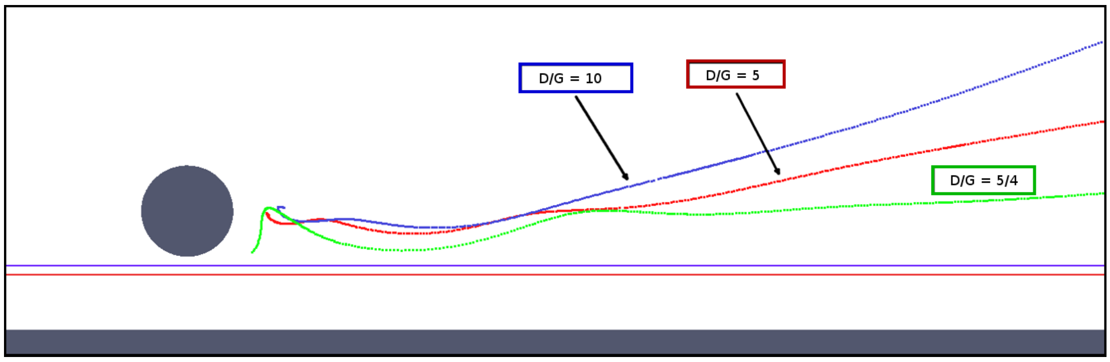
(b) The distance from the wall of the vortices travelling downstream at $x = 20$ corresponding to twenty cylinder diameters.

Figure 8.7: Parameter values: $D/G = 10$, $Re \in [120, 300]$. The wall is positioned at $y = 0$ and the cylinder center at $(x, y) = (0, \frac{D}{2} + G)$.

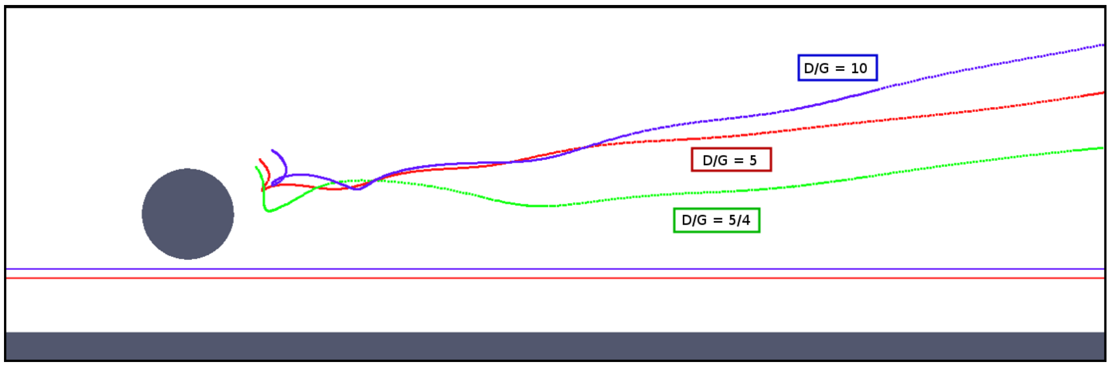
In figure 8.7a it is seen that the distance downstream of the cylinder where the two vortices are created increases monotonously as the Reynolds number is reduced. It is also seen that the creation point of the counter clockwise rotating vortex near the wall moves further downstream than that of the clockwise rotating vortex as Re is lowered. Figure 8.7b shows that increasing the Reynolds number forces the vortices further away from the wall. It also shows that the counter clockwise rotating vortex is forced from the wall at a steeper angle than the clockwise rotating vortex when the Reynolds number is increased.

Similar tests have been performed for $D/G \in \{5, 5/2, 5/3, 5/4\}$. The same tendency of the creation points moving downstream with decreasing Reynolds number was seen. Due to limited sample sizes at these D/G-values the data is not presented in detail here.

With respect to the influence of the wall on the pathway followed by the vortices it was found that as the cylinder is moved away from the wall towards $D/G = 5/4$ the influence decreases significantly. The pathways followed by the vortices at different distances to the wall are illustrated below in figures 8.1.3 and 8.8a for $Re = 300$.



(a)



(b)

Figure 8.8: $Re = 300$. Trace of (a) counter clockwise rotating and (b) clockwise rotating vortices for different D/G-ratios. The wall is marked by a blue line for $D/G = 10$ and a red line for $D/G = 5$ and dark gray for $D/G = 5/4$. The trace of the vortices at $D/G = 10$ are coloured blue, at $D/G = 5$ they are coloured red and at $D/G = 5/4$ they are coloured green. Domain dimensions: $(x, y) \in [-2, 10] \times [0, 3.5]$.

From the figures its clear to see that the vortices are forced much further from the wall at $D/G = 10$ than at $D/G = 5/4$. At $D/G = 5/4$ only a small deflection in the traces is seen compared to other D/G values.

From figure it is clear to see that after they are created the vortices move around behind the cylinder before starting to travel downstream. This behaviour was observed for most $(Re, D/G)$-values. It is caused by the fact that the extrema in vorticity is created at some point during accumulation of rotating fluid behind the cylinder during the shedding period. First at a later point in time does the rotating fluid start moving downstream with the vorticity extrema at its center. This leaves the vorticity extrema travelling around behind the cylinder for part of its existence before starting its journey downstream.

8.1.4 Vortex Strength Reduction

In the investigation of the creation point and pathway followed by the vortices surviving downstream of the cylinder an interesting question becomes. *How quickly the magnitude of the vortices decrease?*.

If the magnitude of the vortices decreases to an insignificant level only a few cylinder diameters downstream the exercise of tracing them far from their creation point becomes pointless. On the other hand, if the vortices remain intense far downstream tracing their path is interesting.

The measure for vortexes magnitude used here is the magnitude of vorticity ω at its center. The distance downstream is measured as the distance from the cylinder center, $x = 0$, directly downstream measured in cylinder diameters.

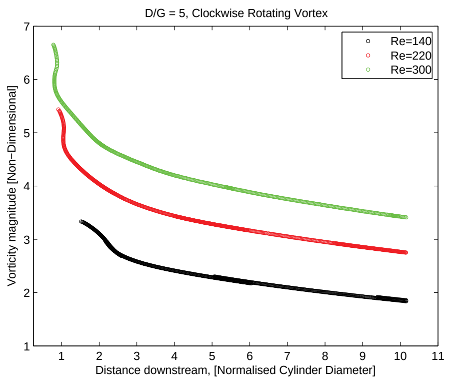
Note: *The temporal dependence of the vortices movement downstream is not considered here. Hence the time it takes the vortex to travel the first two cylinder diameters downstream may be the same it takes it to travel the next eight cylinder diameters. Thus measured in time the vortex decrease rate might be the same everywhere in the flow.*

The investigation of the intensity has been performed for the values $D/G \in \{10, 5, 5/2, 5/3, 5/4, 0\}$ and *Re*: $Re \in \{140, 220, 300\}$. The magnitude of both the clockwise rotating vortex and the counter clockwise rotating vortices were tracked.

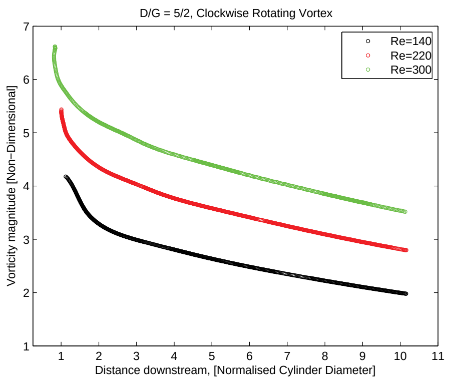
Multiple findings were made regarding the magnitude of the vortices dependence on *Re* and D/G and how it decreases as the vortices move downstream. The findings are listed below and are also considered in the following paragraphs which include figures presenting the vorticity data.

- The intensity of both the clockwise and counter clockwise rotating vortex depends on *Re* for all D/G-values. Increasing *Re* increases the vortex intensity at the point of creation and the intensity stays higher downstream. The data is shown in figures 8.9 and 8.10.
- Introducing the moving wall increases the vorticity for both vortices compared to a cylinder in free flow at D/G ratios smaller than 5.
- The magnitude of both vortices are nearly independent of D/G until $D/G < 5/2$ at all tested *Re*. Passing this value, i.e. moving the cylinder closer to the wall, the intensity of the vortices decrease in magnitude. This is true both for the magnitude at creation and the magnitude downstream. This behaviour is shown in figure 8.11.
- The rate of the decrease in vorticity as a function of distance downstream is for most parameter values separated in two phases. In the first phase where the vorticity extrema stays behind the cylinder for an extended period of time the rate of decrease is very high. In the second phase where the vortex travels downstream it is much lower. In most cases the rate of decrease was found to be approximately constant during the second phase.
  - For the clockwise rotating vortex the rate of decrease in vorticity during the second phase was found to only dependent slightly on D/G for $D/G > 5/2$. For the counter clockwise rotating vortex (the vortex closest to the wall) the rate changed considerably with D/G.
  - The rate of decrease in vorticity during the second phase for the clockwise rotating vortex increased with decreasing *Re* for all D/G-values. The same was found for the counter clockwise rotating vortex until $D/G = 5/2$.
- Past eight cylinder diameters downstream the change in vorticity has become approximately linear.

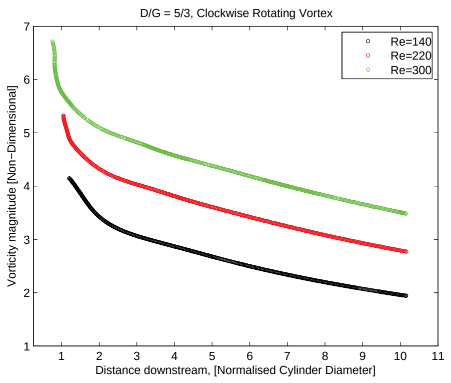
Vorticity Magnitude: The magnitude were calculated for both the clockwise and counter clockwise rotating vortices and it was found that the intensity increases with increasing Re for all D/G-values. The figures 8.9 and 8.10 show the magnitude of the vorticity as a function of the distance the vortices have travelled downstream measured in cylinder diameters. These figures are sorted by D/G-ratio with varying Re.



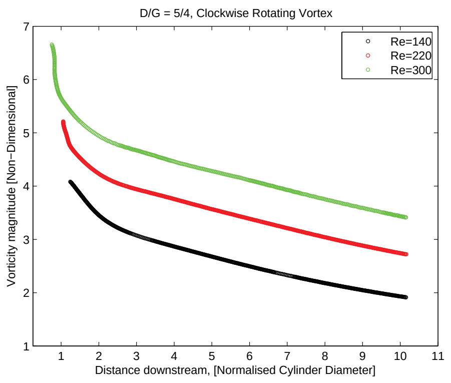
(a)



(b)

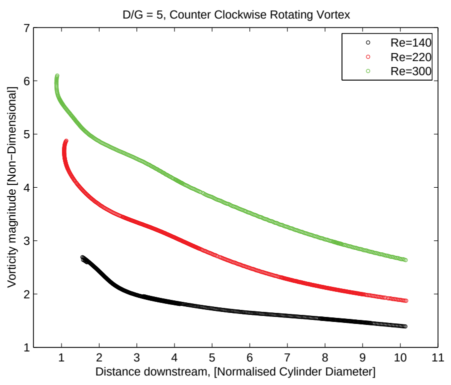


(c)

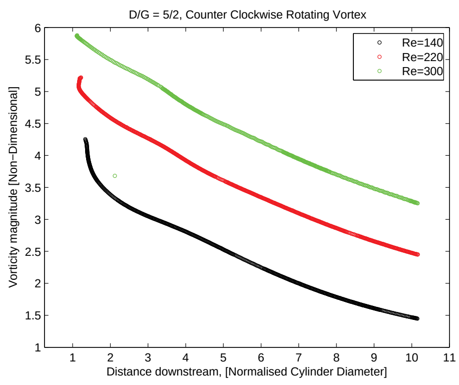


(d)

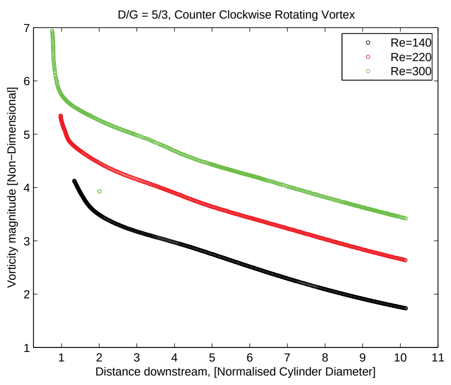
Figure 8.9: Graphs showing the magnitude of the vorticity at the center of the clockwise rotating vortex as a function of its distance travelled downstream of the cylinder from its creation point. $x = 0$ corresponds to the x-coordinate of the center of the cylinder. Each plot is for a fixed D/G-ratio with different Re values.



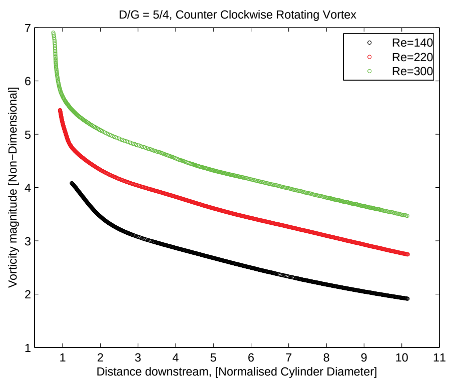
(a)



(b)



(c)

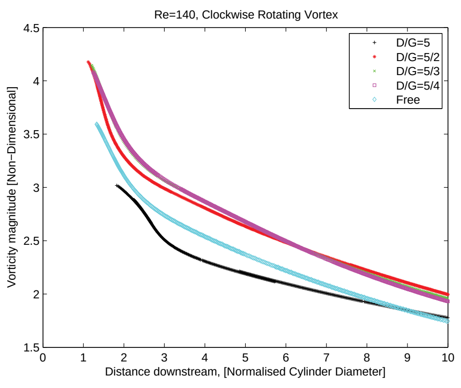


(d)

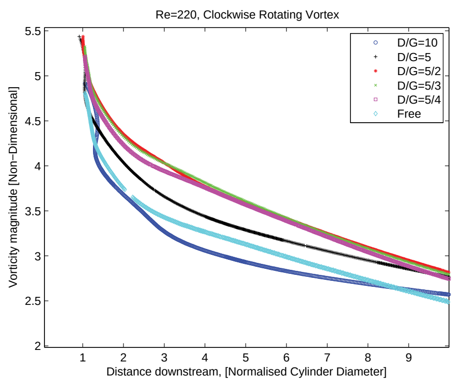
Figure 8.10: Graphs showing the magnitude of the vorticity at the center of the counter clockwise rotating vortex as a function of its distance travelled downstream of the cylinder from its creation point. $x = 0$ corresponds to the x-coordinate of the center of the cylinder. Each plot is for a fixed D/G-ratio with different Re values.

From the figures the increase in vorticity with increasing Re for all D/G-values is easily seen. It is also clear from the figures that the rate of decrease in vorticity may be considered to happen in two distinct phases. This behaviour will be discussed in the next paragraph.

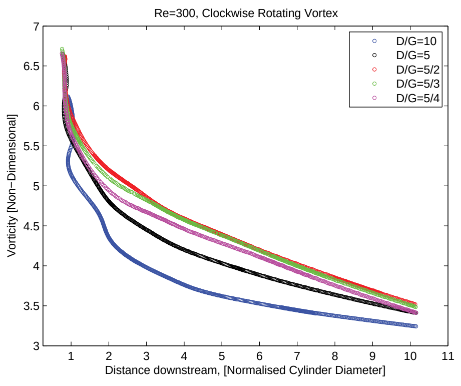
Another interesting finding is that introducing the wall actually made the vortices stronger for $5/4 \leq D/G < 5/2$. Also the magnitude and change in vorticity almost does not vary with D/G for $D/G \leq 5/2$. When moving the wall from $D/G = 5/2$ to $D/G = 5$ however, a clear drop in vorticity is seen. This shows that the cylinder must be close to the wall for it to decrease the strength of the vortices. However at the close range the wall has a clear effect on the intensity of the vortices shed from it. The behaviour described here is shown in the figure 8.11, which presents the same data as the figures 8.9 and 8.10 but sorted by fixed Re instead.



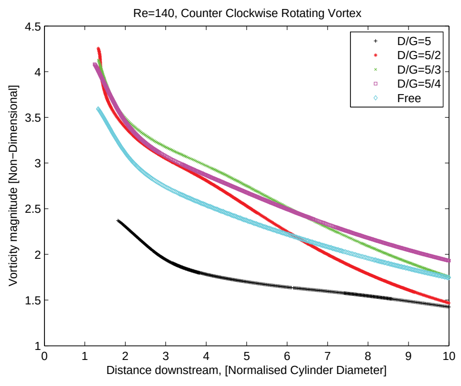
(a)



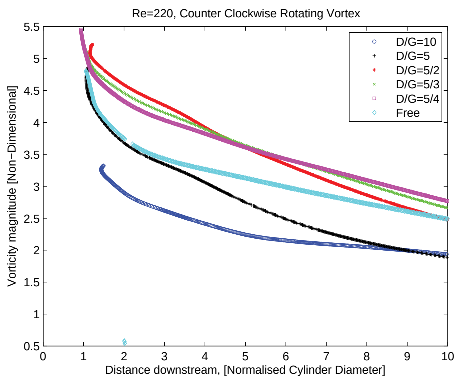
(b)



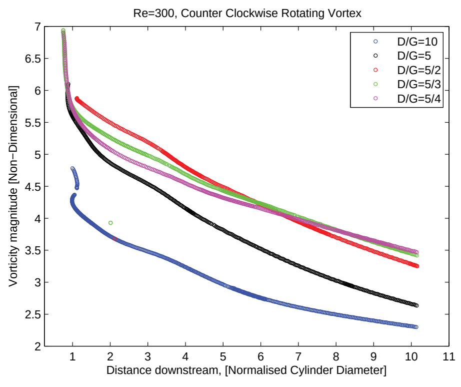
(c)



(d)



(e)



(f)

Figure 8.11: Graphs showing the magnitude of the vorticity at the center of the (a),(b),(c) counter clockwise rotating vortex and (d),(e),(f) clockwise rotating vortex as a function of its distance travelled downstream of the cylinder. $x = 0$ corresponds to the x-coordinate of the center of the cylinder. Each plot is for a fixed Re-value with different D/G-ratios.

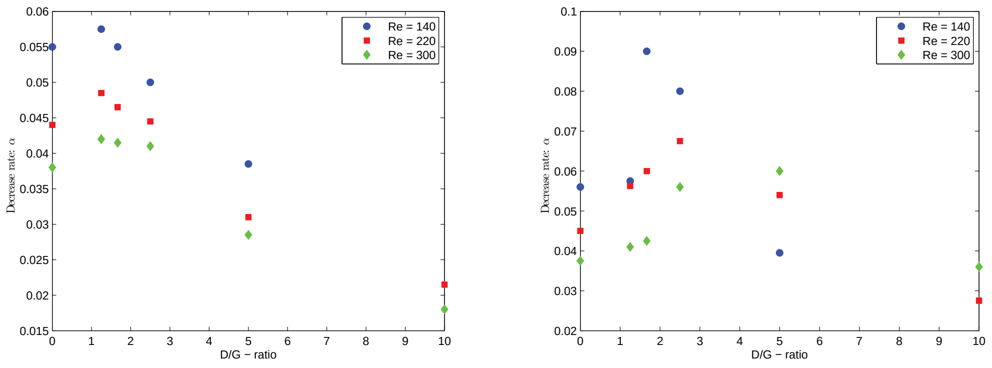
Vorticity Decrease Rate: From the figures presented in the previous paragraph it is clear to see that for most Re and D/G values the vorticity decreases rapidly over the first one to two cylinder diameters downstream. At around $x = 3$ the decrease rate slows down significantly and past seven cylinder diameters it has become approximately constant for most of the Re and D/G values. That is, it is observed that the vorticity decreases at two distinctly different rates close to the cylinder and further downstream.

Note: *As stated earlier part of the reason for the initial high decrease rate in vorticity magnitude may be attributed to the finding that the vortices tend to stay some time behind the cylinder after their creation before travelling downstream.*

In order to investigate the approximately constant decrease rate the magnitude of the vorticity past $x = 8$ has been scaled with its value at $x = 8$ as,

$$\omega_{\text{Scaled}}(x) = \left|\frac{\omega(x)}{\omega(x=8)}\right|, \quad x \geq 8. \tag{8.1}$$

Graphs showing the approximately linear decrease in vorticity between $x = 8$ and $x = 10$ for $\omega_{\text{Scaled}}(x)$ for both the clockwise and counter clockwise rotating vortex are presented in figure 8.13. From the data presented in the figure an approximately constant fractional decrease rate α may be calculated and is found to lie between 0.02 and 0.09 depending on Re and D/G-ratio. This have been done and the results are presented in the figure 8.12



(a) Clockwise Rotating Vortex.

(b) Counter Clockwise Rotating Vortex.

Figure 8.12: Constant decrease rate approximations, α calculated based on data presented in figure 8.13.

From figure 8.12 it may be seen that introducing the wall actually initially increases the decrease rate in all cases. As the cylinder is moved closer to the wall however it is seen that the decrease rate drops significantly. For the clockwise rotating vortex the drop in decrease rate is observed for $D/G = 5/4$ for all Re-values. For the counter clockwise rotating vortex the drop in decrease rate seems to depend on Re where increasing Re makes the drop happen closer to the wall. A jump is observed for $Re = 140$ between $D/G = 5/4$ and $D/G = 5/3$ for the counter clockwise rotating vortex to which no explanation was found.

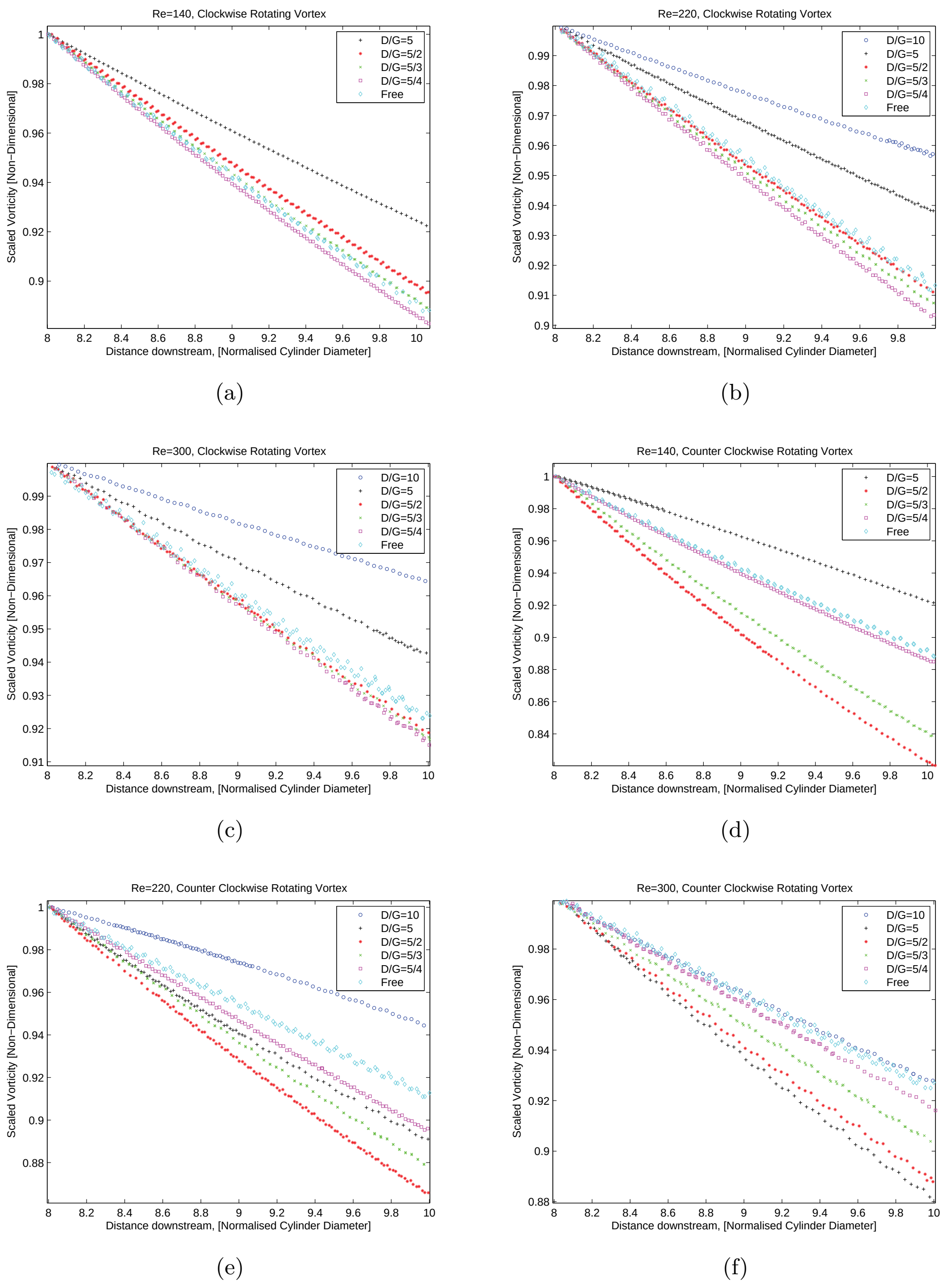


Figure 8.13: Each plot is for a fixed Re-value with different D/G-ratios. The plots are of scaled vorticity, ω_{Scaled}, as a function of distance downstream of the cylinder, x: $\omega_{\text{Scaled}} = \left|\frac{\omega(x)}{\omega(x{=}8)}\right|$. Figures (a),(b),(c) show data for the clockwise rotating vortex. Figures (d),(e),(f) show data for the counter clockwise rotating vortex.

Investigating the data in figure 8.13 in detail shows that the decrease is slightly slower then linear for almost all cases. That is, $\frac{\partial \omega}{\partial x}$ is not constant but decreases slightly as a function of x. However the approximately constant decrease rate from $x = 8$ can be used to provide a lower bound on vortex magnitude further downstream.

In order to estimate the lower bound one may use the expression,

$$|\omega(x)| = |\omega(8)|(1 - \alpha(x - 8)) \ , \quad x \geq 8, \alpha \in [0.02, 0.09]. \tag{8.2}$$

Magnitude Decrease Downstream: Considering the vortices from their creation points, their magnitude may be estimated to have decreased by between 40 percent and 60 percent at 10 cylinder diameters downstream. Thus when considering the vortices from the point where they are strongest they have only halved in strength ten diameters downstream.

Excluding the first rapid decrease in magnitude, i.e. considering the vortices from 3 cylinder diameters downstream, the intensity of the vortices have decreased by between 20 percent and 40 percent at 10 cylinder diameters downstream.

Using $\alpha = 0.09$ it can be estimated that the vortices have all but died out at 20 cylinder diameters downstream. Instead using $\alpha = 0.02$ the vortices have only lost roughly 25 percent of their magnitude from 8 to 20 cylinder diameters downstream.

Remembering that this is a lower bound on the vorticity since the actual decrease rate slows down slightly as a function of x, the vortices in fact remain slightly stronger downstream than the approximation presented above suggests.

Choice of zero point for time: For multiple purposes it is practical to define a zero point, t_0, in time as a reference point during a shedding period. As the shedding is periodic the choice of t_0 may be made freely yet it is important to choose carefully for the visualisation and understanding of some results. As both shedding frequency and flow structure depends on D/G and Re it is not obvious what choice should be made as it should be general enough to be applicable for all simulations. For all Reynolds numbers above the critical Reynolds number for vortex shedding Re_{crit} a single event is observed to always occur regardless of the choice of Re and D/G. This event is the appearance of the clockwise rotating vortex in center-saddle bifurcation above and behind the cylinder. The center-saddle bifurcation point in question is highlighted by a circle in figure 8.14.

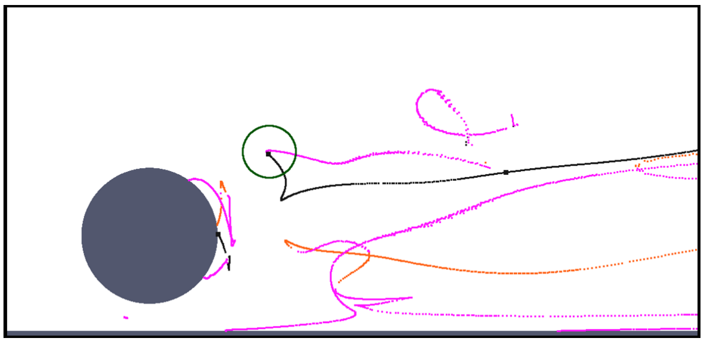


Figure 8.14: Critical point trace for $Re = 140$, $D/G = \frac{5}{2}$ with highlight of the spatial point where the extrema-saddle creation event which define t_0 occurs. The domain dimensions: $(x, y) \in [-1, 4] \times [0, 2.4]$

By defining the center-saddle bifurcation event as the beginning of a shedding cycle we now have a well-defined t_0 from which the temporal value of all other events may be defined. An added benefit of this choice of t_0 is that it may been used to estimate the shedding period with high accuracy.

Period of shedding / Strouhal number: The shedding period in non-dimensionalized time, $T = t\frac{U_\infty}{D}$ and the corresponding Strouhal numbers have been recorded for all simulations and selected results are displayed in figure 8.15.

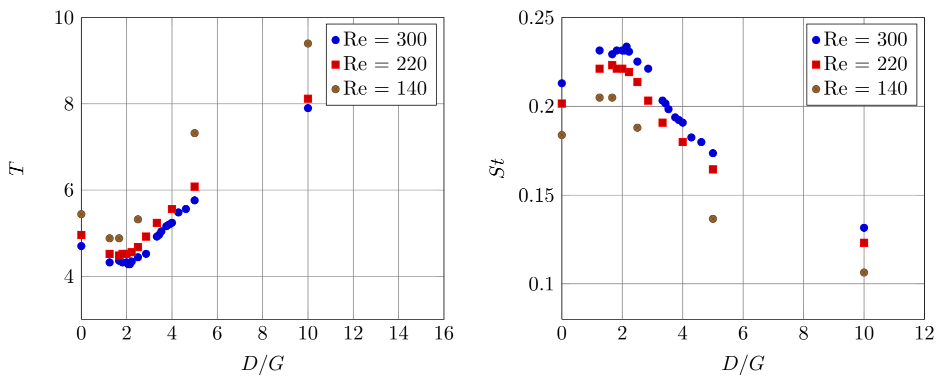


Figure 8.15: Scatter plot of (a) the non-dimensionalized period and (b) the non-dimensionalized frequency. Note that $D/G = 0$ corresponds to the cylinder in free flow.

From figure 8.15 it can be seen that varying the D/G-value has a greater effect on the Strouhal number than varying the Reynolds number in the range investigated. Moving the cylinder very close to the wall almost doubles the shedding period compared to the cylinder in free flow. In contrast doubling Re only decrease the period by roughly 15%. It is also seen that the Strouhal number appears to exhibit a maximum around $D/G \approx 2$ for all three values of Re. This is in complete agreement with findings presented in [8, figure 17] by Huang and Sung.

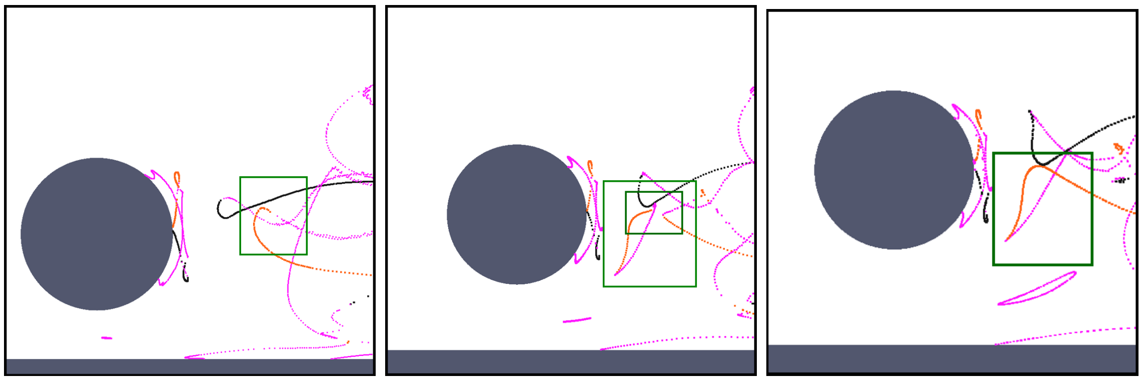
Figure 8.15 also shows that the wall still influences the shedding period at $D/G = 5/4$.

8.2 Formation and Disappearance of Extrema-Saddle Pair

In the previous sections of this chapter, the vortex trace method has been used to trace vortices through the flow, track their magnitude and to identify and illustrate several bifurcations in time at fixed $(Re, D/G)$-values. All bifurcations away from the wall and cylinder observed so far have all been saddle-center bifurcations in time. In this section the occurrence of a bifurcation which alters the structure of the flow as Re and D/G are varied is treated.

A structure has been identified which only exists over a limited part of the shedding period, T, and only for certain $(Re, D/G)$-values. This structure consists of the formation and disappearance of a counter clockwise rotating vortex behind the cylinder. This extrema does not survive downstream but instead is created and annihilated behind the cylinder. The idea is to identify the $(Re, D/G)$-pair at which this structure appears in the flow and provide an explanation of how this happens.

An illustration of the structure in question is provided in figure 8.16. The figure shows the extrema and saddle traces for three simulations with $Re = 280$ and $D/G \in \{3.16, 2.14, 1.67\}$ respectively. These choices of D/G corresponds to the situation before the closed center-saddle pair appears, during its existence and after it may no longer be found.



(a) $D/G = 3.16$, The creation point of the counter clockwise rotating vortex surviving downstream is seen inside the green box.

(b) $D/G = 2.14$, A vortex-saddle pair appears and vanishes behind the cylinder before the creation point of counter clockwise rotating vortex.

(c) $D/G = 1.67$ the annihilation point for the vortex-saddle pair appearing behind the cylinder and the creation point of the counter clockwise rotating vortex have merged.

Figure 8.16: Reynolds number: $Re = 280$. Illustration of the appearance and disappearance of a closed saddle-center structure in the flow when D/G is varied.

From the areas marked with green squares in these figures it may be observed that varying the parameter D/G from 3.16 to 2.14 causes the structure to change. At $D/G = 2.14$ an extrema and a saddle is created at some t_c in a center-saddle bifurcation. This pair disappears again in another center-saddle bifurcation at some other time t_d before the vortex surviving downstream is created at a third time t_{ds}. Varying D/G further from 2.14 to 1.67 apparently results in the four critical points highlighted with small square in figure 8.16c merging in a new bifurcation.

The result of the appearance and disappearance of this structure is that the creation of the vortex closest to the wall jumps from right behind the cylinder to a more symmetric position behind and below the cylinder.

Note: *The following investigation of this structural change illustrate the sensitivity of the method of tracing the extrema of the vorticity. Even though the flow looks very similar as the values of D/G and Re are varied across the threshold where the closed center-saddle structure exists the method clearly identifies this structure.*

Scaling the Time: As seen in figure 8.15 the shedding period changes as a function of both Re and D/G. This change in period leads to the choice of a simple linear rescaling of the time during a single shedding period. Thus the time over a period $t \in [0, T]$ for a given $(Re, D/G)$-pair is rescaled to $\tilde{t} = [0, 1]$. This allows for a better illustration of the bifurcation diagrams presented in the following section. The reader may consult figure 8.15 for the exact St number for each $(Re, D/G)$-pair at $Re \in \{140, 220, 300\}$. For different Reynolds numbers the St number may be estimated fairly accurately from the figure as no irregular variation in St was observed at any $(Re, D/G)$-value.

8.2.1 Constant *Re* Bifurcation Diagrams

The game now becomes to identify at what $(Re, D/G)$-values the bifurcation that leads to the structural change illustrated in figure 8.16 occurs. To do this a large number of simulations at varying Re and D/G values have been performed.

Based on the data from these simulations this section presents a series of 2D bifurcation diagrams in D/G and $\tilde{t}$, each for fixed Reynolds number. These 2D diagrams corresponds to slices through the 3D parameter space given by $(Re, D/G, \tilde{t})$. As defined in a previous paragraph $\tilde{t} = 0$ corresponds to the point in time where the clockwise rotating vortex is shed from the topside of the cylinder. The diagrams then shows at what fraction of a full shedding period the counter clockwise rotating vortex is shed from the bottom side of the cylinder. This event is denoted by a (gray) dot. More importantly the diagrams show at what fraction of the period the saddle-center pair forming the closed structure appears, denoted by a (red) dot, and disappears, denoted by a (blue) dot, for all values of $(Re, D/G)$ where this structure exists. By scanning through different Reynolds numbers this method allows for a precise identification of the Re-value at which the closed center-saddle cycle no longer appears for any D/G-values.

Re = 300:

The largest Reynolds number considered in this investigation is $Re = 300$. Figure 8.17 shows that at this Re the closed center-saddle structure exists for a wide variety of D/G-values.

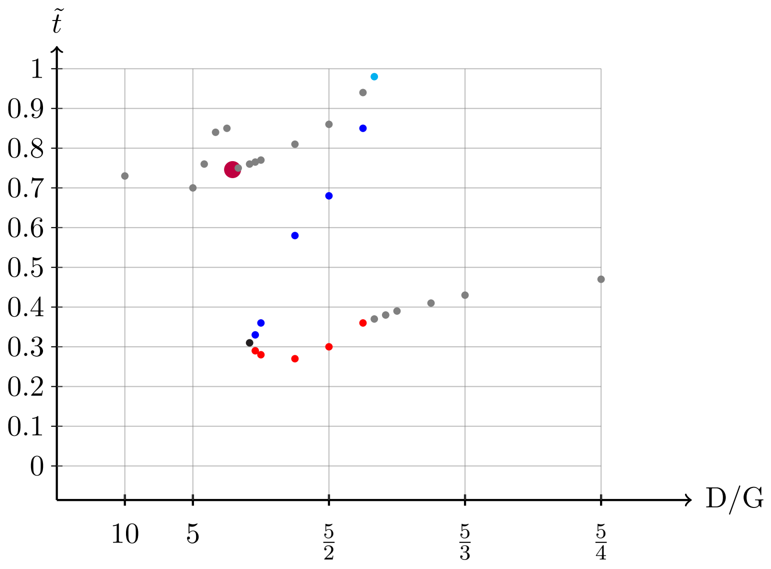


Figure 8.17: $Re = 300$. Bifurcation diagram in the 2-parameter space $(D/G, \tilde{t})$. (Gray) dots mark the creation time for the counter clockwise rotating vortex surviving downstream. (Red) marks the creation point for the closed center-saddle structure. (Blue) marks the annihilation point of the closed center-saddle structure.

From the figure it is seen that as $D/G \to 5/4$ the shedding of the counter clockwise rotating vortex nears $\tilde{t} = 0.5$. This corresponds to perfectly alternating shedding from the topside and bottom side of the cylinder as is the case for the cylinder in free flow. When the cylinder is moved closer to the wall, i.e. $D/G \to 10$ it is seen that the two vortices travelling downstream are shed closer to each other in time.

The (light blue) dot at the parameter values $(D/G_0, \tilde{t}) = (2.22, 0.98)$ marks the point at $Re = 300$ in the $(D/G_0, \tilde{t})$ parameter space where the closed saddle-center structure merges with the creation point for the vortex surviving downstream. That is, it marks the point in parameter space where the structure of the flow goes from the one illustrated in figure 8.16b to the structure illustrated in 8.16c. This transition happens in saddle-center bifurcation as illustrated in figure 8.18.

The (black) dot at roughly $(D/G, \tilde{t}) = (3.52, 0.31)$ marks the point where the closed saddle-center structure appears.

The jump in the data marked by the large (purple) dot at $(D/G, \tilde{t}) \approx (4, 0.75)$ illustrates a point in the parameter space where the clockwise rotating vortex used to define $\tilde{t} = 0$ undergoes a bifurcation of its own. This bifurcation results in the creation point for this vortex changing position instantaneously. Thereby the time between the creation of the clockwise and counter clockwise vortices changes abruptly.

This bifurcation is illustrated in figure 8.19.

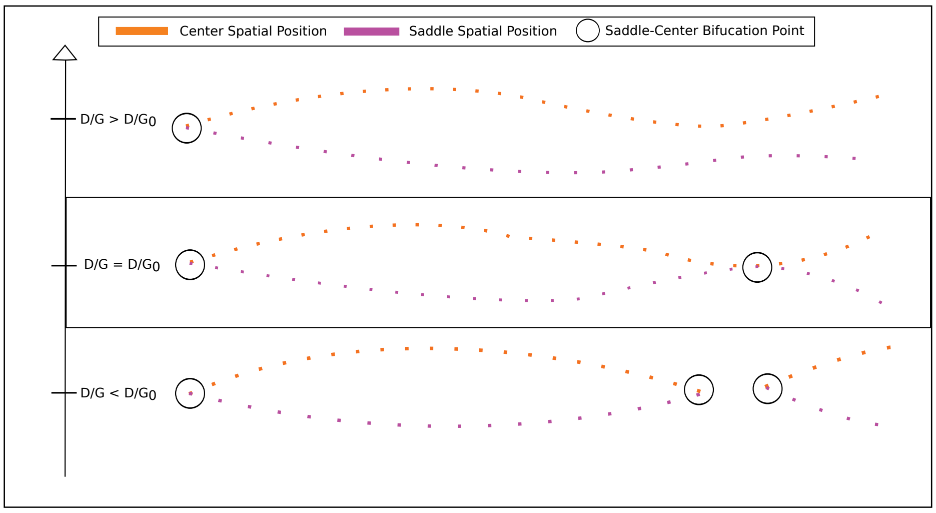
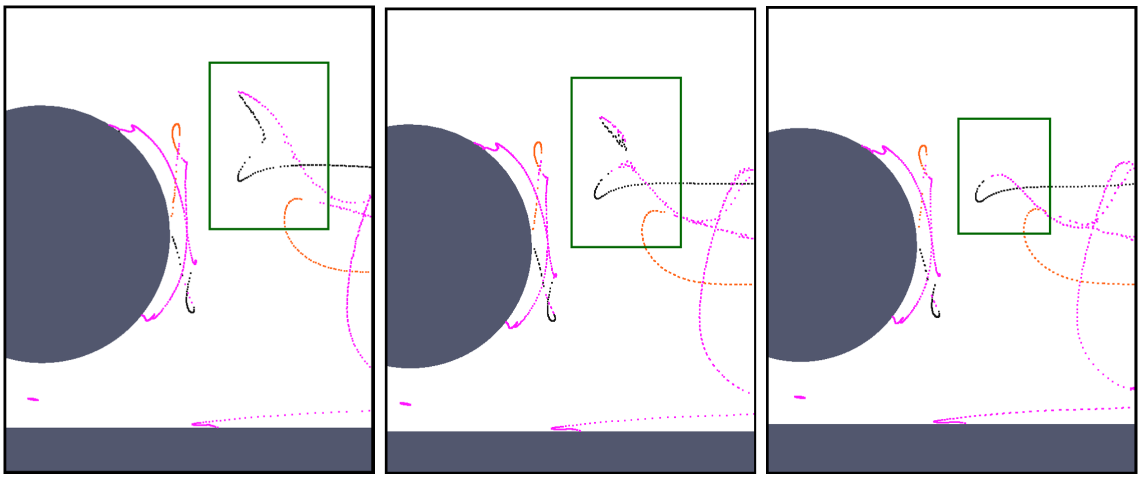


Figure 8.18: Sketch of the center and saddle spatial position at different instances in time, t_i, for three different D/G-values. Before $D/G < D/G_0$, at $D/G = D/G_0$, and after $D/G > D/G_0$ the closed saddle-center structure merges with the creation point for the vortex surviving downstream.



(a) $D/G = 4.0$, the creation point of the clockwise rotating vortex is above the cylinder.

(b) $D/G = 3.87$, A center-saddle pair appears and vanishes above the cylinder and the creation point of clockwise rotating vortex has jumped down behind the cylinder.

(c) $D/G = 3.75$, the creation point of the clockwise rotating vortex is behind the cylinder

Figure 8.19: Reynolds number: $Re = 300$. Illustration of the jump in creation point of the clockwise rotating vortex.

In order to identify at what $(Re, D/G)$-value the closed saddle-center structure marked by blue and red dots in figure 8.17 seizes to exist, the Reynolds number is now lowered in increments of 20, and a series of simulations for different D/G-values performed at each Re.

Lowering Re:

The figures 8.20 through 8.23 show slices of the $(Re, D/G, \tilde{t})$ parameter domain at $Re \in \{280, 260, 240, 220\}$. From these figures it is immediately observed that lowering Re narrows the span of D/G-values for which the closed saddle-center structure exists. It is seen that the point at which the closed structure merges with the vortex surviving downstream stays roughly at $D/G \approx 2.22$. The point at which the closed saddle-center structure emerges reduces in D/G-value as Re is reduced. No evidence that another type of bifurcation then the codimension 1 saddle-center bifurcation happens as Re is lowered has been found. The closed saddle-center structure simply seizes to exist as Re is lowered passed a critical value denoted R_0.

Instead of a different type of bifurcation in the flow it is believed that the disappearance of the structure may be explained by a bifurcation in a mathematical parameter c_1 controlling the saddle-center bifurcation, see equation (2.14).

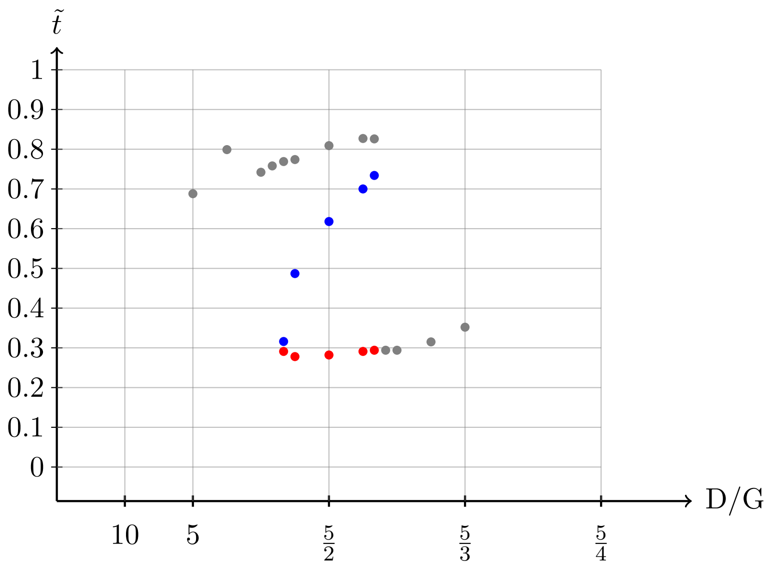


Figure 8.20: Reynolds number $Re = 280$. See caption of figure 8.17 for figure explanation.

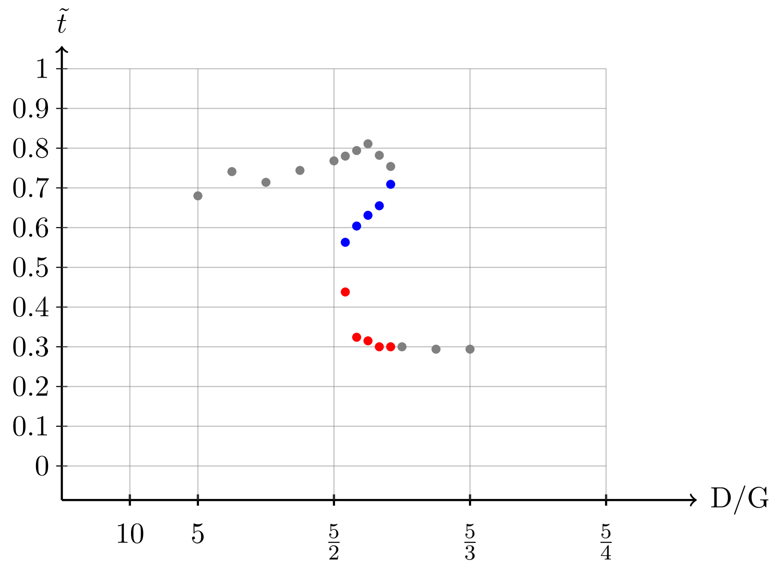


Figure 8.21: Reynolds number $Re = 260$. See caption of figure 8.17 for figure explanation.

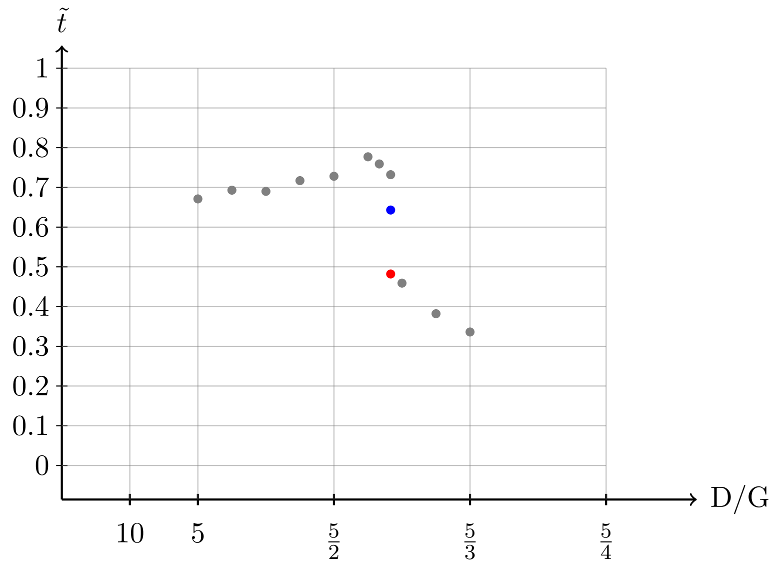


Figure 8.22: Reynolds number $Re = 240$. See caption of figure 8.17 for figure explanation.

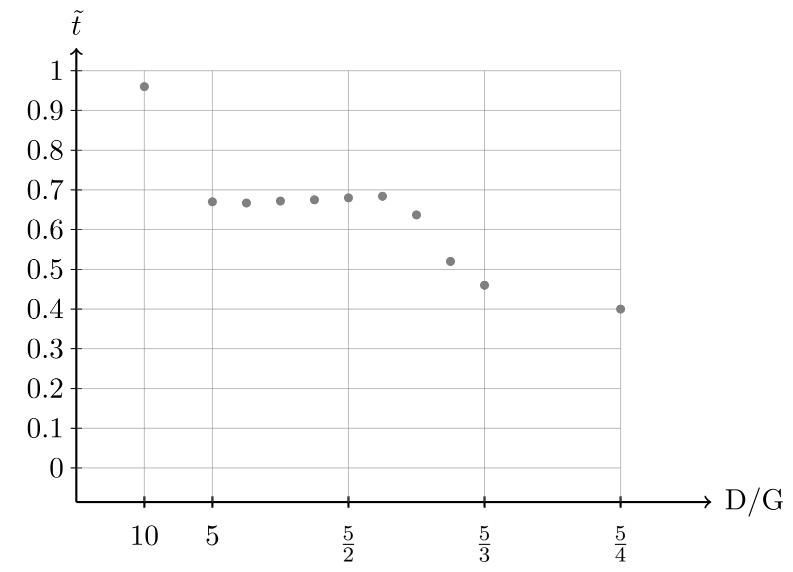


Figure 8.23: Reynolds number $Re = 220$. See caption of figure 8.17 for figure explanation.

A model for the change in c_1 is presented here. Remember that all the data points on the figures 8.20 through 8.23 corresponds to parameter values $(Re, D/G, \tilde{t})$ at which a saddle-center bifurcation occurs in the flow. Each of these bifurcations happen at a given position $(x_{\text{crit}}, y_{\text{crit}})$ in the physical domain away from the wall and cylinder.

From section 2.1, it is known that for a codimension 1 bifurcation, close to $(x_{\text{crit}}, y_{\text{crit}})$ the vorticity ω, may be written through a normal-form transform as,

$$\omega = \frac{1}{2}y^2 + c_1 x + \frac{1}{3}x^3 + O(4), \quad (x_{\text{crit}}, y_{\text{crit}}) = (0,0). \tag{8.3}$$

where c_1 is the mathematical parameter depending on the physical parameters $Re, D/G$ and $\tilde{t}$.

As was shown in section 2.1 the saddle-center bifurcation happens as c_1 crosses zero. Thus bifurcation happens at all points in parameter space where the $c_1(Re, D/G, \tilde{t})$-field attains the value zero, $\{(Re_i, D/G_i, \tilde{t}_i) \mid c_1(Re_i, D/G_i, \tilde{t}_i) = 0\}$.

From figures 8.22 and 8.23 it is seen that the saddle-center structure seizes to exist as Re is lowered from $Re = 240$ to $Re = 220$. The way the path formed by the dots in the diagrams changes from a Z-shape at $Re = 300$ to a flat shape at $Re = 220$ suggests that the disappearance of the closed structure happens through a vertical tangent at the critical Reynolds number value, $Re = Re_0$ and D/G-ratio, $D/G = D/G_0$. Assuming this is the case the $(D/G, \tilde{t})$ bifurcation diagram at Re_0 may be sketched as illustrated in figure 8.24.

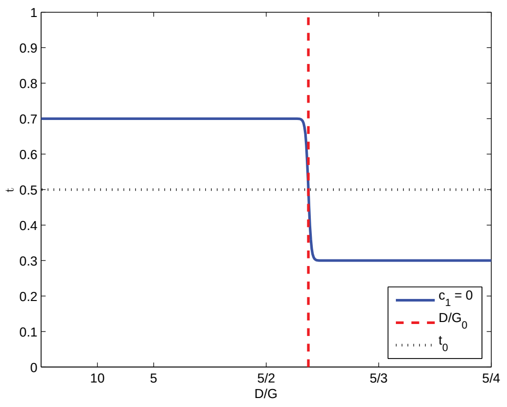
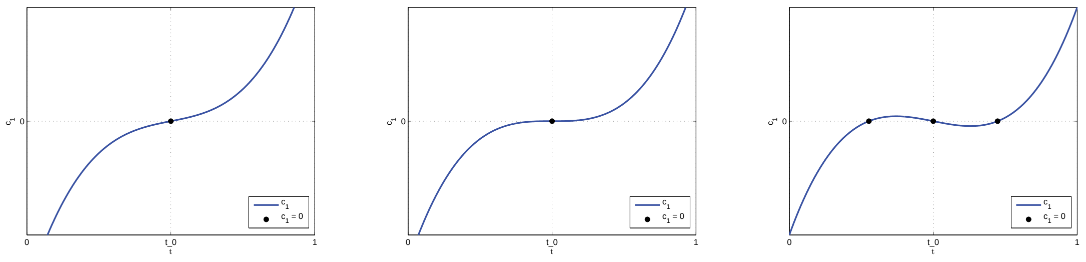


Figure 8.24: Sketch of the $(D/G, \tilde{t})$ bifurcation diagram at $Re = Re_0$. The value of D/G_0 and $\tilde{t}_0$ is marked by dashed lines.

The change in shape suggests that the c_1-parameter undergoes a bifurcation at the set of parameter values, $(Re, D/G, t) = (Re_0, D/G_0, t_0)$. Fixing $D/G = D/G_0$ and consecutively fixing Re at three different values $Re^* \in \{Re_{-1}, Re_0, Re_1\}$ with $Re_{-1} < Re_0 < Re_1$ yields the three different curves for $c_1(Re^*, D/G_0, t)$ sketched in figure 8.25.



(a) $Re^* = Re_{-1}$. (b) $Re^* = Re_0$. (c) $Re^* = Re_1$.

Figure 8.25: Three sketches of slices of the c_1-surface at $(Re^*, D/G_0, \tilde{t})$ for $\tilde{t} \in [0, 1]$.

Figure 8.26b show $c_1(Re_0, D/G_0, \tilde{t})$, i.e. c_1 at the exact Reynolds number where the closed saddle-center structure disappears corresponding to a slice along D/G_0 on figure 8.25. This curve fulfils the following requirements:

$$c_1(Re_0, D/G_0, \tilde{t}_0) = 0, \quad \frac{\partial c_1}{\partial \tilde{t}}|_{(Re_0, D/G_0, \tilde{t}_0)} = 0, \tag{8.4}$$
$$\frac{\partial^2 c_1}{\partial^2 \tilde{t}}|_{(Re_0, D/G_0, \tilde{t}_0)} = 0, \quad \frac{\partial^3 c_1}{\partial^3 \tilde{t}}|_{(Re_0, D/G_0, \tilde{t}_0)} \neq 0.$$

Consider now $c_1(Re^*, D/G_0, \tilde{t})$ at $Re^* = Re_{-1}$ and $Re^* = Re_1$ corresponding to lowering and increasing Re around Re_0. For $Re = 240$ we known that the closed saddle-center structure exists for a narrow range of D/G-values and that for $Re = 220$ the structure no longer exists. This means that at Re_{-1} only a single point exists where $c_1(Re_{-1}, D/G_0, \tilde{t}) = 0$ and for Re_1 three such point exists. This idea is sketched in the figures 8.26a and 8.25c.

The observed change in behaviour of $c_1(Re^*, D/G_0, \tilde{t})$ around Re_0 means that c_1 must also fulfil a fifth condition given by:

$$\frac{\partial}{\partial Re}\left(\frac{\partial c_1}{\partial t}\right)|_{Re_0, D/G_0, \tilde{t}_0} \neq 0. \tag{8.5}$$

With the five conditions presented in (8.4) and (8.5) fulfilled by c_1 at $(Re_0, D/G_0, \tilde{t}_0)$ the model accurately describes the change in c_1 causing the closed saddle-center structure to disappear.

The idea presented above is the simplest explanation of the variation of c_1 as the structure in the bifurcation diagrams 8.20 through 8.23 changes. This model of the dependence of c_1 on $(Re, D/G, \tilde{t})$ around the bifurcation point may be tested by identifying $(Re_0, D/G_0, \tilde{t}_0)$ precisely and evaluating the five requirements at these values.

$Re < Re_0$: At $Re = 220$ and below the closed saddle-center structure no longer appears behind the cylinder.

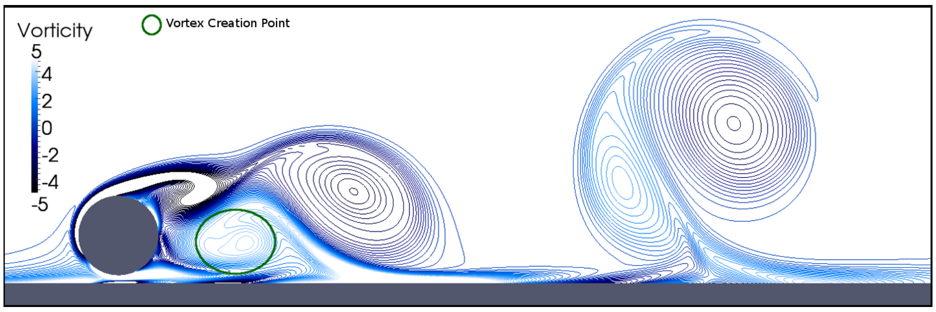
8.3 Vortex Creation Point Jumping Downstream

Another, more structurally significant, bifurcation is a jump in the creation point for the counter clockwise rotating vortex surviving downstream. This bifurcation was discovered for the cylinder very close to the wall $D/G = 10$ and marks a much larger change in the flow structure then the previously studied bifurcation.

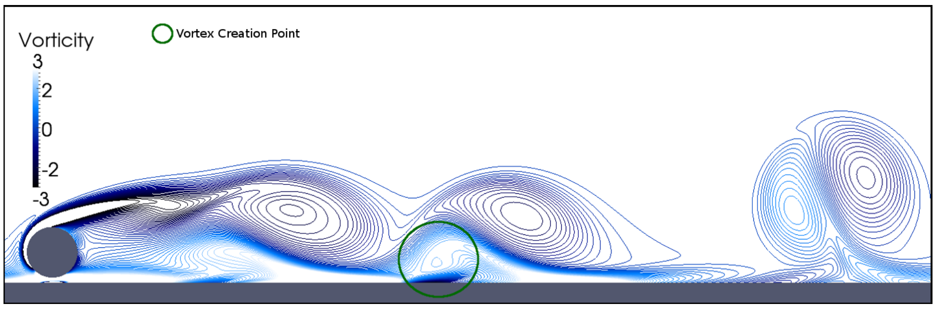
The search for the bifurcation was motivated by the findings presented in figure 8.7a. Here a significant jump in x-coordinate of the creation point for the counter clockwise rotating vortex is between $Re = 200$ and $Re = 220$.

At $Re > 220$ the counter clockwise rotating vortex created behind the cylinder is found to survive the full length downstream. The data in figure 8.7a suggests that for $Re \leq 200$ this may no longer be the case. The counter clockwise rotating vortex surviving downstream in now created near the wall at more then four cylinder diameters downstream.

To clearly illustrate the different positions of the generation of the vortex figure 8.26 presents vorticity contours at the point in time when the vortex is created at $Re = 300$ and $Re = 160$. Here it may be seen that for $Re = 300$ the lower vortex is created close to the cylinder whereas for $Re = 160$ the creation point of the lower vortex has moved considerably downstream.



(a) *Re* = 300. The vortex created right behind the cylinder survives downstream.



(b) *Re* = 160. The vortex created behind the cylinder quickly dies out and a new vortex is formed near the wall further down stream.

Figure 8.26: $D/G = 10$. Vorticity contours at the point in time where the counter clockwise rotating vortex is created near the wall. The green circles marks the approximate position of the creation of the vortices. Domain dimensions: (a) $(x, y) \in [-1.5, 10] \times [0, 3.5]$, (b) $(x, y) \in [-1, 17] \times [0, 5.5]$.

In order to investigate this change in structure the first step was to localise the bifurcation to $Re \in [210, 220]$. The model problem was then solved for, $Re \in \{210, 212.5, 215, 217.5, 220\}$ and the traces of the centres and saddle-points in time were calculated. These are shown in figures 8.27 and 8.28.

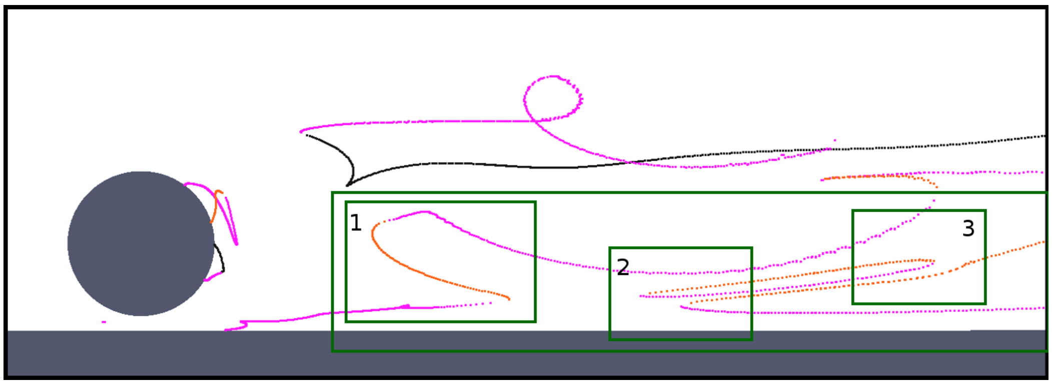
Consider first figure 8.27a. In the green square marked by 1 it is seen that a counter clockwise rotating vortex is created behind the cylinder in a saddle-center bifurcation. A short distance downstream it disappears again in another saddle-center bifurcation. Further downstream, in the area marked by the green square with the number 2, another two extrema are created in saddle-center bifurcations. These travel downstream until one of the extrema is swallowed in a saddle-center bifurcation and the last surviving vortex continues downstream.

In contrast, in figure 8.28b it is seen that the counter clockwise rotating vortex created behind the cylinder survives all the way downstream.

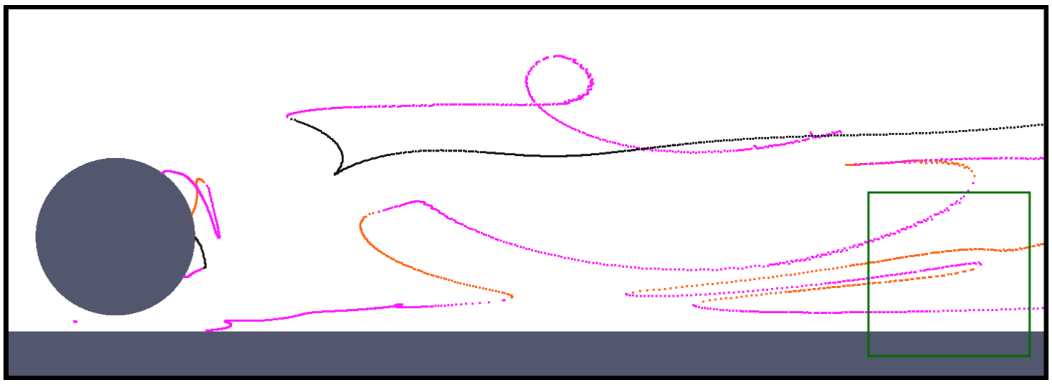
The bifurcation causing the change in structure is captured at $Re = 217.5$ in figure 8.28a highlighted by a green square. From 8.28a to 8.28b it is seen that the change happens in a saddle-center bifurcation where the creation point of the vortex surviving downstream merges with the annihilation point for the vortex created behind the cylinder.

It should be noted that no such jump in the creation point was observed for the clockwise rotating vortex which survives downstream. That is, that vortexes creation point moves smoothly downstream with lowering *Re*.

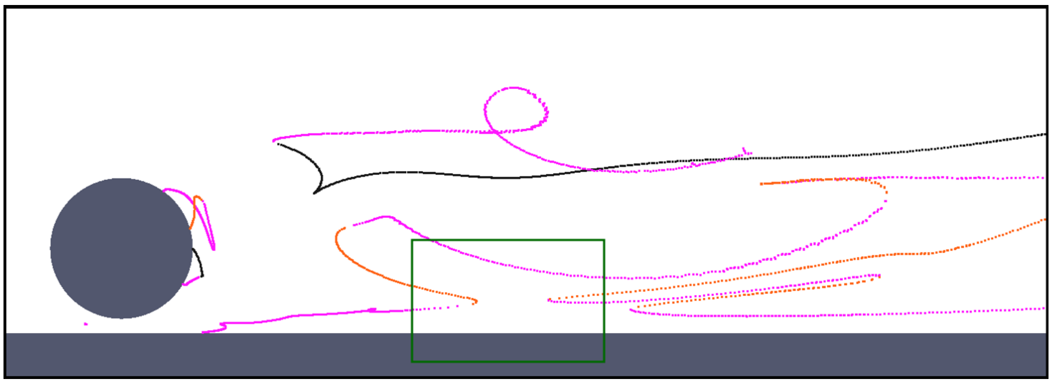
Codimension 2 Bifurcation: Another change of structure may be observed by comparing the area marked by the green square with number 3 in figure 8.27a and the area marked by a green square in figure 8.27b. This change in structure is of little importance on the overall flow but interesting because its the only none saddle-center bifurcation observed which is localised away from the wall or cylinder. This bifurcation is most likely of codimension 2 as will be shown here.



(a) $Re = 210$.

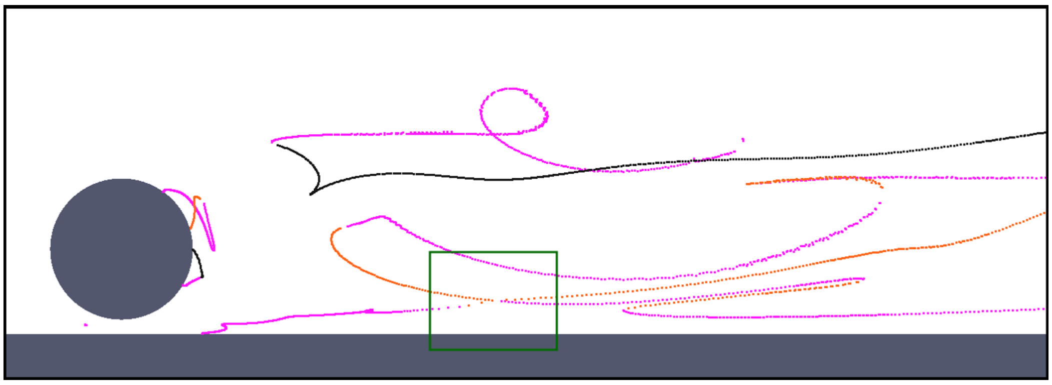


(b) $Re = 212.5$.



(c) $Re = 215$.

Figure 8.27: $D/G = 10$. The center of the cylinder is positioned at $(x_c, y_c) = (0, \frac{D}{2} + G)$. Domain dimensions: $(x, y) \in [-1, 6] \times [0, 2.25]$.


(a) $Re = 217.5$.

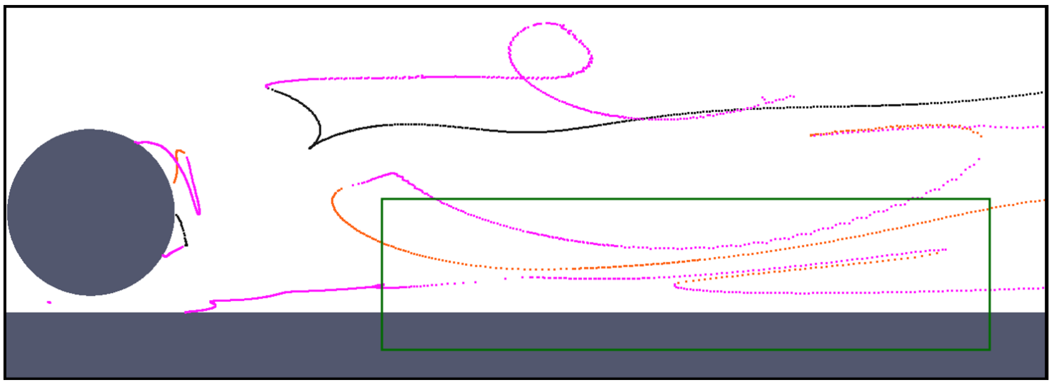

(b) $Re = 220$.

Figure 8.28: $D/G = 10$. The center of the cylinder is is positioned at $(x_c, y_c) = (0, \frac{D}{2} + G)$. Domain dimensions: $(x, y) \in [-1, 6] \times [0, 2.25]$.

Remember that a bifurcation of codimension 2 means that the structural change in the flow depends on two mathematical parameters, e.g. c_1 and c_2. In [2, Section III] Brøns derives all possible codimension 2 bifurcations for the stream function which fulfil the requirements given in **Theorem 1**[29]. Here Brøns presents two bifurcation diagrams containing all possible bifurcations. These diagrams are provided in figure 8.29.

[29]Remember that stream function and vorticity are equivalent.

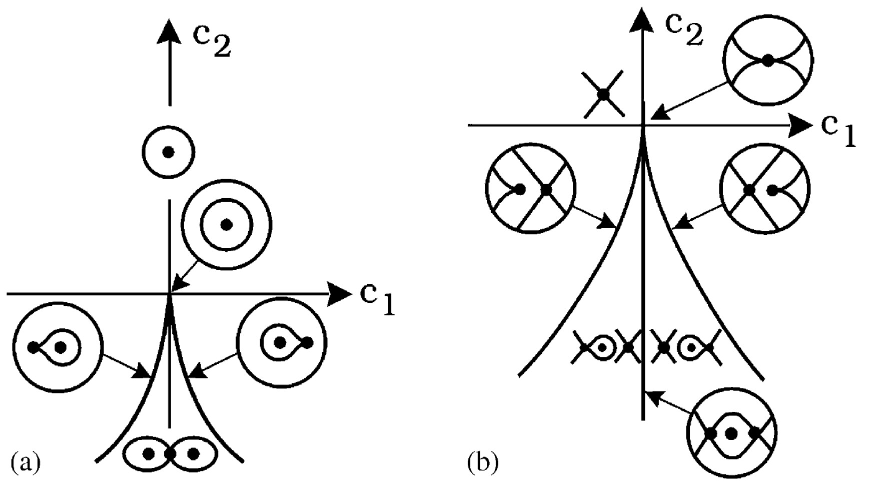


Figure 8.29: All possible codimension 2 bifurcations for $(\dot{x}, \dot{y}) = (-\omega_y, \omega_x)$ where $\omega = \sum_{i,j} a_{i,j} x^i y^j$. A black dot floating alone correspond to a center. A black dot with four lines emanating from it corresponds to a saddle. The figure is borrowed from [2, Figure 3.3], courtesy of Brøns.

By inspecting figure 8.29a and equating c_1 with Re and c_2 with t it is possible to identify the bifurcation which is most likely happening as Re is varied from $Re = 210$ to $Re = 212.5$ across some $Re_0 \Leftrightarrow c_1 = 0$.

From looking at the area contained in the green box marked by 3 in 8.27a one observes two centres and a saddle move downstream until the topmost center and the saddle merges in a saddle-center bifurcation leaving the bottom center travelling downstream. This event corresponds to $Re < Re_0 \Leftrightarrow c_1 < 0$ and $c_2 = t$ growing in figure 8.29a. By placing a line parallel to the c_2 axis for $c_1 < 0$ on this figure the bifurcation where two centres and a saddle turns to a single center as the vortex travels downstream from the cylinder may be seen.

For the bifurcation happening in the green square on 8.27.b a similar analysis may be performed, only this time $Re > Re_0 \Leftrightarrow c_1 > 0$. Hence the two centres and the saddle passes across the bifurcation curve in figure 8.29.a making the lower center merge with the saddle and leaving the upper center travelling downstream.

By inspecting figure 8.29 in detail it may be seen that no other type of codimension 2 bifurcation fits this behaviour. Hence it can be concluded that this is the type of bifurcation happing as Re is varied across Re_0. At $Re = Re_0$ the bifurcation corresponds to the known pitchfork bifurcation where two centres and a saddle merge to form a single center. The three situations, $Re < Re_0, Re = Re_0$ and $Re > Re_0$, are sketched in figure 8.30 for clarity.

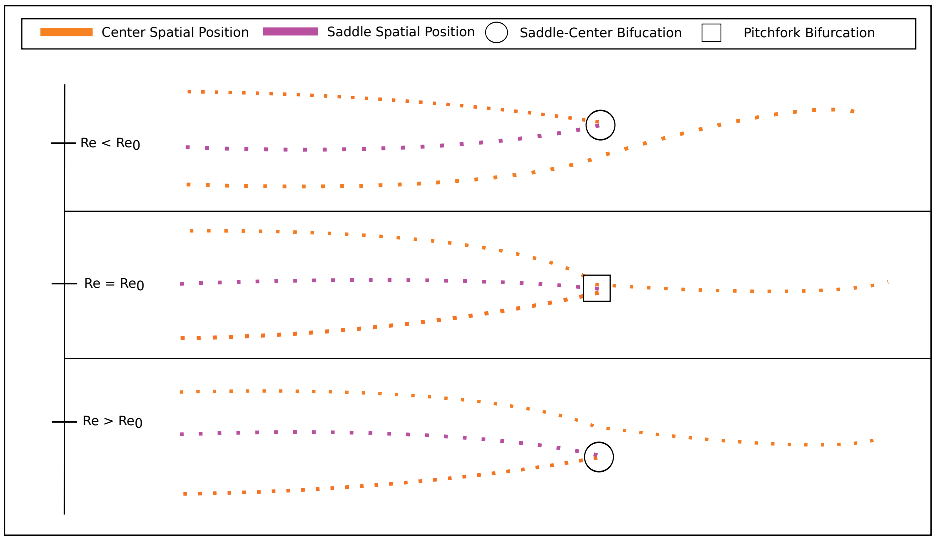


Figure 8.30: Sketch of the three different vortex and saddle-point traces at $Re < Re_0$, $Re = Re_0$, $Re > Re_0$ for the codimension 2 bifurcation.

8.4 Uncertainty Quantification

This section presents selected results obtained in the investigation of applying the principles of the Stochastic Collocation Method presented in section 2.4. The SCM is used to obtain a functional relationships between an uncertain input parameter and quantities of the flow as well as predicting the mean and variance of the quantities. In all cases the Reynolds number is considered as the random variable.

Initially the problem of the cylinder in free flow is considered. This is done to compare results obtained using the SCM to published results and results obtained from other simulations performed for this project. First Re is assumed to be uniformly distributed and the quantities of pressure drag on the cylinder and the Strouhal number are considered. Afterwards Re is assumed to be normally distributed, and the same quantities are considered.

Secondly the problem of the cylinder near the moving wall is considered. Here the Re is considered uniformly distributed and the pathways of the clockwise rotating vortex surviving downstream and its magnitude are the quantities of interest.

The reason for choosing the uniform and normal distributions is that they in the authors mind reflect two general cases in real world application. The uniform distribution corresponds to a design choice, i.e. in the process of designing a system it is possible to choose some design parameter in the interval $[a, b]$. In this context it is very valuable to obtain information about how a systems behaviour depends on the choice without trying all configurations. The normal distribution on the other hand reflects uncertainties in measurements and/or knowledge of the system parameter.

8.4.1 Cylinder in free flow

The first investigation of the application of UQ is performed on a cylinder in an open flow, see figure 1.1a. For this problem Henderson [10], has presented results which were also used to verify the simulations in chapter 7. These results are used here to compare with results obtained using UQ.

In order to apply the SCM to the problem the approach outlined in section 2.4.4 is followed.

Uniformly distributed Reynolds number: First the Reynolds number is taken to be a random variable following a uniform distribution on the interval $Re \in \mathcal{U}[50, 600]$. This gives rise to the PDF, $f_{X,u} = \frac{1}{550}$. As presented in section 2.4.3 the uniform distribution has the Legendre polynomials as its gPC basis. With the distribution and gPC basis known the next step is to choose a set of collocation nodes, $\Theta_M = \{X_j\}_{j=1}^{M}$, which in this case are determined by identifying the zeros of the $(M + 1)$'th Legendre polynomial.

Three cases for different polynomial order, $M \in [1, 2, 4]$, were considered in order to observed the change in the gPC approximation for the quantities of interest. The collocation nodes in the standard interval $\tilde{Re} \in [-1, 1]$ and the corresponding interval of interest $Re \in [50, 600]$ are presented in table 8.1,

Basis Order:	**Nodes**:	
	$Re \in [50, 600]$	$\tilde{Re} \in [-1, 1]$
First:	166.2287	-0.5774
	483.7713	0.5774
Second:	111.9859	-0.7746
	325	0
	538.0141	0.7746
Fourth:	75.8005	-0.9062
	176.9209	-0.5385
	325	0
	473.0791	0.5385
	574.1995	0.9062

Table 8.1: The collocation points for the uniformly distributed random variable $Re \in [50, 600]$ for different approximation orders. The linear transformation needed to go from the standard interval $\tilde{Re} \in [-1, 1]$ to an arbitrary interval $Re \in [a, b]$ is $Re = \frac{b-a}{2}\tilde{Re} + \frac{b+a}{2}$.

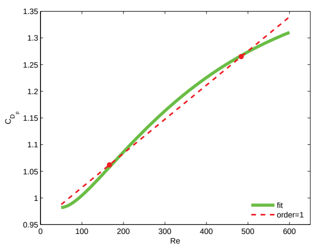
Using the collocation values for Re the problem of the cylinder in an open flow has been simulated until periodic shedding has been achieved. The Strouhal number, St and mean pressure drag coefficient C_{D_p} have then been calculated. The values are provided in table 8.2.

Basis Order:	St	C_{D_p}
First:	0.191	1.062
	0.225	1.265
Second:	0.174	1.021
	0.216	1.180
	0.227	1.293
Fourth:	0.155	0.991
	0.194	1.068
	0.216	1.180
	0.224	1.262
	0.229	1.298

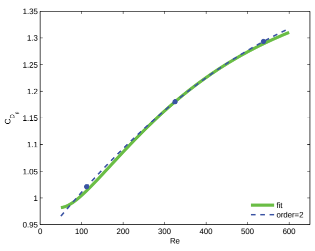
Table 8.2: The St and C_{D_p} at the collocation points for the uniformly distribution random variable $Re \in [50, 600]$ presented in table 8.1.

Based on the values presented in table 8.2 a polynomial expansion in the Reynolds number for each quantity has been calculated as described in the paragraph on interpolation in section 2.4.4.

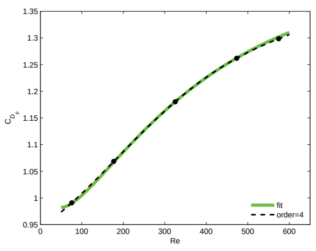
Figures 8.31 and 8.32 show a plot of the polynomial expansions $P_{N,D}[C_{D_p}]$ and $P_{N,D}[St]$ for the C_{D_p} and St respectively using a first, second and fourth order basis. In figure 8.31 the fit created by Henderson [7] based on 15 data points is plotted along with $P_{N,D}[St]$ to illustrate the excellent agreement.



(a) First order UQ based polynomial expansion.



(b) Second order UQ based polynomial expansion.

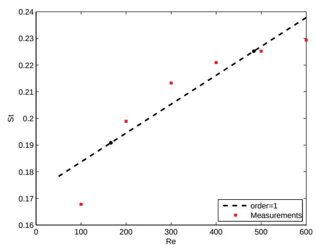


(c) Fourth order UQ based polynomial expansion.

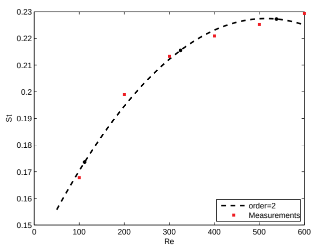
Figure 8.31: Graph of the fit presented by Henderson [7] provided in (7.1) for C_{D_p} based on 15 data points and the UQ based polynomial expansion for C_{D_p} based on 2,3 and 5 data points respectively.

From figure 8.31 it is seen that atleast in the eyeball norm the fit and the fourth order UQ approximation agree very well.

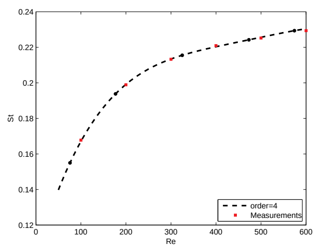
In figure 8.32 the data for the St calculated for $Re \in \{100, 200, 300, 400, 500, 600\}$ presented in section 7.2.1 in table 7.5 is plotted along side the UQ polynomial approximation.



(a)



(b)



(c)

Figure 8.32: Graph of the UQ based polynomial expansion for St based on 2,3 and 5 datapoints along with the Strouha number calculated from simulations for $Re \in \{100, 200, 300, 400, 500, 600\}$.

Here very good agreement between the UQ approximation to fourth order and St data is observed.

As shown in equation (2.63) the statistical mean of the approximation $P_{N,D}[g]$ is given by the first coefficient in the polynomial expansion. As for the variance equation (2.64) shows that this is given by the sum of squares of the rest of the expansion coefficients multiplied by the normalization constant for the orthogonal polynomials. Using this, the mean and variance has been calculated for both the polynomial expansions for the St and the C_{D_p}. The mean and variance of C_{D_p} is compared to those obtained using the fit by Henderson. The results are presented in table 8.3.

Mean:	$\mu_{C_{D_p}}$	μ_{St}
First Order:	1.1636	0.2080
Second Order:	1.1675	0.2071
Fourth Order:	**1.1646**	0.2069
$\mu_{\text{Henderson}}$:	**1.1646**	
Variance:	$\sigma^2_{C_{D_p}}$	σ^2_{St}
First Order:	0.0103	2.9559e-04
Second Order:	0.0104	4.5606e-04
Fourth Order:	**0.0104**	5.1279e-04
$\sigma^2_{\text{Henderson}}$	**0.0105**	

Table 8.3: Mean and variance for C_{D_p} and St calculated using the UQ polynomial expansions for the different expansion orders. Also the mean and variance for C_{D_p} obtained by calculating it from the fit presented by Henderson in [7].

From table 8.3 it is clearly seen that there are excellent agreement for both the mean and variance for the pressure drag coefficient between the results obtained by Henderson using 15 data points and the results obtained using UQ with only 5 data points.

Normally distributed $Re \in \mathcal{N}(300, 50)$: In order to test the performance of UQ when assuming the underlying random variable is normally distributed the same tests as for the uniform distribution presented above have been performed[30].

The normal distribution gives rise to the PDF, $f_{X,G} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, and the Hermite polynomials or the gPC basis, see section 2.4.3. The collocation nodes are in this case determined by using the Golub-Welsch algorithm, which may be found in [21, Section 4.6.2].

Like the uniformly distributed case, three different basis orders, $M \in [1, 2, 4]$, were considered. The collocation nodes using the standard distributions $\tilde{Re} \in \mathcal{N}(0, 1)$ and the corresponding nodes in $Re \in \mathcal{N}(300, 50)$ are presented in table 8.4,

The model problem has been solved at the collocation nodes, and the Strouhal number, St and mean pressure drag coefficient C_{D_p} have been calculated. The values are provided in table 8.5.

[30]As noted in section 6.5 it is actually wrong to use normal distribution for the Re as, $Re \geq 0$. However for the given choice of mean and variance the probability of Re attaining a non-physical value is $\int_{-\infty}^{0} f_{X,G}(x)dx \approx 10^{-9}$.

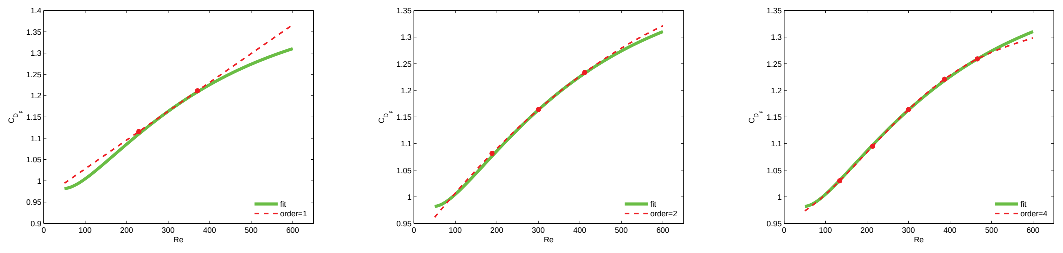
Basis Order:	**Nodes**:	
	$Re \in \mathcal{N}(300,50)$	$\tilde{Re} \in \mathcal{N}(0,1)$
First:	229.2893	-1.4142
	370.7107	1.4142
Second:	188.1966	-2.2361
	300	0
	411.8034	-3.3166
Fourth:	134.1688	-1.7321
	213.3975	-0.5385
	300	0
	386.6025	1.7321
	465.8312	3.3166

Table 8.4: The collocation points for the normally distribution random variable $Re \in \mathcal{N}(300,50)$ for different approximation orders. The nodes $Re \in \mathcal{N}(\mu,\sigma)$ are obtained by the transformation $\tilde{Re} \in \mathcal{N}(0,1)$ is $Re = \mu + \sigma\tilde{Re}$.

Basis Order:	St	C_{D_p}
First:	0.203	1.116
	0.218	1.211
Second:	0.195	1.081
	0.212	1.164
	0.221	1.234
Fourth:	0.181	1.030
	0.200	1.095
	0.212	1.164
	0.219	1.221
	0.224	1.259

Table 8.5: The St and C_{D_p} at the collocation points for the for the normally distribution random variable $Re \in \mathcal{N}(300,50)$ presented in table 8.4.

Polynomial expansions in the Reynolds number for each quantity have been calculated for each basis order and are presented in the figures 8.33 and 8.34 for C_{D_p} and St respectively. In figure 8.33 the fit created by Henderson [7] based on 15 data points is plotted along with $P_{N,D}[St]$.



(a) First order UQ based polynomial expansion.

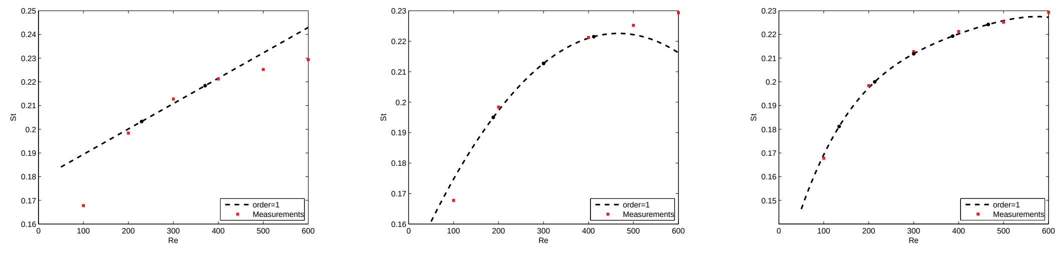
(b) Second order UQ based polynomial expansion.

(c) Fourth order UQ based polynomial expansion.

Figure 8.33: Graph of the fit presented by Henderson [7] provided in (7.1) for C_{D_p} based on 15 data points and the UQ based polynomial expansion for C_{D_p} based on 2,3 and 5 data points respectively.

Just as for the uniformly distributed Re it is seen from figure 8.33 that the fourth order gPC expansion $P_{N,D}[C_{D_p}]$) agrees well with the fit by Henderson.

In figure 8.34 the data for the St presented in section 7.2.1 in table 7.5 is plotted along side the UQ polynomial approximation.



(a)

(b)

(c)

Figure 8.34: Graph of the UQ based polynomial expansion for St based on 2,3 and 5 datapoints along with the Strouhal number calculated from simulations for $Re \in \{100, 200, 300, 400, 500, 600\}$.

Again just as for the uniform case very good agreement between the UQ approximation to fourth order and St data is observed.

The mean and variance have been calculated for both the polynomial expansions for the St and the C_{D_p}.

Using the fit by Henderson, $f_H(x)$ it is possible to calculate a mean and variance assuming $Re \in \mathcal{N}(300, 50)$ by,

$$\mu_{\text{Henderson}} = \int_{50}^{1000} f_H(x) \frac{1}{50\sqrt{2\pi}} e^{-\frac{(x-300)^2}{5000}} dx, \tag{8.6}$$

$$\sigma_{\text{Henderson}} = \int_{50}^{1000} (\mu_N - f_H(x))^2 \frac{1}{50\sqrt{2\pi}} e^{-\frac{(x-300)^2}{5000}} dx. \tag{8.7}$$

The reason for the limited integration interval is that $f_H(x)$ is only valid in this range. This of course leads to slightly wrong mean and variance. However as $f_H(x) \simeq 1$ and the value of the integral of the normal PDF outside this interval is $\int_{-\infty}^{50} f_{X,G}(x)dx + \int_{1000}^{\infty} f_{X,G}(x)dx \approx 2.866 \cdot 10^{-7}$, the error is very small. The results are presented in table 8.6.

Mean:	$\mu_{C_{D_p}}$	μ_{St}
First Order:	1.1636	0.2108
Second Order:	1.1627	0.2118
Fourth Order:	1.1620	0.2111
$\mu_{\text{Henderson}}$:	**1.1614**	
Variance:	$\sigma^2_{C_{D_p}}$	σ^2_{St}
First Order:	0.00114	2.865e-05
Second Order:	0.00116	3.6697e-05
Fourth Order:	0.00133	3.2317e-05
$\sigma^2_{\text{Henderson}}$	**0.00122**	

Table 8.6: Mean and variance for C_{D_p} and St calculated using the UQ polynomial expansions for the different expansion orders. Also the mean and variance for C_{D_p} obtained by calculating it from the fit presented by Henderson in [7].

From table 8.6 it is seen that the UQ approximation of the mean seems to converge towards the mean obtained from $f_H(x)$. However neither the mean or variance agrees completely with those obtained from $f_H(x)$. This suggests, either that a higher order basis is needed, that the fit by Henderson is inaccurate or that there are a problem with the UQ method.

An Issue of Measuring Quantities: During the work behind the results presented above a potential problem with using the SCM approach was encountered. The fundamental idea of UQ is that very few samples are needed to obtain highly accurate statistics for a given quantity. The fact that only a few samples are used appears to raise an issue however.

If the process of measuring a given quantity contains some uncertainty in itself, this uncertainty has the potential of impacting the accuracy of the SCM method. If the uncertainty in the measuring is in itself random the fact that only very few samples are used may be an issue compared to e.g. the Monte Carlo method where the large number of samples should balance the random uncertainty. If the uncertainty is skewed to one side however the problem will be the same for either methods.

A concrete example of uncertainty in measuring a quantity is the measurement of the Strouhal number. In order to calculate St the vortex shedding period must be calculated. The period is estimated by considering a sequence of equidistant snap shots in time. The discretization in time inevitably introduces an uncertainty in measuring the period, which translates to an uncertainty in measuring St. Due to the high number of data sets over a single period in the present work, the error in measuring the St is very small, however one can imagine that this is not always the case. In order to illustrate the problem the Strouhal number obtained at each of the five data points for the forth order gPC expansion have been randomly perturbed by up to plus or minus three percent five separate times,

$$St_{\text{perturbed}} = St \cdot (1 + 0.03\ r), \quad r \in \mathcal{U}[-1, 1]. \tag{8.8}$$

The resulting five gPC expansions $P_{4,i}[St_{\text{perturbed}}], i \in \{1, 2, 3, 4, 5\}$ along with the expansion without perturbations are plotted in figure 8.35.

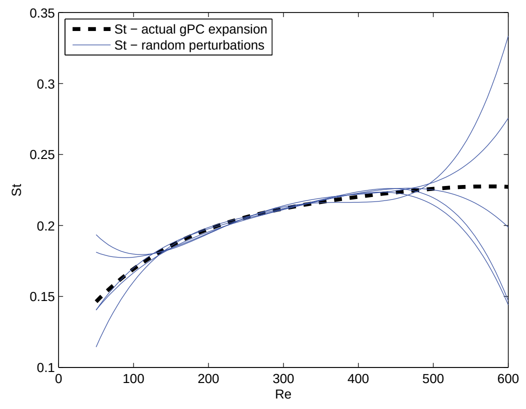


Figure 8.35: Fourth order gPC expansions for St shown in figure 8.34c along with five random perturbations of the form $St_{\text{perturbed}} = St \cdot (1 + 0.03\ r)$, where $r \in \mathcal{U}[-1, 1]$.

From the figure it is immediately apparent how much the expansions differ in the two ends of the interval $Re \in [50, 600]$ when St is perturbed. This shows that the gPC expansion at fourth order is very sensitive to the accuracy of the measurement of St which may lead to false conclusions if care is not exercised. Whether this difference carries over to the statistics have also been investigated. The means and variances obtained from the expansions are presented in table 8.7.

Mean:	μ_{St}	σ^2_{St}
St:	0.2111	$\mathbf{3.2317 \cdot 10^{-5}}$
St_1:	0.2110	$3.6013 \cdot 10^{-5}$
St_2:	0.2124	$\mathbf{5.0570 \cdot 10^{-5}}$
St_3:	0.2119	$3.8944 \cdot 10^{-5}$
St_4:	0.2112	$4.6501 \cdot 10^{-5}$
St_5:	0.2107	$3.9928 \cdot 10^{-5}$

Table 8.7: Mean and variance for the actual St measurements along with five random perturbations calculated using the gPC expansion of fourth order.

From the table it may be seen that the impact of adding the random noise is small for the mean value, however for the variance there is up to a factor of 1.5 difference in the results.

8.4.2 Cylinder Near Moving Wall

This section presents results of applying the SCM to the problem of the cylinder near the moving wall at a fixed D/G-ratio. Here statistics are presented for the pathway followed by the clockwise rotating vortex surviving downstream and its magnitude. The uncertain parameter is taken to be the Reynolds number. Re is assumed distributed uniformly on the interval $Re \in \mathcal{U}[150, 240]$. The lower bound on Re was chosen to make sure that the vortex exists for all Re-values.

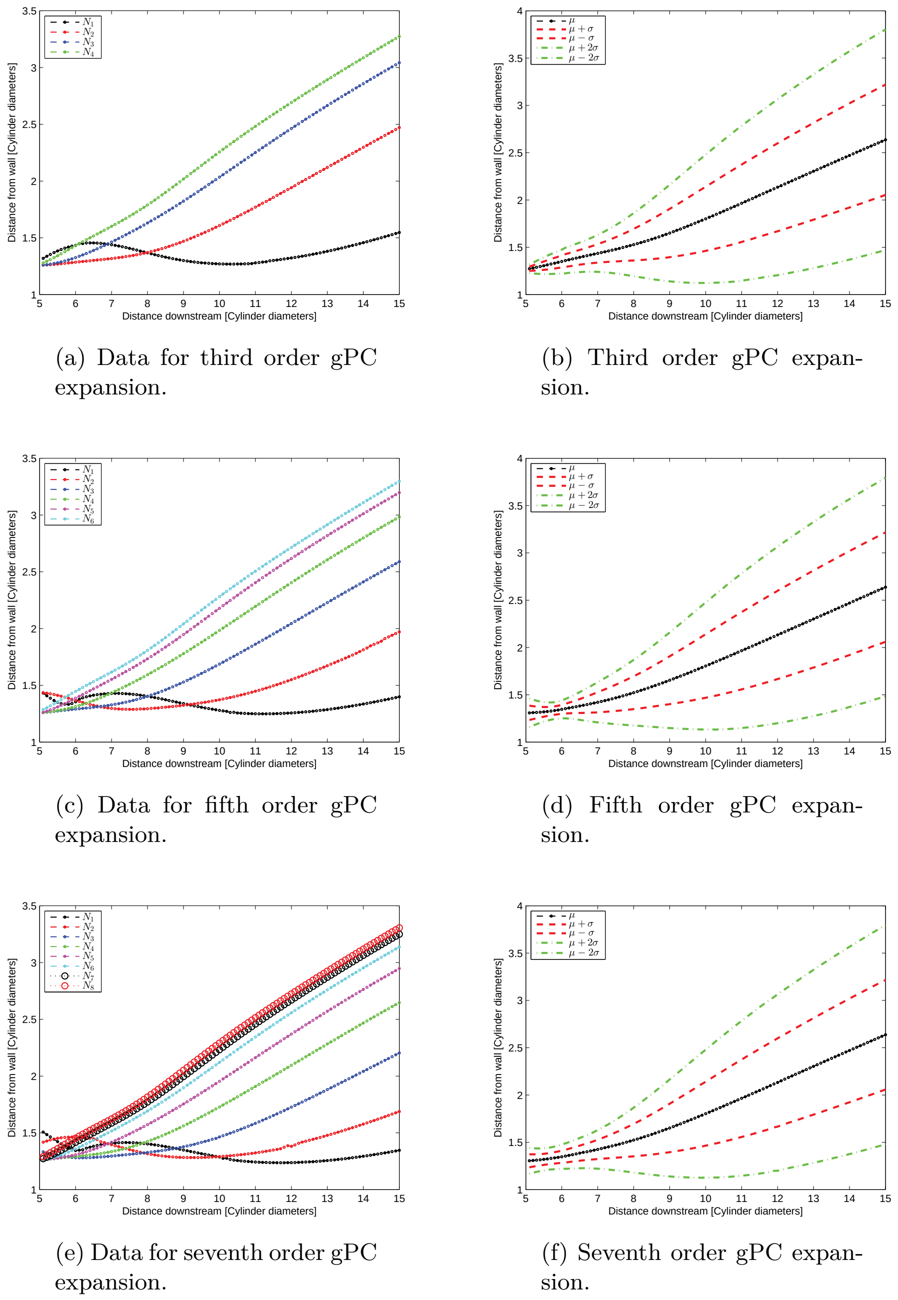
Three different investigations using increasing expansion orders for the gPC basis have been performed. These are third, fifth and seventh order expansions and the collocation nodes are given in table 8.8.

Basis Order:	**Nodes**:	
	$Re \in [150, 240]$	$\tilde{Re} \in [-1, 1]$
Third:	156.2489	-0.8611
	179.7009	-0.3400
	210.2991	0.3400
	233.7511	0.8611
Fifth:	153.04	-0.9325
	165.25	-0.6612
	184.26	-0.2386
	205.74	0.2386
	224.75	0.6612
	236.96	0.9325
Seventh:	151.7870	-0.9603
	159.1500	-0.7967
	171.3510	-0.5255
	186.7454	-0.1834
	203.2546	0.1834
	218.6490	0.5255
	230.8500	0.7967
	238.2130	0.9603

Table 8.8: The collocation points for the uniformly distributed random variable $Re \in [150, 240]$ for different approximation orders.

For each node the field data at the trace points of the clockwise rotating vortex was stored. All the data was then interpolated onto one hundred equidistant x-coordinate values downstream of the cylinder, in the interval $x \in [5, 15]$.

Pathway followed: For each of the three UQ approximations the gPC approximation to the y-coordinate of the vortex center was calculated at each x-coordinate. That is $y(Re)|_{x=\text{fixed}}$ was approximated by $P_{N,D}[y]$. Using the coefficients of $P_{N,D}[Y]$ allowed the calculation of the mean path followed by the vortex and the variance of the path. The mean of the path along with the mean plus and minus one and two standard deviations are shown on figure 8.36 (b),(d) and (f).



(a) Data for third order gPC expansion.

(b) Third order gPC expansion.

(c) Data for fifth order gPC expansion.

(d) Fifth order gPC expansion.

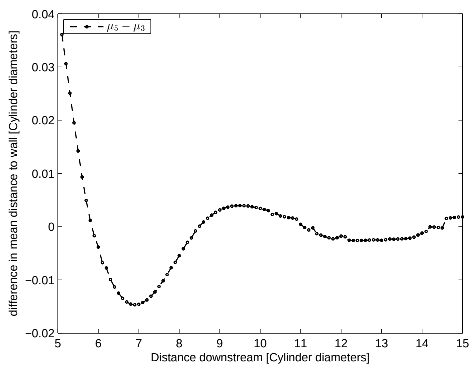
(e) Data for seventh order gPC expansion.

(f) Seventh order gPC expansion.

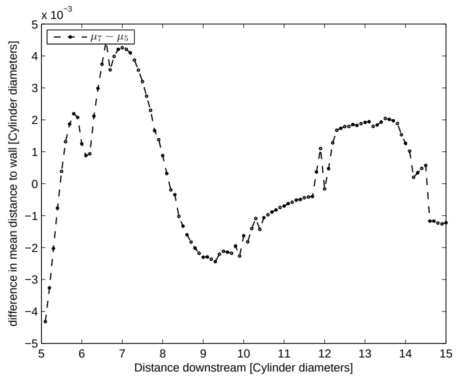
Figure 8.36: (a),(c),(e): Vortex pathway data used for UQ approximation. (b),(d),(f): Mean pathway followed by the vortices and mean path plus and minus one and two standard deviations.

Figure 8.36 (a),(c) and (e) show the pathway data used to construct the gPC expansions.

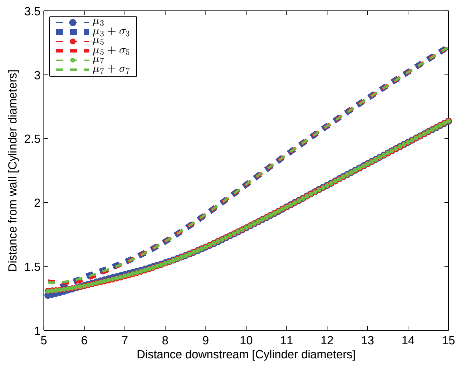
From figure 8.36 (b),(d) and (f) it is hard to see any significant difference in the mean or variance for the different expansions. Therefore the differences in mean between the third and fifth and the fifth and seventh order expansions are plotted in figures 8.37a and 8.37b. Also all three mean value traces are plotted on top of each other in figure 8.37c and a zoom of $x \in [5, 8]$ is provided in figure 8.37d.



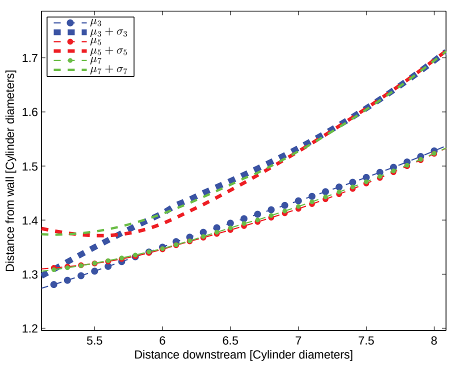
(a) Difference between the mean calculated using the fifth and the third order gPC expansions. i.e. $\mu_5 - \mu_3$.



(b) Difference between the mean calculated using the seventh and the fifth order gPC expansions. i.e. $\mu_7 - \mu_5$.



(c) Plot of the mean and mean plus one standard deviation obtained from the third, fifth and seventh order gPCs expansions.



(d) Zoom of part of the domain of figure (c).

Figure 8.37

From figure 8.37c it may be seen that there is good agreement between the means and variances calculated using either of the three different SCM approximations past $x = 8$. It can be seen from figure 8.37d that the mean and variance deviates between the different approximations in the interval $x \in [5, 7.5]$. This suggests that at least for the fifth order approximation the mean and variance have not converged in this part of the domain and thus a higher order basis is needed to capture the statistics accurately.

In order to investigate if the gPC expansions have converged it is also possible use the expansion coefficients, $\tilde{g}_k$. If one observes a drop in magnitude for the coefficients of the higher order modes it is an indication that the lower order modes contained in the expansion captures the behaviour of the quantity in question. The magnitude of the expansion coefficients for the seventh order $P_{7,D}[Y]$, are shown in figure 8.38. For the following analysis one should disregard the value of the zeroth order mode as this is simply the mean of the expansion and does not determine its shape.

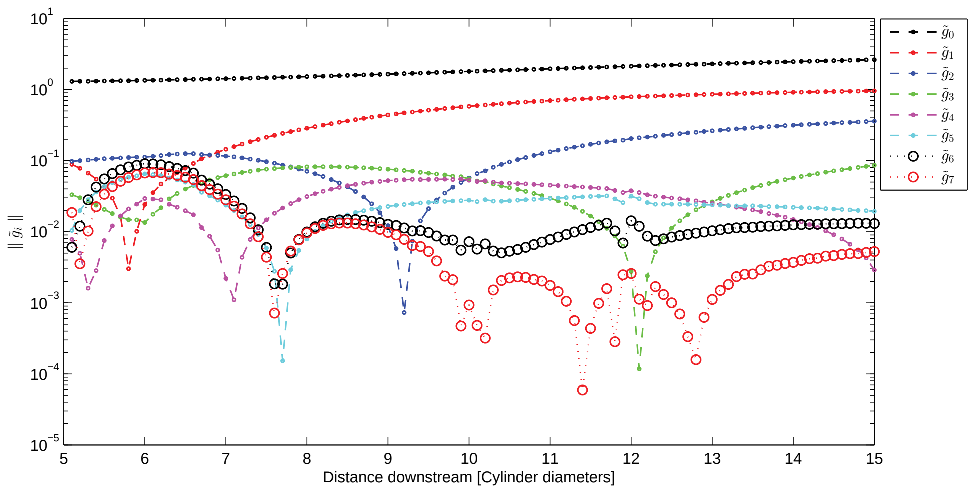


Figure 8.38: Expansion coefficients $\tilde{g}_k$ of the seventh order gPC expansions at each of the one hundred equidistant points downstream of the cylinder in the interval $x \in [5, 15]$.

From this figure it is seen that the coefficients in the range $x \in [5, 7]$ all lie in or close to the interval $\tilde{g}_k \in [10^{-1}, 10^{-2}]$, i.e no consistent drop in magnitude is observed. This means that all modes are almost equally present in the expansion which in turn indicates that the expansion has not converged yet. For $x \in [8, 15]$ however it is seen that the higher order modes, $N \in \{5, 6, 7\}$, are between one and two orders of magnitude lower then several of the lower order modes. This indicate that the behaviour of $y(Re)|_{x=\text{fixed}}$ is captured very well by the lower order modes in the expansion. For $x \in [11, 15]$ it is seen that the $\tilde{g}_k$'s consistently drop for all modes drop with increasing k indicating that here the behaviour is captured fully by the gPC expansion. However one would still need to perform a test using an even higher order gPC expansion to check if the behaviour of the expansion coefficients observed here indeed continue for higher order modes.

For the part of the domain where the gPC expansion seems to have converged, the data presented in figure 8.36f may be used to accurately determine the likelihood that a part of the domain downstream of the cylinder contains the center of the clockwise rotating vortex if it is known that $Re \in \mathcal{U}[150, 240]$ without knowledge of the value of Re.

In conclusion figure 8.38 suggests that the gPC expansion has not yet converged and a higher order basis is needed to be able to draw completely trustworthy conclusions about the pathways of the vortices.

Vorticity Magnitude: The statistics for the magnitude of the vorticity along the trace have been calculated. Here it was found that the statistics had indeed converged at the fifth order expansion, leading to the conclusion that the UQ approximation captures the statistics accurately. The mean along with mean plus and minus one and two standard deviations in $x \in [5, 15]$ are presented in figure 8.39.

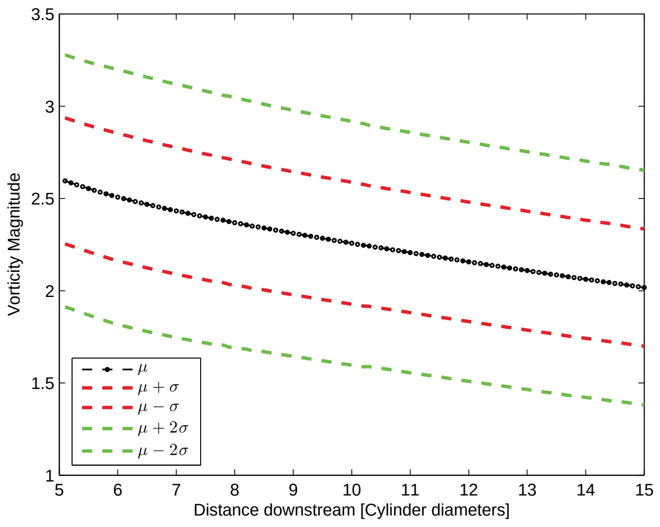
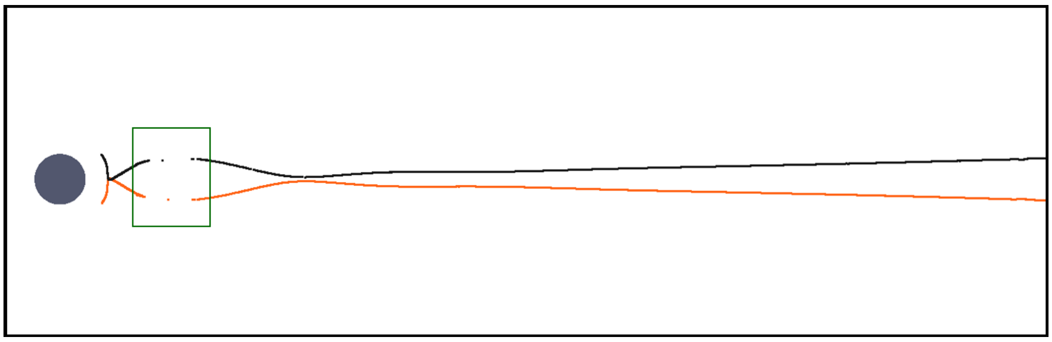


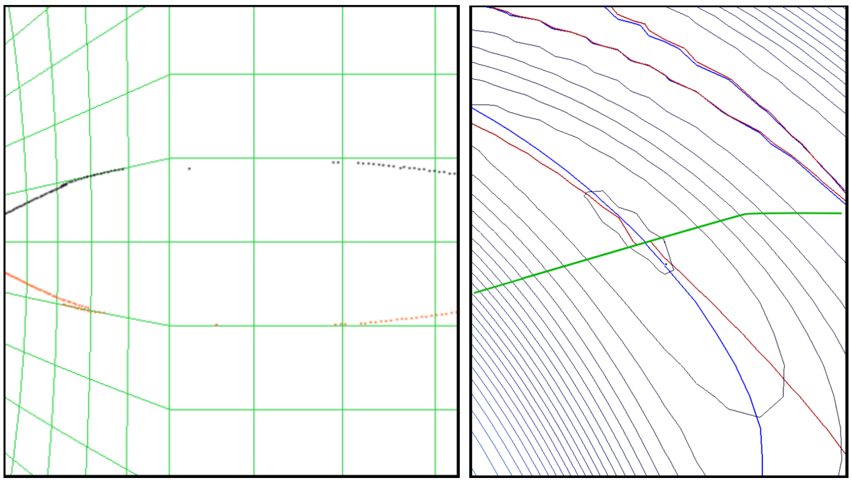
Figure 8.39: Mean vorticity magnitude downstream of the cylinder for a uniformly distributed random variable, Reynolds number over the interval $Re \in [150, 240]$.

From the figure it may be noted that the linear decrease in vorticity observed for all Re investigated earlier is also seen for the mean, as one would expect.

8.5 Method Limitation

During the application of the vorticity extrema trace method a limitation due to the numerical solution of the model problem was identified. The galerkin based SEM method used for solving the model problem only guarantees C^0-continuity[31] of the solution across the elements which make up the domain. The vorticity extrema trace method relies on the second derivative of the solution for calculating the $\omega_x = 0$ and $\omega_y = 0$ contours. Thus these contours are not guaranteed to be continuous across elements. The possibility for discontinuities means that if an extrema in vorticity travels very close to an interface between two elements over a significant distance there is a risk that any discontinuity in ω_x and or ω_y will hide the extrema.


(a) Clockwise and counter clockwise rotating vortex paths.


(b) Zoom of vortex paths with underlying finite element mesh added for clarity.

(c) Zoom around a vorticity extrema in the area where the method fails to capture it.

Figure 8.40: $Re = 300$, Cylinder in free flow. Illustration of the vorticity extrema trace method failing to capture part of vortex paths and the reason behind the failure. (c) Vorticity contours are black, element edge is green, $\omega_x = 0$ contours are blue and $\omega_y = 0$ contours are red.

For the cylinder in free flow at $Re = 300$ using an "unlucky" choice of mesh the above

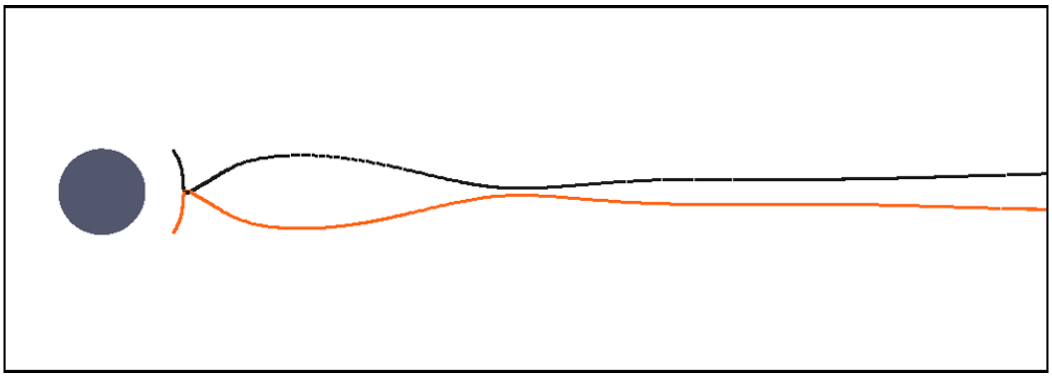
[31] C^0 continuity is continuity in a function over its domain but not necessarily in any of its derivatives.

described behaviour was observed. The problem is illustrated in figure 8.40.

Figure 8.41a shows the full pathways of the clockwise and counter clockwise rotating vortices surviving downstream. From here it is clear to see that part of the path is missing. Figure 8.41b shows a zoom of the area marked by a green square in figure 8.40a with the underlying finite element mesh added for clarity. From here it is easy to see that the vorticity extrema follow element edges in the mesh for the part not picked up by the method. A further zoom on the clockwise rotating vortex, including vorticity contours marking the extrema, at a time step where the extrema is not captured by the algorithm is presented in figure 8.40c.

Here the blue lines mark the $\omega_x = 0$ contours and the red lines the $\omega_y = 0$ contours. Focusing on the element edges marked by the green line it may be seen that $\omega_x = 0$ and $\omega_y = 0$ does not cross but instead jump across each other. This is due to a discontinuity in $\omega_y = 0$ at the element edge. The jump leads to the algorithm not catching the vorticity extrema.

Even though this is clearly a flaw in the present implementation of the method, it should be noted that the failure to capture the trace was only encountered for this specific simulation. The simulation at $Re = 300$ for the cylinder in free flow was redone using a different mesh where the vortex extrema did not follow any element edges. The mesh had $N = 3000$ elements and the SEM simulation used tenth order basis functions on each element. Here both vortex traces where captured perfectly as shown in figure 8.41.


(a) Clockwise and counter clockwise rotating vortex paths on new mesh.

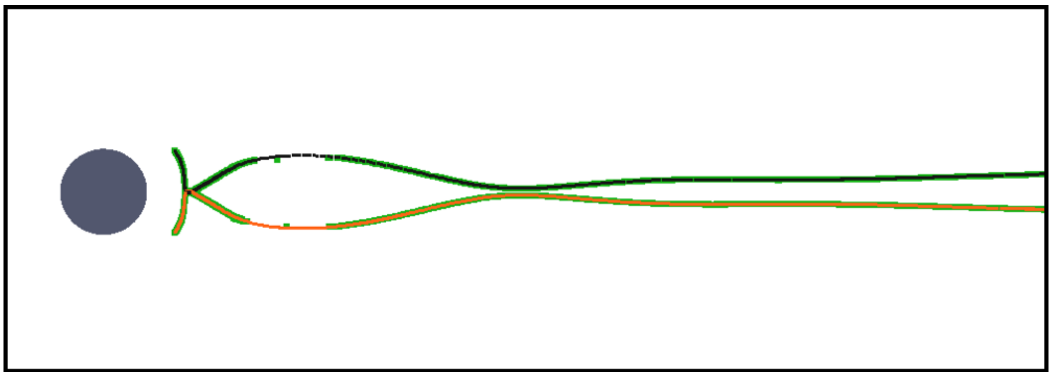

(b) Overlay of the clockwise and counter clockwise rotating vortex paths of the new and old meshes. The vortex paths from the old mesh are displayed as a broad green trace.

Figure 8.41: $Re = 300$, Cylinder in free flow. $N = 3000, P = 10$.

From figure 8.41b it is seen that the vortices follow the exact same path which indicate that the SEM method captured there physics correctly for the lower resolution. This means that it was the method for capturing the critical points in vorticity that caused the problem.

When the method is used to identify bifurcations in the flow the problem is not critical. This is because a sudden disappearance of an extrema away from a domain boundary is not possible. Any extrema away from a boundary can only disappear as a consequence of two or more extrema merging in a bifurcation. This means that the artificial disappearance of an extrema caused by the flaw described above may be identified and disregarded during analysis.

If the method is used to trace vortex paths with the intent on calculating quantities along them the problem is somewhat bigger. Here a technique for mending the problem in a given simulation could be to rerun the simulation using a different mesh. Alternatively the number of elements and basis order used could be increased further to minimize any discontinuity across the elements, however this method is very costly. Yet another technique could be to modify the contour algorithm used to attempt to capture discontinuities and correct them.

9 Conclusion and Future Work

This final chapter provides a short summery conclusion on the work done in this project as well as some ideas for future work.

In Conclusion

Two fluid dynamic model problems have been investigated. The first consisted of a cylinder in an open flow and was investigated in the Reynolds number regime $Re \in [100, 600]$. The second consisted of introducing a moving wall near the cylinder and was investigated for $Re \in [20, 300]$. In both cases a SEM based solver was used to obtain velocity and pressure fields used for post processing and analysis. The pros and cons of the SEM have been discussed and the main steps in applying the method to a simple model problem have been highlighted. The importance of using higher order meshes to mesh curved geometries when using the SEM have been illustrated. A series of convergence and validation tests comparing to previously published results, [8] [7] [10], have been presented to support the validity of the obtained results.

A definition of a vortex as an extrema in vorticity have been given which allow for unique identification of all vortices in a flow. By the use of existing theory from the field of DST the stationary points of the vorticity field at a given point in time have been shown to correspond to critical points for a dynamical system. This allowed the analysis of structural changes in the flow as a function of time, Re and/or D/G using existing theory developed for a dynamical system governed by the stream function, [2].

A method for identifying the stationary points in vorticity and determining their type, based on a method by Brøns et al [12], have been presented. The implementation of the method using **Paraview**'s python module have been described and all code supplied in appendices. The method have been used to trace vortex and vorticity saddle-point movement patterns at a variety of $(Re, D/G)$-values. This have allowed the investigation of vortex movement patterns and magnitude decrease downstream from the cylinder and how these depend on Re and D/G. Regarding the magnitude it was found that the decrease far downstream of the cylinder was close to linear for all Re and D/G values. Regarding the movement patterns it was found that close to the wall increasing Re forced the vortices away from the wall at a steeper and steeper angle. Decreasing Re moved the creation point of the vortices further downstream.

Using the tracing method an analysis of structural changes in the vortex patterns with varying time, Re and D/G was also performed. It has been found that almost all vortices observed in the flow are created and/or annihilated in simple codimension 1 saddle-center bifurcations away from solid boundaries. Only a single codimension 2 bifurcation was observed in the flow. Although interesting because it was different this bifurcation had no visible effect on the large scale structure of the flow. It was found that only saddle-points are created on the moving wall. On the cylinder surface downstream of the flow both centres (vortices) and saddle-points are created. Vortices created on the backside of the cylinder never travel downstream however but are instead annihilated in a saddle-center bifurcation close to the cylinder.

The Strouhal number at different D/G and Re values have been recorded and the stabilizing effect of the wall mapped. Here it was found that moving the cylinder close to the wall

$D/G = 10$ the critical value for the transition from stationary flow to periodic shedding more then doubled from $Re_{\text{crit}} \approx 46 \pm 1$ for the cylinder in free flow to $Re_{\text{crit}} \approx 100$.

A small scale investigation of the application of the UQ based SCM was performed using both the cylinder in free flow and near the moving wall. Here it was found by comparison with previous results that the method produces accurate functional dependencies and statistics for desired quantities depending on an underlying uncertain quantity with very few model problem realisations. It was also shown however that in order for the method to produce reliable results at low order it must be possible to measure desired quantities very accurately. If random noise was introduced the SCM based approximations varied significantly and this variation carried over to the calculation of the variance.

Finally a limitation of the method for tracing the stationary points in vorticity was discussed and different remedies suggested.

Further Work

A number of additional points of interest which could be addressed moving forward are listed below.

- **Optimizing performance of the SEM solver:** It could be beneficial to investigate, the trade off between the computations needed to perform single time step, as the number of elements is increased to maintain accuracy as the polynomial order of the basis functions is decreased in order to increase the minimum stable time step size. An equilibrium point between a time step size which allows much fewer steps to be taken to reach the desired solution and the time it takes to solve the problem at each individual time step is bound to exist. If this equilibrium is found simulations can be performed with "optimal" speed thus freeing resources to perform more tests or increasing the possible problem complexity. In short, an investigation of the choice of the optimal choice for dt, h and p would be interesting and beneficial.
- **Pushing the limit of the SEM with regards to Re:** An investigation of how high values of Re can be handled using the SEM for solving the NS-equation would be interesting.
- **Widening the parameter domain for the bifurcation investigations:** In the investigations performed in the current parameter domain it was seen that the effect of the wall was not gone at the smallest D/G-ratio investigated. Even though it is deemed unlikely it could be possible that new discoveries may be made moving the wall further from the cylinder. Furthermore the cylinder could still be moved even closer to the wall to see if any new structures in the vortex patterns emerged. At the same time the Reynolds number could be increased beyond $Re = 300$, however this would move the simulations further from the breakdown of two dimensionality. The increase of Re could in turn prompt the introduction of fully three dimensional simulations of the model problems which would require a more extensive theory in order to analyse any new emerging structures.
- **Considering more complex model problems:** Another possibility could be to apply the method to more complex two dimensional model problems. One example which was considered briefly in the present work is rotating the cylinder while it is near the wall. Due to time limitations this model problem was not investigated more

then by a few initial simulations but these investigations suggested that new vortex structures might be observed as the rotation velocity is varied.

- **Considering the size of a vortex:** In the investigation of a vorticis magnitude as it travelled downstream only the magnitude at its center was studied. Another interesting point could be to investigate the size and size change of the vortex as it moves through the flow. Here the size could be defined by the area around the vortex center within which the magnitude is larger then a fraction β of the magnitude at the vortex center.

- **Performing more thorough investigations of the SCM used on the model problem:** Higher order gPC expansions could be used to see if this reduces the sensitivity to random perturbations in the quantities of interest. Also multivariate problems where both Re and D/G are allowed to contain uncertainty could be investigated, e.g. in order to compare the sensitivity of the solution on either parameter.

10 References

[1] James D. Meiss. *Differential Dynamical Systems.* SIAM, 2007.

[2] Morten Brøns. Streamline topology: Patterns in fluid flows and their bifurcations. *ADVANCES IN APPLIED MECHANICS*, 41, 2007.

[3] DONGBIN XIU and JAN S. HESTHAVEN. High-order collocation methods for differential equations with random inputs. *SIAM J. SCI. COMPUT*, 27(3):1118–1139, 2005.

[4] Dongbin Xiu. *Numerical Methods for Stochastic Computations - A Spectral Method Approach.* Princeton University Press, 2010.

[5] http://www.nektar.info/.

[6] http://www.paraview.org/Wiki/ParaView/.

[7] R.D. Henderson. Details of the drag curve near the onset of vortex shedding. *Physics of Fluids*, 7:2102–2104, 1995.

[8] Wei-Xi Huang and Hyung Jin Sung. Vortex shedding from a circular cylinder near a moving wall. *Journal of Fluids and Structures*, 23:1064–1076, 2007.

[9] Morten Brøns, Bo Jakobsen, Kristine Niss, Anders V. Bisgaard, and Larsk K. Voigt. Streamline topology in the near wake of a circular cylinder at moderate reynolds numbers. *Journal of Fluid Mechanics*, 584:23–43, 2007.

[10] R.D. Henderson. Nonlinear dynamics and pattern formation in turbulent wake transition. *Journal of Fluid Mechanics*, 352:65–112, 1997.

[11] Frank M. White. *Viscous Fluid Flow.* McGRAW-HILL BOOK COMPANY, 1974.

[12] Morten Brøns and Anders V. Bisgaard. Bifurcation of vortex breakdown patterns in a circular cylinder with two rotating covers . *Journal of Fluids Mechanics*, 568:329–349, 2006.

[13] Randall J. Leveque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge University Press, 2002.

[14] A. P. Engsig-Karup. *The Spectral/hp-Finite Element Method for Partial Differential Equations.* Scientific Computation Section, DTU Informatics, Technical University of Denmark, Bldg. 321, DK-2800 Kgs.-Lyngby, Denmark, 2012.

[15] David A. Kopriva. *Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers (Scientific Computation).* Springer, 2009.

[16] Spencer Sherwin George Em Karniadakis. *Spectral/hp Element Methods for Computational Fluid Dynamics.* Oxford Science Publications, 2005.

[17] Anders Logg, Kent-Andre Mardal, and Garth N. Wells. *Automated Solution of Differential Equations by the Finite Element Method.* Springer, 2012.

[18] George Em Karniadakis, Moshe Israeli, and Steven A. Orszag. High-order splitting methods for the incompressible navier-stokes equations. *Journal of Compuational Physics*, 97:414–443, 1991.

[19] Randall J. LeVeque. *Finite Difference Methods for Ordinary Partial Differential Equations - Steady-State and Time-Dependent Problems.* Springer, 2012.

[20] Christophe Geuzaine and Jean-Francois Remacle. ***Gmsh Reference Manual***. http://www.geuz.org/gmsh, 1.12 edition, August 2003.

[21] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3ed Edition: The Art of Scientific Computing.* Cambridge University, 2007.

[22] http://www.cc.dtu.dk/.

[23] *MPI: A Message-Passing Interface Standard, Version 3.0.* High Performance Computing Center Stuttgart, 2012.

[24] L. I. G. Kovasznay Communicated by Sir Geofferey Taylor. Laminar flow behind a two-dimensional grid. 1947.

A Appendix

A.1 Sample mesh used for simulations

This appendix consists of the following.

- A sample **Gmsh .geo** file used to generate a mesh for use with **Nektar++** for incompressible Navier-Stokes simulations.
- Instructions on how to generate the appropriate **.msh** file assuming **Gmsh** Version 2.7.0 is used.

.Geo file for Hybrid Mesh for Cylinder Near Wall: Below is a rather lengthy block of text containing the **.geo** file used to generate a mesh for a cylinder near a wall with the parameter $D/G = 5/4$. To use it, copy-paste the text to an empty file and store it as **AnyName.geo**.

```
//////////////////////////////////////////////////////////////////////////////
/* This file contains the geometry for meshing a cylinder near a wall
(a domain boundary) in 2D for use in incompressible Navier-Stokes simulations
of steady and turbulent flow (Re ¬ 20-300)  */

/* IMPORTANT FOR USE WITH Nektar++: Nektar++ has a counter clockwise edge
definition which must be respected when creating surfaces. Due to this all
denominations are started in the lower right corner of any region and
increases counterclockwise around that region. This starts with vertices,
then edges, then surfaces and lastly volumes. */
//////////////////////////////////////////////////////////////////////////////

/*///// Domain paramters and coordinates used for meshing /////*/
// Distance from wall to cylinder
G = 4.0/5.0;

// Cylinder parameters
r_inner = 0.5; // radius of physical cylinder
r_outer = 0.5 + 0.2 * G; // Artifical outer cylinder for meshing purposes
D = 2.0 * r_inner; // Cylinder Diamter
x_c_c = 0.0; y_c_c = r_inner + G; // Cylinder centrum: x_c(enter)_c(ylinder)

// Relationship between cylinder diameter and distance to wall
DoverG = D / G;

// Domain dimensions
x_u_l = 30.0 * D; x_l_l = -10.0 * D; // x-axis limits, x_u(pper)_l(imit)
y_u_l = 20.0 * D; y_l_l = 0.0 * D; // y-axis limits, y_u(pper)_l(imit)

// x-coordinate for vertical area division lines: x_v(ertical)_h(elp)_X
x_v_h_1 = x_c_c-1.5;  x_v_h_2 = x_c_c;
x_v_h_3 = x_c_c+2.0; x_v_h_4 = 24.0;
// y-coordinate for horizontal area division line: y_h(orizontal)_h(elp)_X
y_h_h_1 = y_c_c; y_h_h_2 = y_c_c + 2.0;
y_h_h_3 = 6.0;

// Unstructured mesh devider: x_u(n)s(tructured)
x_us = x_v_h_4+1; y_us = y_h_h_3+2;
/*///// Constants i.e. coordinates for meshing /////*/

/*///// Points for creating underlying geometry /////*/
// Points on the background domain //
Point(1) = {x_l_l,y_l_l,0}; Point(2) = {x_v_h_1,y_l_l,0};
Point(3) = {x_v_h_2,y_l_l,0}; Point(4) = {x_v_h_3,y_l_l,0};
Point(5) = {x_v_h_4,y_l_l,0}; Point(6) = {x_l_l,y_h_h_1,0};
Point(7) = {x_v_h_1,y_h_h_1,0}; Point(8) = {x_v_h_2,y_h_h_1,0};
Point(9) = {x_v_h_3,y_h_h_1,0}; Point(10) = {x_v_h_4,y_h_h_1,0};
Point(11) = {x_l_l,y_h_h_2,0}; Point(12) = {x_v_h_1,y_h_h_2,0};
Point(13) = {x_v_h_2,y_h_h_2,0}; Point(14) = {x_v_h_3,y_h_h_2,0};
Point(15) = {x_v_h_4,y_h_h_2,0}; Point(16) = {x_l_l,y_h_h_3,0};
Point(17) = {x_v_h_1,y_h_h_3,0}; Point(18) = {x_v_h_2,y_h_h_3,0};
Point(19) = {x_v_h_3,y_h_h_3,0}; Point(20) = {x_v_h_4,y_h_h_3,0};
Point(21) = {x_l_l,y_u_l,0}; Point(22) = {x_v_h_1,y_u_l,0};
Point(23) = {x_v_h_2,y_u_l,0}; Point(24) = {x_v_h_3,y_u_l,0};
Point(25) = {x_v_h_4,y_u_l,0};
// Points on the background domain //

// Points for construction of the cylinders //
// Inner Cylinder
Point(26) = {x_c_c+r_inner,y_c_c,0}; Point(27) = {x_c_c,y_c_c+r_inner,0};
Point(28) = {x_c_c-r_inner,y_c_c,0}; Point(29) = {x_c_c,y_c_c-r_inner,0};
// Outer Cylinder
Point(30) = {x_c_c+r_outer,y_c_c,0}; Point(31) = {x_c_c,y_c_c+r_outer,0};
Point(32) = {x_c_c-r_outer,y_c_c,0}; Point(33) = {x_c_c,y_c_c-r_outer,0};
// Points for construction of the cylinders //

// Additional Background points //
// X-upper limit points
Point(34) = {x_u_l,y_l_l,0.0}; Point(35) = {x_u_l,y_h_h_1,0.0};
Point(36) = {x_u_l,y_h_h_2,0.0}; Point(37) = {x_u_l,y_h_h_3,0.0};
Point(38) = {x_u_l,y_u_l,0.0};

// Points for creating unstructured zone
Point(40) = {x_us,y_l_l,0.0}; Point(41) = {x_us,y_h_h_1,0.0};
Point(42) = {x_us,y_h_h_2,0.0}; Point(43) = {x_us,y_h_h_3,0.0};
Point(44) = {x_us,y_us,0.0}; Point(45) = {x_v_h_4,y_us,0.0};
Point(46) = {x_v_h_3,y_us,0.0}; Point(47) = {x_v_h_2,y_us,0.0};
Point(48) = {x_v_h_1,y_us,0.0}; Point(49) = {x_l_l,y_us,0.0};
Point(50) = {x_us,y_u_l,0.0}; Point(51) = {x_u_l,y_us,0.0};
/*///// Points for creating underlying geometry /////*/

/*///// Lines for creating underlying geometry /////*/
Line(1) = {1, 2}; Line(2) = {6, 7}; Line(3) = {11, 12}; Line(4) = {16, 17};
Line(5) = {2, 3}; Line(6) = {7, 32}; Line(7) = {29, 33}; Line(8) = {12, 13};
Line(9) = {17, 18}; Line(10) = {3, 4}; Line(11) = {26, 30}; Line(12) = {30, 9};
Line(13) = {13, 14}; Line(14) = {18, 19}; Line(15) = {4, 5}; Line(16) = {9, 10};
Line(17) = {14, 15}; Line(18) = {19, 20}; Line(19) = {1, 6}; Line(20) = {6, 11};
Line(21) = {11, 16}; Line(22) = {2, 7}; Line(23) = {7, 12}; Line(24) = {12, 17};
Line(25) = {3, 33}; Line(26) = {28, 32}; Line(27) = {27, 31};
Line(28) = {31, 13}; Line(29) = {13, 18}; Line(30) = {4, 9}; Line(31) = {9, 14};
Line(32) = {14, 19}; Line(33) = {5, 10}; Line(34) = {10, 15};
Line(35) = {15, 20};
Circle(36) = {26, 8, 27}; Circle(37) = {27, 8, 28}; Circle(38) = {28, 8, 29};
Circle(39) = {29, 8, 26}; Circle(40) = {30, 8, 31}; Circle(41) = {31, 8, 32};
Circle(42) = {32, 8, 33}; Circle(43) = {33, 8, 30};
Line(81) = {16, 49}; Line(82) = {49, 21}; Line(83) = {17, 48};
Line(85) = {18, 47}; Line(87) = {19, 46}; Line(89) = {20, 45};
Line(91) = {40, 41}; Line(92) = {41, 42}; Line(93) = {42, 43};
Line(94) = {43, 44}; Line(95) = {44, 50}; Line(96) = {34, 35};
Line(97) = {35, 36}; Line(98) = {36, 37}; Line(99) = {37, 51};
Line(100) = {51, 38}; Line(101) = {49, 48}; Line(102) = {48, 47};
Line(103) = {47, 46}; Line(104) = {46, 45}; Line(105) = {21, 22};
Line(106) = {22, 23}; Line(107) = {23, 24}; Line(108) = {24, 25};
Line(109) = {5, 40}; Line(110) = {40, 34}; Line(111) = {10, 41};
Line(113) = {15, 42}; Line(115) = {20, 43}; Line(117) = {45, 44};
Line(118) = {44, 51}; Line(119) = {25, 50}; Line(120) = {50, 38};
/*///// Lines for creating underlying geometry /////*/

/*///// Surfaces on which to create the mesh /////*/
// Structured Mesh Close to Cylinder
Line Loop(44) = {1, 22, -2, -19}; Plane Surface(45) = {44};
Line Loop(46) = {2, 23, -3, -20}; Plane Surface(47) = {46};
Line Loop(48) = {3, 24, -4, -21}; Plane Surface(49) = {48};
Line Loop(50) = {5, 25, -42, -6, -22}; Plane Surface(51) = {50};
Line Loop(52) = {6, -41, 28, -8, -23}; Plane Surface(53) = {52};
Line Loop(54) = {8, 29, -9, -24}; Plane Surface(55) = {54};
Line Loop(56) = {10, 30, -12, -43, -25}; Plane Surface(57) = {56};
Line Loop(58) = {12, 31, -13, -28, -40}; Plane Surface(59) = {58};
Line Loop(60) = {13, 32, -14, -29}; Plane Surface(61) = {60};
Line Loop(62) = {15, 33, -16, -30}; Plane Surface(63) = {62};
Line Loop(64) = {16, 34, -17, -31}; Plane Surface(65) = {64};
Line Loop(66) = {17, 35, -18, -32}; Plane Surface(67) = {66};
Line Loop(68) = {43, -11, -39, 7}; Ruled Surface(69) = {68};
Line Loop(70) = {11, 40, -27, -36}; Ruled Surface(71) = {70};
Line Loop(72) = {-37, 27, 41, -26}; Ruled Surface(73) = {72};
Line Loop(74) = {26, -38, -7, 42}; Ruled Surface(75) = {74};

// Unstructured zone
Line Loop(121) = {4, 83, -101, -81}; Plane Surface(122) = {121};
Line Loop(123) = {9, 85, -102, -83}; Plane Surface(124) = {123};
Line Loop(125) = {14, 87, -103, -85}; Plane Surface(126) = {125};
Line Loop(127) = {18, 89, -104, -87}; Plane Surface(128) = {127};
Line Loop(129) = {115, 94, -117, -89}; Plane Surface(130) = {129};
Line Loop(131) = {113, 93, -115, -35}; Plane Surface(132) = {131};
Line Loop(133) = {111, 92, -113, -34}; Plane Surface(134) = {133};
Line Loop(135) = {109, 91, -111, -33}; Plane Surface(136) = {135};

// Structured far-field
Line Loop(137) = {101,102,103,104,117,95,-119,-108,-107,-106,-105,-82};
Plane Surface(138) = {137};
Line Loop(139) = {118, 100, -120, -95};
Plane Surface(140) = {139};
Line Loop(141) = {110, 96, 97, 98, 99, -118, -94, -93, -92, -91};
Plane Surface(142) = {141};

// Marking surfaces for structured mesh generation
// Structured surfaces in zone of interest
Transfinite Surface {51} = {7,3,33,32};
Transfinite Surface {53} = {7,32,31,13};
Transfinite Surface {57} = {3,9,30,33};
Transfinite Surface {59} = {30,9,13,31};
Transfinite Surface {45,47,49,55,61,63,65,67,69,71,73,75};
Recombine Surface{45,47,49,51,53,55,57,59,61,63,65,67,69,71,73,75};
// Structured surfaces in far field zone
```

```
Transfinite Surface {138} = {49,44,50,21};
Transfinite Surface {140} = {44,51,38,50};
Transfinite Surface {142} = {40,34,51,44};
Recombine Surface {138,140,142};
/*///// Surfaces on which to create the mesh /////*/

/*///// Meshing /////*/
/// Vertical-Lines-Close-To-Cylinder ///
  // Leftmost Elements
  Transfinite Line {1, 2, 3, 4} = 8 Using Progression 0.75;
  // Rightmosts Elements
  Transfinite Line {15, 16, 17, 18} = 45 Using Progression 1.02;
  // Middle Left Away from cylinder
  Transfinite Line {8, 9} = 6 Using Progression 1;
  // Middle Right Away from cylinder
  Transfinite Line {13, 14} = 7 Using Progression 1;
/// Vertical-Lines-Close-To-Cylinder ///

/// Horzontal-Lines-Close-To-Cylinder ///
  // Top lines
  Transfinite Line {29,32,35,24,21} = 8 Using Progression 1.15;
  // Middle lines away from cylinder
  Transfinite Line {20,23} = 4 Using Progression 1; // Before Cylinder
  Transfinite Line {31,34} = 7 Using Progression 1; // After Cylinder
  // Buttom lines away from cylinder
  Transfinite Line {19,22} = 3 Using Progression 1; // Before Cylinder
  Transfinite Line {30,33} = 5 Using Progression 1; // After Cylinder
/// Horzontal-Lines-Close-To-Cylinder ///

/// Around the cylinder ///
  // Buttom line left of cylinder
  Transfinite Line {5} = 8 Using Progression 0.8;
  // Buttom line right of cylinder
  Transfinite Line {10} = 8 Using Progression 1;
  // Lines radial from outer cylinder to rest of domain
  Transfinite Line {12,28,-6} = 10 Using Progression 1.15;
  Transfinite Line {25} = 10 Using Progression 1;
  // Lines radial from inner to outer cylinder
  Transfinite Line {11,27,7,26} = 3 Using Progression 1;
  // Outer and inner cylinder lines
  Transfinite Line {39,43} = 12 Using Progression 1; // Lower Right
  Transfinite Line {38,42} = 10 Using Progression 1; // Lower Left
  Transfinite Line {36,40} = 13 Using Progression 1; // Upper Right
  Transfinite Line {37,41} = 9 Using Progression 1; // Upper Left
/// Around the cylinder ///

/// Domain away from zone of interest
  // Upper-Outer-Zone Vertical lines
  Transfinite Line {82, 95, 100} = 8 Using Progression 1.2;
  // Right-Outer-Zone Horizontal lines
  Transfinite Line {110, 118, 120} = 8 Using Progression 1.2;
  // Lower-Outer-Zone Vertical lines
  Transfinite Line {91, 96} = 2 Using Progression 1;
  Transfinite Line {92, 97} = 3 Using Progression 1;
  Transfinite Line {93, 98} = 5 Using Progression 1;
  Transfinite Line {94, 99} = 2 Using Progression 1;
  // Left-Outer-Zone Horizontal lines
  Transfinite Line {117, 119} = 2 Using Progression 1;
  Transfinite Line {104, 108} = 12 Using Progression 1;
  Transfinite Line {103, 107} = 2 Using Progression 1;
  Transfinite Line {102, 106} = 2 Using Progression 1;
  Transfinite Line {101, 105} = 7 Using Progression 1;
/// Domain away from area of interest

/// Unstructured zone
  Transfinite Line {109, 111, 113, 115} = 2 Using Progression 1; // Left-zone horizontal lines
  Transfinite Line {81, 83, 85, 87, 89} = 2 Using Progression 1; // Middle zone vertical lines
/// Unstructured zone
/*///// Meshing /////*/

/*//// Physical Domain and boundaries ////*/
Physical Line(143) = {21, 20, 19}; // In-flow
Physical Line(144) = {81}; Physical Line(145) = {82}; // In-flow
Physical Line(146) = {1, 5, 10, 15, 110, 105, 106, 107, 108, 119, 120}; // Far-field
Physical Line(147) = {109}; // Far-field
Physical Line(148) = {100, 99, 98, 97, 96}; // Out-flow
Physical Line(149) = {36, 39, 38, 37}; // Wall
Physical Surface(150) = {45, 47, 49, 55, 53, 51, 61, 59, 57,
                         67, 65, 63, 73, 71, 69, 75}; // Domain
Physical Surface(151) = {122, 124, 126, 128, 130, 132, 134, 136}; // Domain
Physical Surface(152) = {142, 140, 138}; // Domain
/*//// Physical Domain and boundaries ////*/
```

If copy-pasting the above text (or any code presented in these appendices) to your favourite text editor fails the reader is welcome to write an email to the author and ask for an electronic copy of the file. Current email: RasmusElCh@gmail.com.

Generating the Mesh (.msh file): To generate a **.msh** file using tenth order basis functions for element edges from the **.geo** file presented above follow the steps below[32].

- Open the **Gmsh** GUI (Version 2.7.0 or higher).
- Click the tab: **File -> Open**.
- In the open window, navigate to folder containing the **.geo** file.
- Click the file name then **OK**. The geometry should now be showing in the GUI.
- Click the drop down menu: **Mesh -> 2D**. A mesh should now have appeared on top of the geometry.
- Click the drop down menu: **Mesh -> Optimize high order**. A window should now have opened.
- In the window, type **10** in the field denoted **Polynomial order** the click **Apply**. Close the window.
- Click the drop down menu: **Mesh -> Save**.

You should now have created and saved the mesh with the same file name as your **.geo** file but with the extension **.msh** instead.

A.2 Nektar++ XML Extract

This appendix contains an extract from a **Nektar++ .xml** file used for simulating the cylinder near the moving wall along with a short explanation. The extract contains all but the geometry of the problem.

```
<NEKTAR>
    <GEOMETRY DIM="2" SPACE="2">
        <!-- Everything in the Geometry is supplied from the .msh file during the conversion by the
              MeshConvert program. -->
        <VERTEX>
      <!-- Vertex ids and their (x,y,z) coordinates go here -->
        </VERTEX>
        <EDGE>
      <!-- Edge ids and the ids of the vertexes constituting the edge go here (Counter clockwise
           ordering is used) -->
        </EDGE>
        <ELEMENT>
      <!-- Element ids and the ids of the edges constituting the element go here (Counter clockwise
            ordering is used) -->
        </ELEMENT>
        <CURVED>
      <!-- Edge ids and information about the type of curve used to form a curved edge goes here. -
           ->
        </CURVED>
        <COMPOSITE>
      <!-- Each edge constituting a domain boundary is put into different composites for later
           application of boundary conditions, Example below: -->
            <C ID="143"> E[524,527,560,563,566,605,608,611,614,617,620,622] </C>

      <!-- Each element constituting part of the domain is put into composites to define the
           problem domain, Examples below: -->
            <C ID="150"> Q[311-1659] </C>
            <C ID="151"> T[0-310] </C>
            <C ID="152"> Q[1660-1904] </C>
        </COMPOSITE>

        <DOMAIN>
```

[32]This guide is based on the Linux 64-bit release of **Gmsh** and the steps may differ on the Windows or Mac OS releases.

```
        <!-- The composites which constitute the domain -->
            C[150,151,152]
        </DOMAIN>
    </GEOMETRY>

    <EXPANSIONS>
        <!-- The COMPOSITEs constituting the domain, the number of modes, NUMMODES, used in the
             solution basis, the TYPE of basis (modal or nodal) and the FIELDS for which a solution
              is desired. -->
        <E COMPOSITE="C[150-152]" NUMMODES="10" TYPE="MODIFIED" FIELDS="u,v,p" />
    </EXPANSIONS>

    <CONDITIONS>
        <!-- Solver and Problem information -->

        <SOLVERINFO>
            <!-- Choice of the type of solver -->
            <I PROPERTY="SolverType" VALUE="VelocityCorrectionScheme" />
            <!-- The equation to solve -->
            <I PROPERTY="EQTYPE" VALUE="UnsteadyNavierStokes" />
            <!-- How to handle convection -->
            <I PROPERTY="EvolutionOperator" VALUE="Nonlinear" />
            <!-- The type of solver: CG or DG -->
            <I PROPERTY="Projection" VALUE="Continuous" />
            <!-- The time stepping scheme -->
            <I PROPERTY="TimeIntegrationMethod" VALUE="IMEXOrder2" />
        </SOLVERINFO>

        <PARAMETERS>
            <P> TimeStep = 0.0005  </P>  <!-- Time step length -->
            <P> NumSteps = 20000  </P>  <!-- Number of time steps -->
            <P> IO_CheckSteps = 80  </P>  <!-- How often to print the solution -->
            <P> IO_InfoSteps = 1000  </P>  <!-- How often to print an i am alive message -->
            <P> Re = 240  </P>  <!-- The Reynolds number -->
            <P> Kinvis = 1/Re  </P>  <!-- Kinematic Viscosity -->
        </PARAMETERS>

        <VARIABLES>
            <!-- giving ids to the fields -->
            <V ID="0"> u </V>
            <V ID="1"> v </V>
            <V ID="2"> p </V>
        </VARIABLES>

        <BOUNDARYREGIONS>
            <!-- Setting different boundary regions -->
            <B ID="0"> C[143-147] </B> <!-- Inflow -->
            <B ID="1"> C[148] </B> <!-- Outflow -->
            <B ID="2"> C[149] </B> <!-- Wall -->
        </BOUNDARYREGIONS>

        <BOUNDARYCONDITIONS>
            <!-- Setting boundary conditions for each region -->
            <REGION REF="0">
                <D VAR="u" VALUE="1" />
                <D VAR="v" VALUE="0" />
                <N VAR="p" USERDEFINEDTYPE="H" VALUE="0" />
            </REGION>
            <REGION REF="1">
                <N VAR="u" VALUE="0" />
                <N VAR="v" VALUE="0" />
                <D VAR="p" VALUE="0" />
            </REGION>
            <REGION REF="2">
                <D VAR="u" VALUE="0" />
                <D VAR="v" VALUE="0" />
                <N VAR="p" USERDEFINEDTYPE="H" VALUE="0" />
            </REGION>
        </BOUNDARYCONDITIONS>

       <FUNCTION NAME="InitialConditions">
       <!-- Setting initial conditions -->
            <E VAR="u" VALUE="1" />
            <E VAR="v" VALUE="0" />
            <E VAR="p" VALUE="0" />
       </FUNCTION>
    </CONDITIONS>
</NEKTAR>
```

A.3 The Nektar++ framework

This appendix is dedicated to giving the reader an introduction to the SEM framework **Nektar++** [5] that may be used as a tool for solving CFD problems with the implemented incompressible Navier-Stokes solver.

The approach adapted in this section is one which is designed to provide the reader with an overview of **Nektar++** from a users perspective. It covers the installation and an overview of the different utilities provided with **Nektar++**. This should provide the reader with enough help to install and use Nektar++.

A.3.1 Installing Nektar++

The reader is referred to **http://www.nektar.info/wiki/3.3/Compile/** for a set of installation instructions for either Mac, Windows or Linux. The author used the Linux instructions and found that these worked without needing to modify anything. The installation and following simulations were based on the following **Nektar++** and third party software versions[33],

- **Nektar version:** nektar++-3.3.0.
- **ThirdParty version:** ThirdParty-3.3.0.

A.3.2 Navigating Nektar++

In the following, the notation **(Nektar Dir)/** is a reference to the folder which contains the downloaded **Nektar++** source code. The default name for this folder is: nektar++-3.3.0.

Once **Nektar++** has been built on the users system the only two folders which the end user should be concerned with is **(Nektar Dir)/builds/** and **(Nektar Dir)/regressionTests/**.

- **(Nektar Dir)/builds/**: This folder contains seven folders, only three of which are of initial interest to the user. The folders of interest are:
  - **(Nektar Dir)/builds/regressionTests/**: This folder contains a file **runtests.sh** which is a shell script that runs a series of tests confirming the all the solvers function as intended. This test should be run every time an update for **Nektar++** is installed.
  - **(Nektar Dir)/builds/solvers/**: This folder contains the solvers currently implemented in **Nektar++**. In order to use any of the solvers all that is needed is a correctly formatted **.xml** file containing your problem (see section A.3.3 on setting up a problem and **.xml** file) in a directory of choice and then from this directory run the solver.
  - **(Nektar Dir)/builds/utilities/**: In this folder there are two new folders of interest, these are:
    - **PostProcessing/**
    - **PreProcessing/**
- **(Nektar Dir)/builds/utilities/PreProcessing/**: In Preprocessing there are three folders each containing a preprocessing tool. Only the mesh converting tool **MeshConvert** in the ./MeshConvert/ folder is of immediate interest. This tool may be used to convert a mesh created in one of the supported formats to the **Nektar++ .xml** format. A full description of how to use the mesh converter tool is available at **http://www.nektar.info/wiki/3.3/Tutorial/MeshConvert**.

[33]All downloadable from `http://www.nektar.info/downloader`.

- **(Nektar Dir)/builds/utilities/PostProcessing/**: This folder contains a subfolder ./Extras/ and a series of post processing tools. The folder also contain a series of post processing tools. All tools are in the form of an executable. The tools of immediate interest to the end user are listed in the following and may roughly be divided into three categories:
  - **Category 1**: Calculations of Quantities from present fields.
    - Gradient:
      - **Name**: CalcGrad
      - **Location**: (Nektar Dir)/builds/utilities/PostProcessing/Extras/
      - **Description**: Calculate the gradient of the field numbered *ifield* in the FILENAME.xml file for the solution stored in FILENAME.fld and prints a new .fld file FILENAME_with_grad.fld
      - **Call**: CalcGrad *ifield* FILENAME.xml FILENAME.fld
      - **Output file**: FILENAME_with_grad.fld
    - Vorticity:
      - **Name**: CalcVorticity
      - **Location**: (Nektar Dir)/builds/utilities/PostProcessing/Extras/
      - **Description**: Calculates the Vorticity of the 2D (u,v) or 3D (u,v,w) velocity fields stored in FILENAME.fld and prints it along with allready existing fields to a new .fld file FILENAME_with_vorticity.fld
      - **Call**: CalcVorticity FILENAME.xml FILENAME.fld
      - **Output file**: FILENAME_with_vorticity.fld
    - L2-norm:
      - **Name**: L2
      - **Location**: (Nektar Dir)/builds/utilities/PostProcessing/Extras/
      - **Description**: Calculate the L2-norm for the field numbered *ifield* in the FILENAME.xml and stored in FILENAME.fld
      - **Call**: L2 *ifield* FILENAME.xml FILENAME.fld
      - **Output text**: printf(L2 of field "ifield" is: XX.XXXX)
    - Linf-norm:
      - **Name**: Linf
      - **Location**: (Nektar Dir)/builds/utilities/PostProcessing/Extras/
      - **Description**: Calculate the Linf-norm for the field numbered *ifield* in the FILENAME.xml and stored in FILENAME.fld
      - **Call**: Linf ifield FILENAME.xml FILENAME.fld
      - **Output**: printf(Linf of field "ifield" is: XX.XXXX)
  - **Category 2**: Conversions for further simulations.
    - Convert a field from one mesh to another on the same domain:
      - **Name**: FieldToField
      - **Description**: Perform the conversion of the fields stored in FIELDSOLDMESH.fld from one mesh OLDMESH.xml to the file FIELDSNEWMESH.fld on another mesh NEWMESH.xml. In order to work after the conversions the domains

in the two .xml files must be the same, i.e. same boundaries and size, but the mesh is allowed to change. Currently only supports a modal basis.
- **Call**: FieldToField OLDMESH.xml FIELDSOLDMESH.fld NEWMESH.xml FIELDSNEWMESH.fld
- **Output file**: FIELDSNEWMESH.fld
- **Location**: (Nektar Dir)/builds/utilities/PostProcessing/Extras/

– **Category 3**: Conversions for visualisation / Post processing with other software.

  * Split the N fields in an **.fld** file to N seperate **.fld** files.
    - **Name**: SplitFld
    - **Call**: SplitFld FILENAME.fld
    - **Output files**: FILENAME_1.fld FILENAME_2.fld ... FILENAME_N.fld

  * Convert **.fld** file to a file which may be opened by **Gmsh**.
    - **Name**: "FldToGmsh"
    - **Call**: FldToGmsh FILENAME.fld
    - **Output file**: FILENAME.pos

  * Convert **.fld** file to a file which may be opened by **Tecplot**.
    - **Name**: FldToTecplot
    - **Call**: FldToTecplot FILENAME.fld
    - **Output file**: FILENAME.tec

  * Convert **.fld** file to a file which may be opened by **Paraview**:
    - **Name**: FldToVtk
    - **Call**: FldToVtk FILENAME.fld
    - **Output**: FILENAME.vtu

  * Convert **Nektar++** session **.xml** file to a file which may be opened by **Tecplot**:
    - **Name**: XmlToTecplot
    - **Call**: XmlToTecplot FILENAME.xml
    - **Output file**: FILENAME.tec

  * Convert Nektar++ session **.xml** file to a file which may be opened by **Paraview**:
    - **Name**: XmlToVtk
    - **Call**: XmlToVtk FILENAME.xml
    - **Output file**: FILENAME.vtu

• **(Nektar Dir)/regressionTests/**: The **regressionTests/** folder contains another folder **Solvers/** which in turn contain six folders named after six of the solvers implemented in **Nektar++**. Each of these folders contain a new folder named **InputFiles/**, in which there are a series of .xml files each containing a fully functional **Nektar++** simulation which the user can use as inspiration / a template for setting up their own problem.

File types: *The file types,* ***.fld, .rst*** *and* ***.chk*** *are the same to* ***Nektar++*** *programs (i.e. it contains the exact same information it is only the name of the file extension that differ). This is simply to allow the user to distiguish between intermediate files during a simulation and the final file.*

Steps of Solver Execution: (assuming paraview and Nektar++ are installed.)

Step 0: choosing the **.xml** file to use.
e.g.: Test_Advection3D_m12_DG_hex.xml. found in the folder
(Nektar Dir)/regressionTests/Solvers/ADRSolver/InputFiles/

Step 1: Place the **.xml** file to a folder created for the simulation.
e.g.: copy Test_Advection3D_m12_DG_hex.xml to /NektarSimulations/NewTest/.

Step 2: From this folder execute the appropriate solver.
e.g.: ~/(Nektar Dir)/builds/solvers/ADRSolver/ADRSolver Test_Advectionn3D_m12_DG_hex.xml

Step 3: from this folder convert output **.fld** file to desired format for visualisation.
e.g.: conversion to **.vtu** format for use in **Paraview** by executing:
~/(user nektar++ folder)/builds/utilities/PostProcessing/
FldToVtk Test_Advection1Dn3D_m12_DG_hex.xml Test_Advectionn3D_m12_DG_hex.fld

Step 4: from this folder open the **.vtu** file with paraview.
e.g.: paraview Test_Advection1Dn3D_m12_DG_hex.vtu

A.3.3 Setting Up a Problem

The reader is referred to
https://www.nektar.info/wiki/3.3/Tutorial?redirectedfrom=Latest/Tutorial for a series of tutorials for setting up problems for the individual **Nektar++** solvers and to
https://www.nektar.info/wiki/3.3/Reference/XmlFileFormat
for a general introduction to the **.xml** session files used to set up PDE problems to be solved using **Nektar++**.

A.4 Validation of Nektar++

Before using any piece of simulation software it is essential to validate the correctness of the solutions obtained by the software. If the validation step is skipped one at best risks loosing a lot of valuable time working with useless simulation data. At worst one risks using faulty data to draw conclusions which may be completely wrong with catastrophic results as a result.

This section documents a series of simulations performed **Nektar++**'s advection-diffusion-reaction (ADR) solver, to investigate the frameworks convergence properties, as well as the incompressible Navier-Stokes solver, **IncNavierStokesSolver** used in the project. In (A.4.1) the ADR-solver is used on a problem for which the analytical solution is known to illustrate the claimed convergences rate of the Spectral/hp-element method for both p- and h- refinement. In (A.4.2) the correctness of the incompressible Navier-Stokes solver, named **IncNavierStokesSolver** is validated using two model problems for which the analytical solution is known[34].

A.4.1 Simple Domain Convergence Test

In order to investigate and verify that the convergence rates for both h- and p- refinement agrees with the theoretical results a simple model problem with a known solution is formulated. The chosen problem is a 2D helmholtz problem on the domain illustrated in figure A.1a with Direchlet boundary conditions and the given forcing function f,

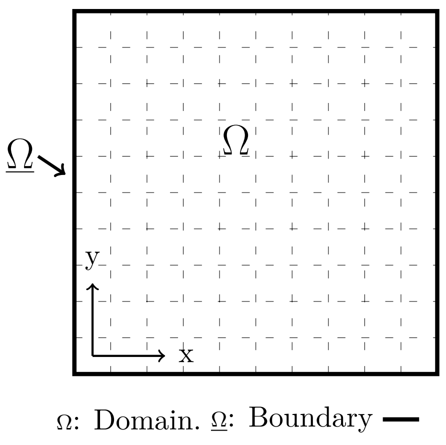
$$\begin{aligned} \nabla^2 u - \lambda u = f, \quad (x,y) \in \Omega = [-1,1]^2, \\ u(x,y) = 0 \quad (x,y) \in \underline{\Omega}, \\ f(x,y) = -(\lambda + 2\pi^2)\sin(\pi x)\sin(\pi y). \end{aligned} \tag{A.1}$$

This problem has the exact solution,

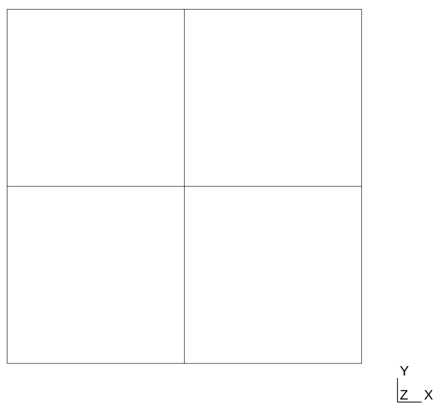
$$u_{\text{exact}} = \sin(\pi x)\sin(\pi y), \quad (x,y) \in [-1,1]^2,$$

as may be verified easily by insertion.

The solver in **Nektar++** which handles Helmholtz problems is called **ADRSolver** and is in general capable of solving a large class of Advection-Diffusion-Reaction problems (hence the name). A very simple "base mesh" consisting of four elements has been prepared using **Gmsh** and is shown in figure A.1b,



(a) Illustration of domain for simple Helmholtz problem.



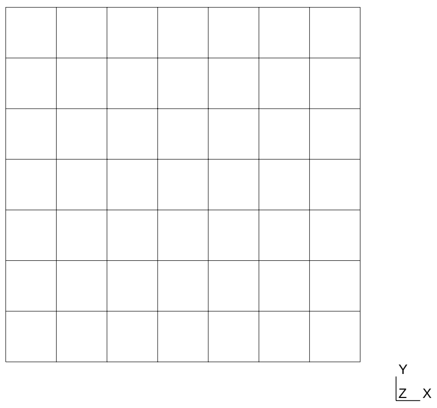
(b) The "base mesh" exportet from **Gmsh**.

Figure A.1

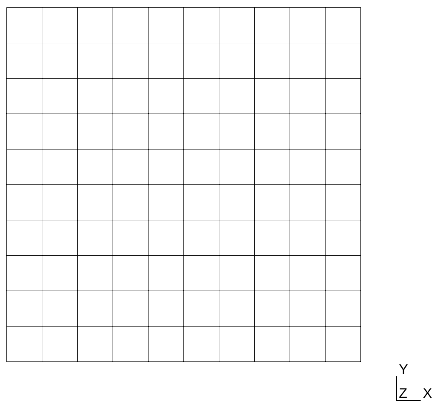
[34]This test is an execution of the tutorial found on http://www.nektar.info/wiki/3.3/Tutorial/IncNavierStokesSolver/VCS, with minor modifications

Based on this base mesh both an h- and p- convergence test has been performed. In both cases the error was measured in the L_2-norm.

In the h- convergence test the number of modes was set to, $N_{\text{modes}} = 2$, corresponding to two first order polynomials as basis functions, i.e. $p = 1$. This choice in the order of the expansion essentially reduces the SEM to the classical FEM. During the test the mesh granularity was gradually decreased by dividing the mesh into equally sized elements as shown in the figures A.1b through A.3b and hence decrease the width of each element.

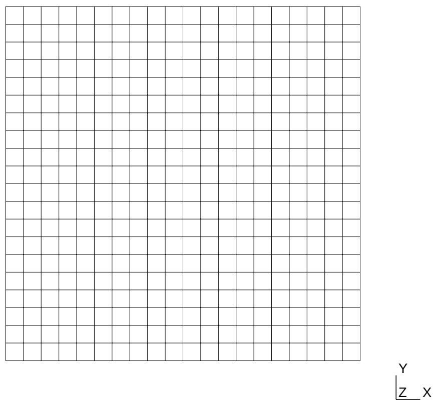


(a) Refined mesh with N=7 elements in each direction leading to a total of $N_{\text{total}} = 49$ elements.

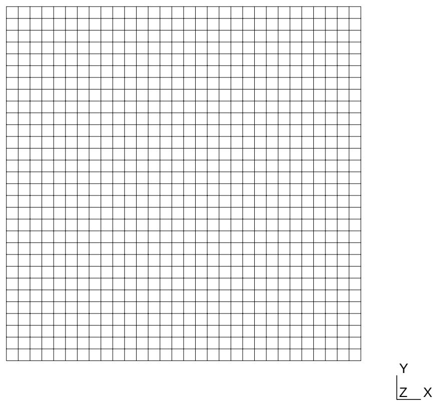


(b) Refined mesh with N=10 elements in each direction leading to a total of $N_{\text{total}} = 100$ elements.

Figure A.2



(a) Refined mesh with N=20 elements in each direction leading to a total of $N_{\text{total}} = 400$ elements.



(b) Refined mesh with N=30 elements in each direction leading to a total of $N_{\text{total}} = 900$ elements.

Figure A.3

The resulting error as a function of the elemental side length h, has been plotted in figure A.4.

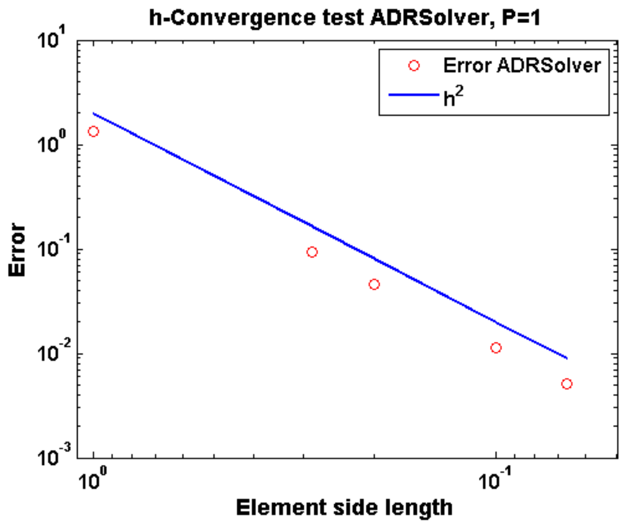


Figure A.4: h-convergence plot for the Nektar++ solver **ADRSolver**. The expected second order convergence is observed.

From theory it is expected that the standard FEM has second order convergence, as is exactly what we observe from the convergence plot.

Next the p- convergence test is performed. Here the number of elements in each direction is fixed to $N = 2$, i.e. the "base mesh" and the number of modes in the expansion is increased, i.e. the polynomial order p is increased. The convergence results for this test are presented in figure A.5.

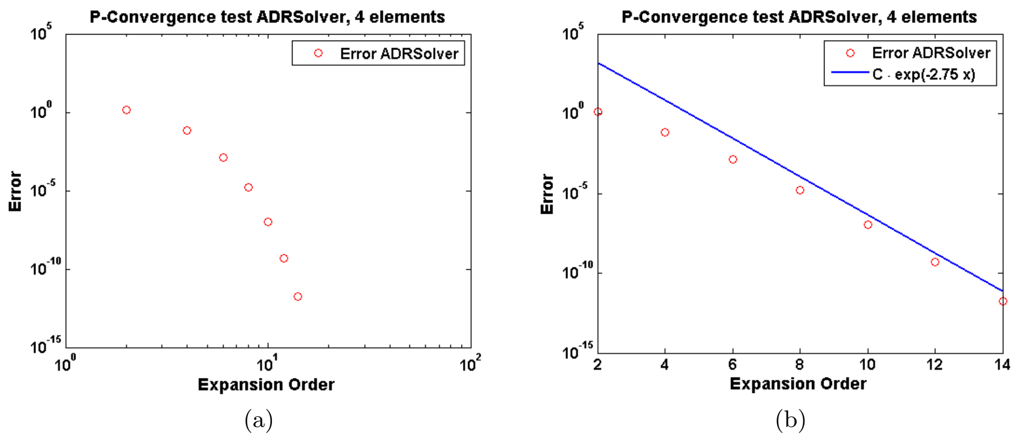


Figure A.5: P-convergence rate for the Nektar++ solver **ADRSolver**. The expected exponential convergence rate is observed.

From the SEM theory we expect that the convergence rate for p-refinement should be exponential as true solution is infinitely smooth. This is indeed exactly what we see in figure A.5.

As both the convergens rates investigated above behave as expected it is concluded that the **ADRSolver** seems to be implemented correctly.

A.4.2 Incompressible Navier-Stokes Solver

In order to confirm the correctness of the solver incompressible Navier-Stokes two test problems with known solutions are investigated. Both of these problems are part of the tutorial for the **IncNavierStokesSolver** found on the **Nektar++** wiki at **http://www.nektar.info/wiki/3.3/Tutorial/IncNavierStokesSolver/VCS**.

The first problem is that of a simple 2D Channel flow where a isothermal incompressible fluid is travelling down a channel as illustrated in figure A.6a where both the channel and the analytical solution are illustrated. The problem is described by the Navier Stokes and continuity equations and the Reynolds number (Re), boundary conditions and true solution are given by,

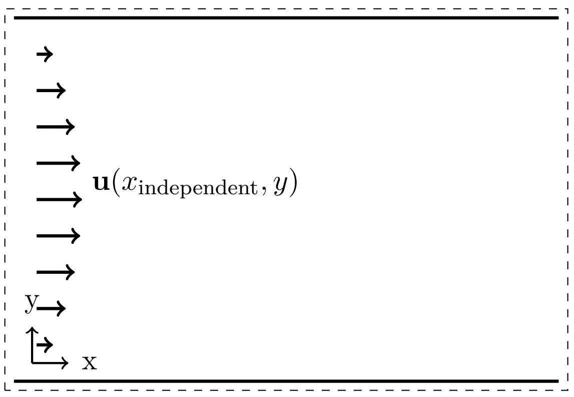
$$\begin{aligned}
\text{Reynolds number:}\quad & Re = 1 \\
\text{Wall BC's:}\quad & \mathbf{u} = \langle 0,0\rangle,\ \ \frac{d^n p}{d\xi^n} = 0 \\
\text{Inflow BC's:}\quad & \mathbf{u} = \langle y(1-y),0\rangle,\ \ \frac{d^n p}{d\xi^n} = 0 \\
\text{Outflow BC's:}\quad & \langle \frac{d}{dx}u, \frac{d}{dy}v\rangle = \langle 0,0\rangle,\ \ p = 0 \\
\text{Exact solution:}\quad & \mathbf{u} = \langle y(1-y),0\rangle,\quad p = -2(x-1),\quad (x,y)\in\Omega = [0,1]\times[0,1],
\end{aligned}$$

where the domain is $(x,y)\in[0,1]\times[0,1]$

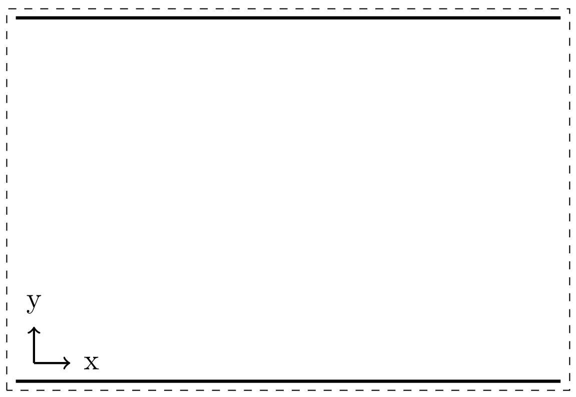
The second problem is the 2D-Kovasznay flow problem [24], in the same channel flow domain as for the earlier problem, see figure A.6b, but where the boundary conditions and Reynolds number are changed in order to produce the desired flow. The Reynolds number, boundary conditions and true solution are now given by,

$$\begin{aligned}
\text{Reynolds number:}\quad & Re = 40, \\
\text{Wall BC's:}\quad & \mathbf{u} = \langle 0,0\rangle,\ \ \frac{d^n p}{d\xi^n} = 0, \\
\text{Inflow BC's:}\quad & \mathbf{u} = \langle 1-1.619\cos(2\pi y), -0.2483\sin(2\pi y)\rangle,\ \ \frac{d^n p}{d\xi^n} = 0, \\
\text{Outflow BC's:}\quad & \langle \frac{d}{dx}u, \frac{d}{dy}v\rangle = \langle 1-0.3815\cos(2\pi y), -0.0585\sin(2\pi y)\rangle,\ \ p = 0.4272 \\
\text{Exact solution:}\quad & \mathbf{u} = \langle (1-\exp(-0.964x)\cos(2\pi y)), \frac{-0.964}{2\pi}\exp(-0.964)\sin(2\pi y)\rangle, \\
& p = 0.5(1-\exp(-2\cdot 0.964x)),
\end{aligned}$$

where the domain is, $(x,y)\in[0,1]\times[-0.5,1.5]$.



(a) A schematic for the domain used in a simple 2D-channel flow with the inflow at the left side and outflow at the right side of the channel and walls on the top and buttom. A vector field illustrating velocity profile for the true solution (only varying in y), $\mathbf{u}(x, y) = \langle y(1-y), 0\rangle$, is included in the schematic.



(b) A schematic for the domain used in a Kovasznay flow with the inflow at the left side and outflow at the right side of the channel and walls on the top and bottom.

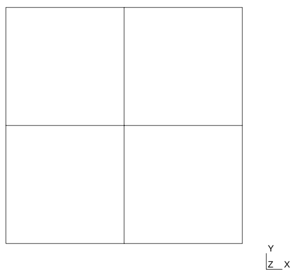
Figure A.6

It is seen that the exact solution for both problems are steady and thus the Incompressible Navier-Stokes solver should produce these steady solutions after a the initial transient period. In order to investigate whether this is indeed the case both problems have been simulated using the **IncNavierStokesSolver** using the continuous galerkin formulation and the **IMEXOrder1** time integration method.

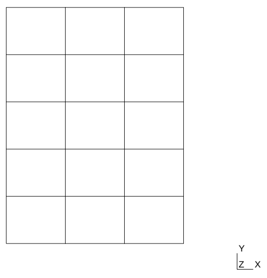
The channel flow problem has been simulated for 1000 time steps using a time step size of 0.001 starting from the initial conditions,

$$\langle u, v, p\rangle = \langle 0, 0, 0\rangle.$$

The computational domain is presented in figure A.7a from which it is seen that it only consists of four elements and the computations were performed using three modes, i.e. using second order polynomials. It is noted that as the true solution for the velocity field is a second order polynomial in y, and the true solution for the pressure field a first order polynomial in x it is expected that the SEM will recover the solutions exactly using the three modes.



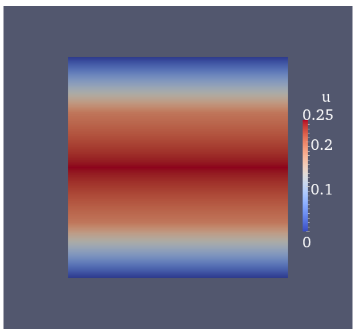
(a) The computational domain for the 2D-channel flow simulation.



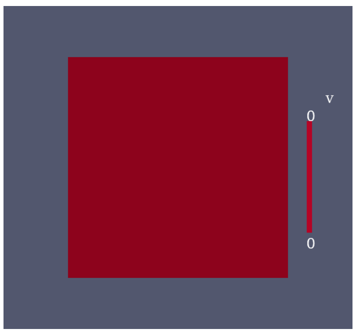
(b) The computational domain for the 2D-Kovasznay flow simulation.

Figure A.7

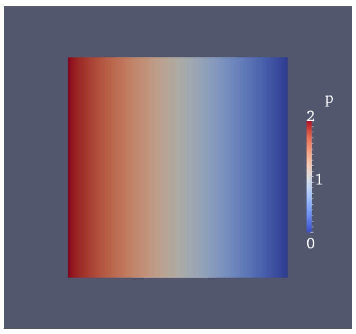
The simulation produced the results shown in figures A.8a through A.8c for u, v and p respectively.



(a) The x-component of the velocity field for the 2D-channel flow.



(b) The y-component of the velocity field for the 2D-channel flow.



(c) The pressure field for the 2D-channel flow.

Figure A.8

By inspecting the L_2-norm for the difference between the computed and exact solutions,

$$\begin{aligned}\parallel u_{\text{exact}} - u_{\text{computed}} \parallel_{L_2} &\approx 6.1062310^{-16},\\ \parallel v_{\text{exact}} - v_{\text{computed}} \parallel_{L_2} &\approx 1.4219110^{-16},\\ \parallel p_{\text{exact}} - p_{\text{computed}} \parallel_{L_2} &\approx 9.9920110^{-15},\end{aligned}$$

it is seen that this is indeed only different at the level of machine precision, and thus the solver has indeed provided the exact solution as predicted.

It was found that for the Kovasznay flow a longer time period is needed before the flow converges and thus 20000 time steps are taken at a time step size of 0.001. The initial conditions are set to be,

$$\langle u, v, p\rangle = \langle 1, 0, 0\rangle.$$

The computational domain for the Kovasznay simulation is presented in figure A.7b from which it is seen that it consists of fifteen elements. The Kovasznay flow is not mealy polynomial however and thus the true solution will not be captured exactly using a finite number of modes. For this reason a p-refinement test has been performed to see how the error in L_2-norm depends on the polynomial order for this flow. The result of the test is presented in figure A.9.

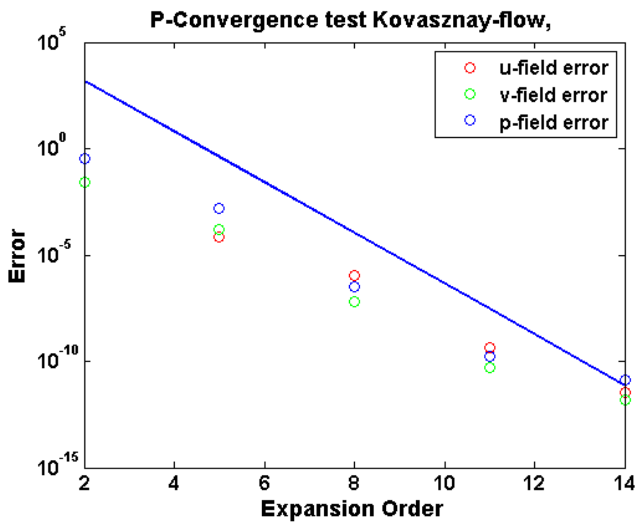
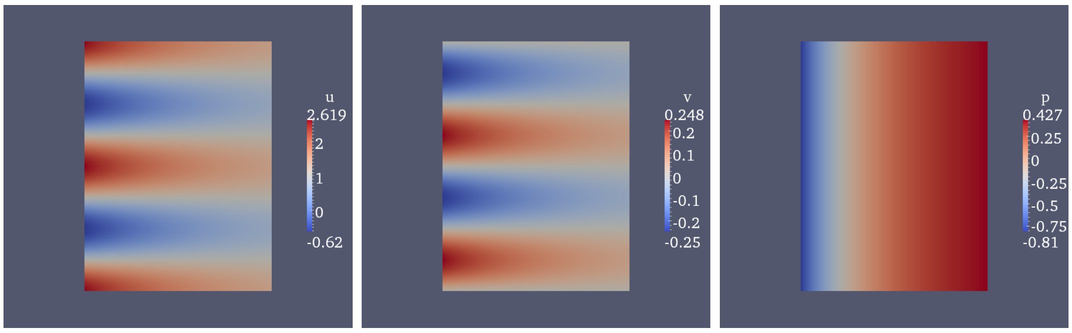


Figure A.9: P-convergence plot for the Nektar++ solver **IncNavierStokes-Solver** for the Kovasznay-flow. The expected exponential convergence rate is observed for all fields.

From figure A.9 it is seen that the error converges (more or less) exponentially as a function of polynomial order as expected until it hits machine precission.

In figures A.10(a), (b) and (c) the numerical solution for the Kovasznay flow using 15 modes is presented,



(a) The x-component of the velocity field for the 2D-Kovasznay flow.

(b) The y-component of the velocity field for the 2D-Kovasznay flow.

(c) The pressure field for the 2D-Kovasznay flow.

Figure A.10

In the tests of the Nektar++ incompressible Navier-Stokes solver above it was shown that the solution converges towards to true solution at the rate expected from a SEM solver. Based on these tests the author feel confident that the **IncNavierStokesSolver** works as expected and on this basis will proceed with simulations of a much different scope.

A.5 Authors Software

This appendix contains the scripts and source code written to perform the postprocessing on simulation data obtained with **IncNavierStokesSolver** from the **Nektar++** framework.

A.5.1 Shell Scripts

All Shell scripts written for the project. The shell scripts are all executed from a Linux Shell by navigating to the folder containing the script and executing the command: ./ScriptName.sh.

FullDataTreatment.sh : This script executes all other scripts for the postprocessing phase. The scripts must be placed as illustrated on figure 4.1 for the shell script to work.

```
# A shell script for executing the full post processing procedure on a
# CFD-simulation using Nektar++ for the Master Thesis:
# "Advanced CFD-Techiques for identification of structures in flows"

## Requirements for execution of the script:
#
# Shell scripts:
# NektarPP.sh
# DomainCropping.sh
# ZeroContour_CriticalPoints.sh
# VorticityExtremaType.sh
# CriticalPointTrace.sh
#
# C++ Programs:
# DomainCropper
```

```
#
# Python scripts:
# ContourAndCriticalPoints.py
# CriticalPointIdentification.py
# VTUFormatPointDataCombiner.py
#
# Nektar++ programs:
# CalcVorticity
# CalcGrad
# FldToVtk
#
# Execution: From Linux Shell in the folder where FullDataTreatment.sh is
    placed
# Command: ./FullDataThreatment.sh
##

## Nektar++ Post Processing
./ShellScripts/NektarPP.sh

## Domain Cropping
./ShellScripts/DomainCropping.sh

## Q_x = 0 Contour and Vorticity Critical Points
./ShellScripts/ZeroContour_CriticalPoints.sh

## Critical Point Type Identification
./ShellScripts/VorticityExtremaType.sh

## Tracing the different types of critical points
./ShellScripts/CriticalPointTrace.sh


## Arranging files in subfolders
mkdir ../Datafiles/FullDataCropped
mkdir ../Datafiles/ContourVorticityXZero
mkdir ../Datafiles/VorticityCriticalPoints
mkdir ../Datafiles/CriticalPointType
mkdir ../Datafiles/CriticalPointTrace

mv ../Datafiles/*_cropped.vtu ../Datafiles/FullDataCropped/
mv ../Datafiles/*_Filtered_Contour_QxZero_Vorticity.vtp ../Datafiles/
    ContourVorticityXZero/
mv ../Datafiles/*_Extrema_Vorticity.vtp ../Datafiles/VorticityCriticalPoints/
mv ../Datafiles/*_Maxima.vtu ../Datafiles/CriticalPointType/
mv ../Datafiles/*_Minima.vtu ../Datafiles/CriticalPointType/
mv ../Datafiles/*_Saddle.vtu ../Datafiles/CriticalPointType/
mv ../Datafiles/*_Trace.vtu ../Datafiles/CriticalPointTrace/
```

NektarPP.sh : This script executes all **Nektar++** based postprocessing. It is important to modify the script by providing the correct path to the **Nektar++** programs used.

```
## This shell script executes Nektar++ post processing programs which
## calculates the vorticity as well as its gradient and hessian matrix.
## Finally the resulting .fld file containing all fields is converted to a .
    vtu
```

```
## file for further post processing in Paraview.

##
## WARNING: Only one (the correct) .xml file is allowed to be present in the
    folder
## WARNING: Only the .chk files related to the .xml file is allowed to be
    present
## WARNING: Use this script only for 2D-NavierStokes solutions from Nektar++
##          containing the fields u,v,p in that order.
##

# Removing any stray .fld files in the raw data folder
rm ../Datafiles/*.fld

#### Nektar++ programs ####

# The location of your nektar++ vorticity converter
  ## WARNING: must be changed to fit your system ##
CONVERTER=/home/rasmus/Nektar++/builds/utilities/PostProcessing/Extras/
    CalcVorticity
  ## WARNING: must be changed to fit your system ##

# The location of your nektar++ gradient calculator
  ## WARNING: must be changed to fit your system ##
CONVERTER2=/home/rasmus/Nektar++/builds/utilities/PostProcessing/Extras/
    CalcGrad
  ## WARNING: must be changed to fit your system ##

# the location of the FldToVtk routine to convert all the resulting fld-fiels
# to .vtu format
  ## WARNING: must be changed to fit your system ##
CONVERTER3=/home/rasmus/Nektar++/builds/utilities/PostProcessing/FldToVtk
  ## WARNING: must be changed to fit your system ##

#### Nektar++ programs ####

#### Calculation of Vorticity ####

# The chk files
CHK=../Datafiles/*.chk
# The xml file needed to do the conversion
XML=../Datafiles/*.xml

# The conversion to new .fld files containing the vorticity
for f in $CHK
do
  $CONVERTER $XML $f
  rm $f
done

#### Calculation of Vorticity ####

#### Calculation of the gradient of the Vorticity ####

# The newly created .fld files now containing the vorticities
FLD=../Datafiles/*.fld
# The field which gradient is to be calculated
FIELDNR=3

# The conversion to new .fld files containing the gradient of the vorticity
```

```
for f in $FLD
do
  $CONVERTER2 $FIELDNR $XML $f
  rm $f
done

#### Calculation of the gradient of the Vorticity ####

#### Calculating the gradient of each component of the vorticity gradient ####

# The newly created .fld files now containing the vorticity and its gradient
    in
# the (x,y)-plane.
FLD2=../Datafiles/*.fld
# The field for which the gradient is to be calculated
FIELDNR2=4

# The conversion to new .fld files containing the gradient of u
for f in $FLD2
do
  $CONVERTER2 $FIELDNR2 $XML $f
  rm $f
done

# The newly created .fld files now containing the vorticity, its gradient in
# the (x,y)-plane and the gradient of the first component of the vorticity
    gradient
FLD3=../Datafiles/*.fld
FIELDNR3=5

# The conversion to new .fld files containing the gradient of v
for f in $FLD3
do
  $CONVERTER2 $FIELDNR3 $XML $f
  rm $f
done

#### Calculating the gradient of each component of the vorticity gradient ####

#### Converting the resulting .fld files to .vtu format ####
FLD6=../Datafiles/*.fld

$CONVERTER3 $XML $FLD6

# Cleaning up the .fld files
rm ../Datafiles/*.fld

#### Converting the resulting .fld files to .vtu format ####
```

DomainCropping.sh : This script executes the C++ program **DomainCropper**. The script should be modified with the dimensions desired for the cropping.

```
## Shellscript which executes the C++ program: DomainCropper for each of
## a set of .vtu files. DomainCropper crops the data in the .vtu file by
## removing elements with no coordinates inside the three intervals:
```

```
## [x_min, x_max], [y_min,y_max], [z_min,z_max]. It then outputs a new
## .vtu file in the same format without the cropped data.

## WARNING: Set the input parameters to CONVERTER as desired
X_min=-2
X_max=10
Y_min=0
Y_max=4
Z_min=-1
Z_max=1

# The .vtu files needing cropping
VTUCROP=../Datafiles/*.vtu
# The location of the C++ domain cropping program.
  ## WARNING: must be changed to fit your system ##
CONVERTER=./C++Programs/DomainCropper
  ## WARNING: must be changed to fit your system ##

# The cropping of each of the .vtu files
for f in $VTUCROP
do
  $CONVERTER $f $X_min $X_max $Y_min $Y_max $Z_min $Z_max
  rm $f
done
```

ZeroContour_CriticalPoints.sh : The script calls the python script **ContourAndCriticalPoints.py** for all **.vtu** files for a simulation.

```
#!/bin/sh
# A set of .vtu files containing the fields
# ['u', 'v', 'p', 'Qz', 'Qz_x', 'Qz_y', 'Qz_x_x', 'Qz_x_y', 'Qz_y_x', 'Qz_y_y
    ']
# in this order are run through the python script to extract the
# Qz_x = 0 contour and vorticity extrema (Qz_x, Qz_y) = (0,0)
# to two series of .vtp files

# The .vtu files needing extraction
VTUEXTRACT=../Datafiles/*_cropped.vtu
# The location of the Paraview based python script.
  ## WARNING: must be changed to fit your system ##
CONVERTER2=./PythonScripts/ContourAndCriticalPoints.py
  ## WARNING: must be changed to fit your system ##

for f in $VTUEXTRACT
do
  python $CONVERTER2 vtp $f
done
```

VorticityExtremaType.sh : The script calls the python script **CriticalPonitIdentification.py** for all **.vtu** files containing extrema data for a simulation.

```
#!/bin/sh
## Copy this shell script along with the executeable file
## "CriticalPointTypeIdentification.py" to the folder containing your .vtp
    files
## with vorticity extrema and run it to obtain three sets of files containing
## the Critical points which are saddles, maxima and minima respectively.

# The _Contour_Extrema_Vorticity.vtp files are run through the python script
    to
# extract the extremas of each of the three different types to one file each.

# The .vtu files needing extraction
VTPEXTRACT=../Datafiles/*_Extrema_Vorticity.vtp
# The location of the Paraview based python script to be executed
  ## WARNING: must be changed to fit your system ##
CONVERTER2=./PythonScripts/CriticalPointIdentification.py
  ## WARNING: must be changed to fit your system ##

for f in $VTPEXTRACT
do
  python $CONVERTER2 vtu $f
done
```

CriticalPointTrace.sh : The script calls the python script **VTUFormatPointData-Combiner.py** for all **.vtu** files containing data for minima, maxima and saddles.

```
#!/bin/sh
## WRITE DOCUMENTATION

# The _Contour_Extrema_Vorticity.vtp files are run through the python script
    to
# extract the extremas of each of the three different types to one file each.

# The .vtp files used for tracing
VTUMAXIMA=../Datafiles/*_Maxima.vtu
VTUMINIMA=../Datafiles/*_Minima.vtu
VTUSADDLE=../Datafiles/*_Saddle.vtu
# The location of the VTU file combiner
  ## WARNING: must be changed to fit your system ##
CONVERTER2=./PythonScripts/VTUFormatPointDataCombiner.py
  ## WARNING: must be changed to fit your system ##

python $CONVERTER2 $VTUMAXIMA

python $CONVERTER2 $VTUMINIMA

python $CONVERTER2 $VTUSADDLE
```

A.5.2 Python Scripts

This section provides all the Python scripts written for post processing. They have all been written to be executed from a Linux terminal like a bash script.

ContourAndCriticalPoints.py : This script calculates the $\omega_x = 0$ contour and the vorticity extrema.

```
# #!/usr/bin/env python

#
   ##########################################################################################

# Python Script for extracting the Q_x = 0 contour and vorticity extrema
# using using the paraview library (based on VTK), from a .vtu file.
# The extracted data is filtered to only include the set of points where the
# magnitude of the vorticity is larger then the tolerance, tol: abs(Q) > tol
# (set to be atleast 5 orders of magnitude lower then the max(abs(Q))).
# The determination of the extrema's is performed using a contour method where
# first the Q_x = 0 contour is interpolated and then along this contour the
# Q_y = 0 contour is interpolated.
#
# The script uses the argparse library to allow input arguments.
#
# Input Arguments:
# Format: The format of the output file, must currently be: vtp
# File name: The name of the file which is to be processed, its format must be
#            a paraview compatible .vtu
#
# To change the default tolerance (tol = 0.005) one must do so below.
#
# Execution: ./ContourAndCriticalPoints.py FileToBeProcessed.vtu
#
# The output is:
# 1: A .vtp file containing the Q_x = 0 contour.
# 2: A .vtp file containing the extrema coordinates, and the field
# values interpolated at the extrema.
#
#
   ##########################################################################################

#  Written by: Rasmus E. Christiansen
   #
#
   ##########################################################################################


# Package handling
import argparse # Allowing commandline arguments
import paraview # Paraview functionality
import paraview.simple # Paraview functionality
from paraview.simple import * # Paraview functionality

# !!!! Input argument parser !!!! #
parser = argparse.ArgumentParser(description='Process a .vtu file for a
    contour.')
# files to be processed
```

```
parser.add_argument('format', type=str, nargs=1,
                    help='the output format, options: vtp')
parser.add_argument('files', metavar='pfn', type=str, nargs='+',
                    help='the file that is to be treated')
args = parser.parse_args()
# !!!! Input argument parser !!!! #

# !!!! Reading the .vtu file !!!! #
# Reading the input file into the paraview data reader
InputData = XMLUnstructuredGridReader( FileName=args.files )
# Setting the data names
InputData.PointArrayStatus = ['u', 'v', 'p', 'Qz', 'Qz_x', 'Qz_y',
                   'Qz_x_x', 'Qz_x_y', 'Qz_y_x', 'Qz_y_y']
InputData.CellArrayStatus = []
# !!!! Reading the .vtu file !!!! #

# !!!! Setting tolerances for data refinement !!!! #
# Current tolerance for when the vorticity is to small to consider
tol = 0.0005;
# Current maximum for the vorticity
Max_Qz = 10000;
# !!!! Setting tolerances for data refinement !!!! #

# !!!! Performing the calculation and printing data !!!! #
# Creating the Contour object
Contour1 = Contour( PointMergeMethod="Uniform Binning" )
# Calculating the desired isosurface
Contour1.PointMergeMethod = "Uniform Binning"
Contour1.ContourBy = ['POINTS', 'Qz_x']
Contour1.Isosurfaces = [0.0]
UpdatePipeline() # Updating the pipeline to announce that Contour1 is there.

# Creating an object for extracting the data where abs(Q) > tol
ExtractSelection1 = ExtractSelection()
# Defining the conditions under which the data should be extracted
selection_source_1 = ThresholdSelectionSource( ContainingCells=0, InsideOut=0,
     Thresholds=[-Max_Qz, -tol, tol, Max_Qz], ArrayName='Qz', FieldType='POINT'
     )

# Extracting the selection and copying it to the output paraview structure
ExtractSelection1.Selection = selection_source_1
ExtractSelection1.PreserveTopology = 1
Threshold1 = Threshold()
Threshold1.Scalars = ['POINTS', 'vtkInsidedness']
Threshold1.ThresholdRange = [0.0, 1.0]
UpdatePipeline() # Updating the pipeline to announce that Threshold1 is there.

# Outputting the resulting data in .vtp format
if args.format[0] == 'vtp':
  for i in args.files:
    FILENAME = i
    FILENAME = FILENAME.replace('.vtu','_Filtered_Contour_QxZero_Vorticity.vtp
        ')
    OUTPUT = open(FILENAME, "w+")
    # Creating a writer and writing the desired data in .vtp format to the
        output file
    writer = CreateWriter(FILENAME, ExtractSelection1)
    writer.DataMode = 0
    writer.Writealltimestepsasfileseries = 1
    writer.UpdatePipeline()
    del writer
    OUTPUT.close()
else:
  print 'You did not specify a correct output format: vtp'


# Creating the Contour object
Contour2 = Contour( PointMergeMethod="Uniform Binning" )
# Calculating the desired isosurface
Contour2.PointMergeMethod = "Uniform Binning"
Contour2.ContourBy = ['POINTS', 'Qz_y']
Contour2.Isosurfaces = [0.0]
UpdatePipeline() # Updating the pipeline to announce that Contour2 is there.

# Outputting the resulting data in .vtp format
if args.format[0] == 'vtp':
  for i in args.files:
    FILENAME = i
    FILENAME = FILENAME.replace('.vtu','_Extrema_Vorticity.vtp')
    OUTPUT = open(FILENAME,"w+")
    # Creating a writer and writing the desired data in .vtp format to the
        output file
    writer = CreateWriter(FILENAME, Contour2)
    writer.DataMode = 0
    writer.Writealltimestepsasfileseries = 1
    writer.UpdatePipeline()
    del writer
    OUTPUT.close()
else:
  print 'You did not specify a correct output format: vtp'
```

CriticalPointIdentification.py : This script calculates the type of the critical points of the vorticity ω.

```
# #!/usr/bin/env python

###########################################################################
# Python script for identifying and extracting maxima, minima and saddlepoints
# for the critical points of the vorticity in 2D. This is done by considering
# the eigenvalues of the Hessian matrix and utilizing that
# in 2D these may be calculated analytically with relative ease from:
#
# Q = Vorticity
#
# Hessian = [Q_x_x , Q_x_y ; Q_y_x , Q_y_y]
#
# Eigenvalues = 1/2 * ( Q_x_x + Q_y_y +/-
#                 sqrt((Q_x_x + Q_y_y)^2 - 4*(Q_x_x * Q_y_y - Q_x_y * Q_y_x) )
#
# Input parameters:
# * format: The output format, currently only vtu
# * files: The file containing the critical points to be investiagated,
#          .vtp format required.
#
# Execution: ./CriticalPointIdentification.py FileToBeProcessed.vtp
```

```
#
# Output:
# * files: Three files each containing critical points of a specific type,
#          types are: Maximum, Minimum, Saddle.
#
#
################################################################################
#  Written by: Rasmus E. Christiansen                                          #
################################################################################


# Package handling
import argparse # Allowing commandline arguments
import paraview # Paraview functionality
import paraview.simple # Paraview functionality
from paraview.simple import * # Paraview functionality

# !!!! Input argument parser !!!! #
parser = argparse.ArgumentParser(description='Process a .vtp file containing
    critical points for vorticity to determine the critical points type.')
# files to be processed
parser.add_argument('format', type=str, nargs=1,
                    help='the output format, options: vtu')
parser.add_argument('files', metavar='pfn', type=str, nargs='+',
                    help='the file that is to be treated')
args = parser.parse_args()
# !!!! Input argument parser !!!! #

# !!!! Reading the .vtp file !!!! #
InputData = XMLPolyDataReader( FileName=args.files )
InputData = GetActiveSource()
InputData.PointArrayStatus = ['u', 'v', 'p', 'Qz', 'Qz_x', 'Qz_y',
                   'Qz_x_x', 'Qz_x_y', 'Qz_y_x', 'Qz_y_y']
InputData.CellArrayStatus = []
# !!!! Reading the .vtp file !!!! #


# !!!! Performing the calculation and printing data !!!! #

# Calculating the double the value of one of the Eigenvalues for the Hessian
# matrix for the vorticity at each critical point
Calculator1 = Calculator()
Calculator1.AttributeMode = 'point_data'
Calculator1.Function = 'Qz_x_x + Qz_y_y + sqrt((Qz_x_x + Qz_y_y)^2 - 4*(Qz_x_x
    * Qz_y_y - Qz_x_y * Qz_y_x))'
Calculator1.ResultArrayName = 'EigenValueTimes2'

# Calculating the vaule of the product of the two eigenvalues for the Hessian
# matrix for the vorticity at each critival point
Calculator2 = Calculator()
Calculator2.AttributeMode = 'point_data'
Calculator2.Function = 'Qz_x_x * Qz_y_y - Qz_x_y * Qz_y_x '
Calculator2.ResultArrayName = 'EigenValuesProduct'

# Extracting the critical points which are maxima or minima
ExtractSelection2 = ExtractSelection()
selection_source_4780 = ThresholdSelectionSource( ContainingCells=0, InsideOut
    =0, Thresholds=[1e-08, 1e10], ArrayName='EigenValuesProduct', FieldType='
    POINT' )
ExtractSelection2.Selection = selection_source_4780
```

```
# Extracting the critical points which are minima
ExtractSelection3 = ExtractSelection()
selection_source_4896 = ThresholdSelectionSource( ContainingCells=0, InsideOut
    =0, Thresholds=[1e-08, 1e10], ArrayName='EigenValueTimes2', FieldType='
    POINT' )
ExtractSelection3.Selection = selection_source_4896
UpdatePipeline()

# Outputting data for the critical points which are minima in .vtp format
if args.format[0] == 'vtu':
  for i in args.files:
    FILENAME = i
    FILENAME = FILENAME.replace('.vtp','_Minima.vtu')
    OUTPUT = open(FILENAME,"w+")
    # MAKE MORE WRITERS TO WRITE DIFFERENT FORMATS
    # Creating a writer and writing the desired data in .vtp format to the
        output file
    writer = CreateWriter(FILENAME, ExtractSelection3)
    writer.DataMode = 0
    writer.UpdatePipeline()
    del writer
    OUTPUT.close()
else:
  print 'You did not specify a correct output format: vtu'

# Extracting the critical points which are maxima
SetActiveSource(ExtractSelection2)
ExtractSelection4 = ExtractSelection()
selection_source_5012 = ThresholdSelectionSource( ContainingCells=0, InsideOut
    =0, Thresholds=[-1e10, -1e-08], ArrayName='EigenValueTimes2', FieldType='
    POINT' )
ExtractSelection4.Selection = selection_source_5012
UpdatePipeline()

if args.format[0] == 'vtu':
  for i in args.files:
    FILENAME = i
    FILENAME = FILENAME.replace('.vtp','_Maxima.vtu')
    OUTPUT = open(FILENAME,"w+")
    # MAKE MORE WRITERS TO WRITE DIFFERENT FORMATS
    # Creating a writer and writing the desired data in .vtp format to the
        output file
    writer = CreateWriter(FILENAME, ExtractSelection4)
    writer.DataMode = 0
    writer.UpdatePipeline()
    del writer
    OUTPUT.close()
else:
  print 'You did not specify a correct output format: vtu'



# Extracting the critical points which are saddles
SetActiveSource(Calculator2)
ExtractSelection6 = ExtractSelection()
selection_source_5244 = ThresholdSelectionSource( ContainingCells=0, InsideOut
    =0, Thresholds=[-1e10, -1e-08], ArrayName='EigenValuesProduct', FieldType='
    POINT' )
ExtractSelection6.Selection = selection_source_5244
```

```
UpdatePipeline()

if args.format[0] == 'vtu':
  for i in args.files:
    FILENAME = i
    FILENAME = FILENAME.replace('.vtp','_Saddle.vtu')
    OUTPUT = open(FILENAME,"w+")
    # MAKE MORE WRITERS TO WRITE DIFFERENT FORMATS
    # Creating a writer and writing the desired data in .vtp format to the
        output file
    writer = CreateWriter(FILENAME, ExtractSelection6)
    writer.DataMode = 0
    writer.UpdatePipeline()
    del writer
    OUTPUT.close()
else:
  print 'You did not specify a correct output format: vtu'

# !!!! Performing the calculation and printing data !!!! #
```

VTUFormatPointDataCombiner.py : This script takes a batch of **.vtu** files containing vorticity maxima, minima or saddle point information and combines the data stored there to a single file.

```
# #!/usr/bin/env python

###############################################################################
## Python script which takes a set of .vtu files (paraview format) and
    combines
## all field and coordinate information to a single .vtu file.
##
## The purpose of this program is to visualise the pathlines of the critical
## points for the vorticity: grad(w) = 0.
##
## In addition to the combination of the .vtu files
## an extra "field" is introduced which keeps track of what points and
## corresponding field-information belonged to what input file
##
## Input:
## * Files: A bacth of .vtu files (written *.vtu) to be combined to a single .
    vtu file
##
## Execution: ./VTUFormatPointDataCombiner.py *.vtu
##
## Output:
## * File: A single .vtu file containing all information from the input files
##
##############################################################################
#  Written by: Rasmus E. Christiansen                                        #
##############################################################################

## Package handling
import argparse # Allowing commandline arguments
import xml.etree.ElementTree as ET # Python XML-handling package
import copy # Used to create a copy of the XML format used in the input files
```

```
from array import array

# !!!! Input argument parser !!!! #
parser = argparse.ArgumentParser(description='Concatanates the information
    stored in a set of .vtu files.')
# files to be processed
parser.add_argument('files', metavar='pfn', type=str, nargs='+',
                   help='the files that are to be treated')
args = parser.parse_args()
# !!!! Input argument parser !!!! #

# !!!! Information needed to create output .vtp (XML) !!!! #
NumberOfFiles = len(args.files)
NumberOfPoints = 0;
OUTPUT_TEXT = [];
OUTPUT_NAME = [];
# !!!! Information needed to create output .vtp (XML) !!!! #

# !!!! Using the first input file to create XML structure for output !!!! #
OUTPUT_TREE = copy.deepcopy(ET.parse(args.files[0]))
OUTPUT_ROOT = OUTPUT_TREE.getroot()

for ITERATE in OUTPUT_ROOT.iter('DataArray'): # Clearing Data Arrays
  ITERATE.text = ''
  OUTPUT_TEXT.append([''])
  OUTPUT_NAME.append(ITERATE.get('Name'))
# !!!! Using the first input file to create XML structure for output !!!! #


# !!!! Iterating over the input files !!!! #
for File in args.files:
  # Parsing the current .vtp file to the ElementTree handler
  tree = ET.parse(File)
  root = tree.getroot()

  # Find the number of points in the present file
  for ITERATE_1 in root.iter('Piece'):
    NumberOfPoints = NumberOfPoints + int(ITERATE_1.get('NumberOfPoints'))

  # Iterating over all data
  for idx,ITERATE_2 in enumerate(root.iter('DataArray')):
    OUTPUT_TEXT[idx][0] = OUTPUT_TEXT[idx][0] + ITERATE_2.text.replace('\n
            ','').replace('  ',' ')

# !!!! Iterating over the input files !!!! #

# !!!! Creating connectivity and offset arrays !!!! #
CONNECT_OFFSET_TYPES = ['','','']
for f in range(NumberOfPoints):
  CONNECT_OFFSET_TYPES[0] = CONNECT_OFFSET_TYPES[0] + ' ' + str(f)
for f in range(1,NumberOfPoints+1):
  CONNECT_OFFSET_TYPES[1] = CONNECT_OFFSET_TYPES[1] + ' ' + str(f)
for f in range(1,NumberOfPoints+1):
  CONNECT_OFFSET_TYPES[2] = CONNECT_OFFSET_TYPES[2] + ' ' + '1'
# !!!! Creating connectivity and offset arrays !!!! #

# !!!! Filling the output .vtp (xml) !!!! #
# Copying the gathered data
for idx,ITERATE in enumerate(OUTPUT_ROOT.iter('DataArray')):
  INDEX_OF_DATA = OUTPUT_NAME.index(ITERATE.get('Name'))
  ITERATE.text = OUTPUT_TEXT[INDEX_OF_DATA][0]

# Correcting the connectivity and offset arrays
for ITERATE in OUTPUT_ROOT.iter('Cells'):
  for idx,ITERATE_2 in enumerate(ITERATE.iter('DataArray')):
    ITERATE_2.text = str(CONNECT_OFFSET_TYPES[idx])

# Setting the correct number of point data
for ITERATE in OUTPUT_ROOT.iter('Piece'):
    ITERATE.set('NumberOfPoints',str(NumberOfPoints))
    ITERATE.set('NumberOfCells',str(NumberOfPoints))
# !!!! Filling the output .vtp (xml) !!!! #


# !!!! Writing output .vtu (xml) file !!!! #
FILENAME = args.files[0]
FILENAME = FILENAME.replace('.vtu','_Trace.vtu')
OUTPUT_FILE = open(FILENAME, 'w+')
OUTPUT_TREE.write(FILENAME)
OUTPUT_FILE.close()
# !!!! Writing output .vtu (xml) file !!!! #
```

A.5.3 C++ Program

This section contains the source code for the **C++** program **DomainCropper**.

The program utilises the **C++** project **TinyXML**[35], and must be compiled with it. A sample Makefile to compile the program is provided below as well.

DomainCropper.cpp : The **C++** source code for the domain cropper.

```
//
   ////////////////////////////////////////////////////////////////////////////

//
//   File: DomainCropper.cpp
//
//   Description: Utility for cropping a simulation domain used in the spectral
//                element solver Nektar++ AFTER the data has been converted to
//                the .vtu (XML) format (used by paraview) to remove reduntant
//                parts of the domain in the post processing phase.
//
//         The data is returned in .vtu (XML) format with only the
//                redundant elements removed.
//
//         The tool removes all elements with no nodes within the domain
//                of interest provided as input as three intervals:
//                x \in [x_start,x_stop]
//                y \in [y_start,y_stop]
//                z \in [z_start,z_stop]
//
//
```

[35]The project may currently be downloaded from `http://sourceforge.net/projects/tinyxml/`

```
// Written by: Rasmus E. Christiansen
//
//
   ////////////////////////////////////////////////////////////////////////////


#include "tinyxml.h"
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <string>
#include <sstream>
#include <algorithm>
#include <iterator>
#include <vector>

using namespace std;

// Copy string to double array
void stofaNEW (string STR, int NumPoints, double* TempArray) {
    vector<string> TOK;
      istringstream iss(STR);
    copy(istream_iterator<string>(iss),
             istream_iterator<string>(),
             back_inserter<vector<string> >(TOK));

    for (int i = 0; i < 3*NumPoints; ++i)
    {
      TempArray[i] = atof(TOK.at(i).data());
    }
  }

// Identifies the maximum in an array of doubles
void max_finder (double* double_array, double* max_values, int dim, int
    NumPoints)
{
  for (int i = 0; i < dim; ++i)
  {
    max_values[i] = double_array[i];
    for (int j = 0; j < NumPoints; ++j)
    {
      if (double_array[j*dim + i] > max_values[i])
        max_values[i] = double_array[j*dim + i];
    }
  }
}

// Identifies the minimum in an array of doubles
void min_finder (double* double_array, double* min_values, int dim, int
    NumPoints)
{
  for (int i = 0; i < dim; ++i)
  {
    min_values[i] = double_array[i];
    for (int j = 0; j < NumPoints; ++j)
    {
      if (double_array[j*dim + i] < min_values[i])
        min_values[i] = double_array[j*dim + i];
    }
  }
```

```
}

int main(int argc, char *argv[])
{
  // Checking that the correct number of input arguments are passed
  if (argc != 8)
  {
    printf("You must pass the following arguments: \n");
    printf("input.vtu x_start x_stop y_start y_stop z_start z_stop \n");
    return 0;
  }

  // Reading in the desired cropped domain size
  double CroppedDomainSize[2][3];
  for (int i = 0; i < 3; ++i)
  {
    CroppedDomainSize[0][i] = atof(argv[2 + 2*i]);
    CroppedDomainSize[1][i] = atof(argv[3 + 2*i]);
  }

  // Reading the input XML-file
  TiXmlDocument DocIn(argv[1]);
  DocIn.LoadFile();

  // Creating a handle for the XML-file
  TiXmlHandle hDoc(&DocIn);
  TiXmlElement *pElem;

  // Storage for iteration over the elements
  int NumPoints; int dim = 3;
  string ElementData; int CopyPiece = 0;
  double* max_values = new double[3];
  double* min_values = new double[3];

  // Objects needed to output the final TiXml Document //
  TiXmlDocument DocOut;
  TiXmlNode* OutputPieces;

  // Writing the initial parts of the XML (.vtu) file
  TiXmlDeclaration* decl = new TiXmlDeclaration( "1.0", "", "" );
  DocOut.LinkEndChild( decl );

  TiXmlElement* InitialElement1 = new TiXmlElement( "VTKFile" );
  InitialElement1->SetAttribute("type","UnstructuredGrid");
  InitialElement1->SetAttribute("version","0.1");
  InitialElement1->SetAttribute("byte_order","LittleEndian");
  DocOut.LinkEndChild( InitialElement1 );

  TiXmlElement* InitialElement2 = new TiXmlElement( "UnstructuredGrid" );
  InitialElement1->LinkEndChild( InitialElement2 );

  // Looping over all elements (named Piece's in the XML) in the FEM-grid.
  // Placing the handle for the pieces just inside <UnstructuredGrid>
  // i.e. WE ARE NOW AT: pElem->Value() = Piece
  pElem = hDoc.FirstChildElement().FirstChild().FirstChild().ToElement();

  TiXmlNode* Cloned;

  //InitialElement1->LinkEndChild( Cloned );
  while ( pElem )
  {

    // Identifying the number of points of the first element
    NumPoints = atoi(pElem->Attribute( "NumberOfPoints" ));

    // Extracting the Element coordinate data
    ElementData = pElem->FirstChildElement()->FirstChildElement()->GetText();

    double* TempArray = new double[3*NumPoints];

    stofaNEW (ElementData, NumPoints, TempArray);
    max_finder(TempArray, max_values, dim, NumPoints);
    min_finder(TempArray, min_values, dim, NumPoints);

    for (int i = 0; i < dim; ++i)
    {
      if (max_values[i] <= CroppedDomainSize[0][i] ||
        min_values[i] >= CroppedDomainSize[1][i] ) {
      }
      else
      {
        CopyPiece += 1;
      }
    }
    if ( CopyPiece == dim ) {
      Cloned = pElem->Clone();
      InitialElement2->LinkEndChild( Cloned );
    }
    CopyPiece = 0;

    // Switching to the next element (Piece)
    pElem = pElem->NextSiblingElement();
    // clearing the temporary array
    delete[] TempArray;
  }


  // Saving the new XML to the output file
  string Temp = argv[1];
  string OutputName = Temp.substr(0, Temp.size());
  OutputName.append("_cropped.vtu");
  FILE* OutputFile;
  OutputFile = fopen (OutputName.c_str() , "w");

  DocOut.SaveFile( OutputFile );

  fclose(OutputFile);
}
```

Makefile : Sample Makefile for compiling the program.

```
# Variables
CC=g++

TINYXML_CFILES=tinyxml.cpp tinyxmlparser.cpp tinyxmlerror.cpp tinystr.cpp
```

```
# Targets for compilation
all:
  $(CC) $(TINYXML_CFILES) DomainCropper.cpp -o DomainCropper
clean:
  rm -rf *o DomainCropper
```

A.5.4 FEniCS Poisson Solver

A minimal python script for the **FEniCS** Poisson equation solver.

```
"""
Python script for a FEniCS based SEM solver:

Poisson equation is solved with Dirichlet conditions:
Equation: -Laplace(u) = f on the unit square.
BC: u = u0 on the boundary.
IC: u0 = u = sin(5*pi*x)*sin(5*pi*y),
Forcing: f = 50*pi*pi*sin(5*pi*x)*sin(5*pi*y).
"""

# Importing FEniCS and Python libraries
from dolfin import *
import numpy as np

PI = np.pi
Order = 4;
N_elements = 100;
# Create mesh and define nodal function space
mesh = UnitSquareMesh(N_elements, N_elements)
V = FunctionSpace(mesh, 'Lagrange', Order)

# Define boundary conditions
u0 = Expression('(sin(w*x[0])*sin(w*x[1]))',w=5*PI)
def u0_boundary(x, on_boundary):
  return on_boundary
bc = DirichletBC(V, u0, u0_boundary)

# Define variational problem
u = TrialFunction(V)
v = TestFunction(V)
f = Expression('2*w*w*sin(w*x[0])*sin(w*x[1])',w=5*PI)
a = inner(nabla_grad(u), nabla_grad(v))*dx
L = f*v*dx

# Compute solution
u = Function(V)
solve(a == L, u, bc)

# Calculating the L2-norm error
E = errornorm(u0, u, norm_type="l2", degree_rise=1)
Error.append(E)

print Error
```

A.5.5 MATLAB Implementation of the SCM

Below follows the MATLAB functions needed to apply the Stochastic Collocation Method to data obtained for a model problem which depend on a uniformly and normally distributed underlying random variable. Using these functions it is relatively straight forward to write a script for applying the method to data obtained from a set of deterministic simulations. A very basic example script is supplied at the end of the appendix.

PnG: Function for evaluating a gPC expansion at a desired set of points x.

```
function [ PnG ] = PnG( type, tilde_g, x, ab )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Evaluation of polynomial expansion P[g] at the points x for a function
% g given by a sequence of expansion coefficients for g => tilde_g
%
% Input:
% type [scalar]: type of polynomial expansion used based on PDF:
%   1: Uniform PDF: Legendre polynomials
%   2: Gaussian PDF: Hermite polynomials
% tilde_g [vector]: expansion coefficients for g calculated based on PDF of
%                  underlying random variable.
% x [vector]: a set of points on which P[g] is evaluated
% ab [vector]:
%  - Uniform: start and end point for the interval from which x was chosen
%  - Normal: ab(1) = mean, ab(2) = standard diviation
%
% Output:
% PnG [vector]: The evaluation of P[g] at the points x.
%
% Written by: Rasmus E. Christiansen
% Date: 15/03/2013
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Evaluating the expansion polynomials
Pn = zeros(length(x),length(tilde_g));
if (type == 1) % Legendre polynomials
    % transforming x to the standard defintion interval [-1,1]
    x_trans = (2 * x - (ab(1) + ab(2))) / (ab(2) - ab(1));

    for i = 1:length(tilde_g)
        Pn(:,i) = LegendreP_UnNormalized(x_trans,i-1);
    end
elseif (type == 2) % Hermite polynomials
    % Transforming x to fit the standard distribution N(0,1)
    x_trans = (x - ab(1))/ab(2);

    for i = 1:length(tilde_g)
        Pn(:,i) = HermiteP_UnNormalized(x_trans,i-1);
    end
else % Not defined
    error('Use ony 1: Legendre, 2: Hermite');
end

% Calculating the expansion
PnG = Pn*tilde_g(:);

end
```

PnG_Coefs: Function for evaluating the expansion coefficients of a gPC expansion using the interpolation approach $P_{N,D}[g](X_j) = g(X_j)$.

```
function [ tilde_g ] = PnG_Coefs( type , gX )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%
% Using the interpolation approach
%
% Evaluation the expansion coefficients tilde_g for the polynomial expansion
% P[g] of g based on the value of g at the collocation points x determined
% by the PDF for the quantity X.
%
% Input:
% type [scalar]: type of polynomial expansion used based on PDF:
%   1: Uniform PDF: Legendre polynomials
%   2: Gaussian PDF: Hermite polynomials
% gX [vector]: value of g at the collocation points
%
% Output:
% tilde_g [vector]: expansion coefficients for g calculated based on PDF of
%                   underlying random variable.
%
%
% Written by: Rasmus E. Christiansen
% Date: 15/03/2013
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Evaluating the expansion polynomials creating a vandermonde like matrix
VPn = zeros(length(gX),length(gX));

% Determining type of expansion
if (type == 1) % Legendre polynomials
    % Collocation points
    x = JacobiGQ(0,0,length(gX)-1);
    % Creating the Vandermonde-like matrix
    for i = 1:length(gX)
        VPn(:,i) = LegendreP_UnNormalized(x,i-1);
    end
elseif (type == 2) % Hermite polynomials
    % Collocation points
    % integral of weight function for normal distribution
    mu0 = 1;
    % 3-step Recurssion Relation A's and B's.
    B = zeros(length(gX),1);
    A = linspace(1,length(gX),length(gX));
    % Calculating the nodes
    x = GeneralGaussianQuadrature(A,B,mu0);
    % Creating the Vandermonde-like matrix
    for i = 1:length(gX)
        VPn(:,i) = HermiteP_UnNormalized(x,i-1);
    end
else % Not defined
    error('Use only 1: Legendre, 2: Hermite');
end

% Calculating the expansion coefficients
tilde_g = VPn \ gX;

end
```

LegendreP_UnNormalized: The function evaluates and returns the Nth Legendre polynomial at the points x along with the associated normalization factor g_N.

```
function [L,g] = LegendreP_UnNormalized(x,N)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Description:
% Function which evaluates the n'th order Un-normalised Legendre
% Polynomial at the points supplied in the vector x, based on a
% three term recursion formular.
%
% Input:
% x [vector]: Evaluation points
% N [scalar]: Order
%
% Output:
% L [vector]: the value of the Polynomial in the gridpoints x.
% g [scalar]: the normalization factor of the Polynomial.
%
% Written by: Rasmus E. Christiansen
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Assures that the points are stored as a row vector
x=x(:);

PL = zeros(length(x),N+1);
```

```
% Initial values P_0(x) and P_1(x)
PL(:,1) = 1.0;
PL(:,2) = x;

% Forward recurrence using the symmetry of the recurrence
for i=1:N-1
    PL(:,i+2) = (2*i + 1)/(i+1) * x .* PL(:,i+1) - i/(i+1) * PL(:,i);
end

L = PL(:,N+1);
g = 2/(2*N+1);
```

HermiteP_UnNormalized: The function evaluates and returns the Nth Hermite polynomial at the points x along with the associated normalization factor g_N.

```
function [H,g] = HermiteP_UnNormalized(x,N)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Description:
% Function which evaluates the n'th order Un-normalised Hermite
% Polynomial at the points supplied in the vector x, based on a
% three term recursion formular.
%
% Input:
% x [vector]: Evaluation points
% N [scalar]: Order
%
% Output:
% H [vector]: the value of the Polynomial in the gridpoints x.
% g [scalar]: the normalization factor of the Polynomial.
%
% Written by: Rasmus E. Christiansen
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Assures that the points are stored as a row vector
x=x(:);


PH = zeros(length(x),N+1);

% Initial values P_0(x) and P_1(x)
PH(:,1) = 1.0;
PH(:,2) = x;

% Forward recurrence using the symmetry of the recurrence
for i=1:N-1
    PH(:,i+2) = x .* PH(:,i+1) - i * PH(:,i);
end

H = PH(:,N+1);
g = factorial(N);
```

JacobiGQ: Function which calculates the Gauss quadrature points x and associated weights w of the Nth Jacobi polynomial $J_N^{(\alpha,\beta)}$. This function is taken directly from [14].

```
function [x,w] = JacobiGQ(alpha,beta,N)

% function [x,w] = JacobiGQ(alpha,beta,N)
% Purpose: Compute the N'th order Gauss quadrature points, x,
%          and weights, w, associated with the Jacobi
%          polynomial, of type (alpha,beta) > -1 ( <> -0.5).

if (N==0) x(1)= -(alpha-beta)/(alpha+beta+2); w(1) = 2; return; end;

% Form symmetric matrix from recurrence.
J = zeros(N+1);
h1 = 2*(0:N)+alpha+beta;
J = diag(-1/2*(alpha^2-beta^2)./(h1+2)./h1) + ...
    diag(2./(h1(1:N)+2).*sqrt((1:N).*((1:N)+alpha+beta).*...
    ((1:N)+alpha).*((1:N)+beta)./(h1(1:N)+1)./(h1(1:N)+3)),1);
if (alpha+beta<10*eps) J(1,1)=0.0;end;
J = J + J';
```

```
%Compute quadrature by eigenvalue solve
[V,D] = eig(J); x = diag(D);
w = (V(1,:)').^2*2^(alpha+beta+1)/(alpha+beta+1)*gamma(alpha+1)*...
    gamma(beta+1)/gamma(alpha+beta+1);
return;
```

GeneralGaussianQuadrature: Function implementing the Golub-Welsch algorithm, [21, Section 4.6.2], for calculating Gaussian quadrature points x and associated weights w for the class of orthogonal polynomials satisfying the three term recursion relation
$p_{n+1}(x) + (B_n - x)p_n(x) + A_n p_{n-1} = 0, n = \{1, 2, ...\}$.

```
function [ nodes, weights ] = GeneralGaussianQuadrature( A, B, mu0 )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Function for calculating general gaussian quadratures.
% Calculates the nodes and weights to use for gaussian quadrature
% with the orthogonal polynomials which satisfy the recursion
% relation:
%
% p_{n+1}(x) + (B_n - x) p_{n}(x) + A_n p_{n-1} = 0, n = 1,2,...
%
%
% Input:
% A [vector]: Coefficient of the polynomial p_{n-1}
%   Must be sorted as [A_0,A_1,...,A_{n-1}]
% B [vector]: Coefficient of the polynomial p_n
%   Must be sorted as [B_0,B_1,...,B_{n-1}]
% mu0 [scalar]: Integral of the weight function associated with the
%               polynomials
%
% Output:
% nodes [vector]:
% weights [vector]:
%
% Written by: Rasmus E. Christiansen
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
n = length(B);

% Building the matrix
M1 = diag(A(2:end),-1);
M2 = diag(B,0);
M3 = diag(ones(n-1,1),1);
M = M1+M2+M3;

% Determining the eigenvalues and vectors from which nodes and weights are
% found
[weights,nodes] = (eig(M));
% Extract the nodes
nodes = diag(nodes);
% Sorting the nodes in ascending order
[nodes,id] = sort(nodes);

% Calculating the weights
weights = mu0 * weights(1,:).^2;
% sorting the weights to correspond to the sorted nodes
weights = weights(id);

end
```

Example Script: A simple example of applying the SCM for calculating a gPC expansion based on a uniformly distributed random variable. The mean and variance are also calculated and results plottet.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Script: SCMexecution.
%
% Description: Applies the Stochastic Collocation Method for uncertainty
% quantification based on a uniform distribuiotn.
% The resulting polynomial expansion along with its mean and mean plus
% and minus a standard deviation are plottet.
%
% Written by: Rasmus E. Christiansen
% Date: 15/03/2013
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initialization
clc
clear all
close all

% Uncertain Quantity interval for uniform distribution.
ab = [50,600];
% Collocation points
X = [75.8005;176.9209;325;473.0791;574.1995];
% Value of g at X
gX = [0.99088;1.06843;1.180358;1.261836;1.298412];

% Calculating discrete projection coefficients
tilde_g = PnG_Coefs( 1 , gX );

% Calculating the needed normalization factors: g_N = 2/(2*N-1)
gamma_k = 2./(2*(2:1:length(gX))-1)';

% Calculating mean and variance
mean = tilde_g(1);
variance = sum(tilde_g(2:end).*tilde_g(2:end).*gamma_k);

% Plots
x_eval = linspace(50,600,100);
semilogx(x_eval, PnG( 1, tilde_g, x_eval, ab ));
hold on;
semilogx(X,gX,'o');
semilogx(x_eval,mean*ones(length(x_eval),1),'r');
semilogx(x_eval,(mean + sqrt(variance))*ones(length(x_eval),1),'r--');
semilogx(x_eval,(mean - sqrt(variance))*ones(length(x_eval),1),'r--');
```