TALLINN UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Department of Computer Systems

Abiola Taofeek Rashidi

Investigation Of Interpolation Methods For The Distribution Of Local Pressure And Skin Friction Around A Circular Cylinder

Master's Thesis

Supervisor(s): Associate Professor, Jeffrey Andrew Tuhtan

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia Teaduskond

Arvutisüsteemide Osakond

Abiola Taofeek Rashidi

Interpolatsioonimeetodite uurimine kohaliku rõhu ja nahahõõrdumise jaotamiseks ringikujulise silindri ümber

Magistritöö

Juhendaja: Associate Professor, Jeffrey Andrew Tuhtan

Tallinn 2023

Declaration: I hereby declare that this Master's thesis, my original investigation and achievement, submitted for the Master's degree at Tallinn University of Technology, has not been submitted for any degree or examination.

Deklareerin, et käesolev magistritöö, mis on minu iseseisva töö tulemus, on esitatud Tallinna Tehnikaülikooli magistrikraadi taotlemiseks ja selle alusel ei ole varem taotletud akadeemilist kraadi.

Abiola Rashidi

Date: 14.05.2023 . . . . . .

Signature: . . . . . .

Contents

Nomenclature

2D	Two-Dimensional
3D	Three-Dimensional
BDS	Biotrauma Detection System
csv	Comma-delimited value (file format)
g	gram
IDE	Integrated Development Environment
kPa	Kilo-Pascal
m	meters
MAE	Mean Average Error
mbar	Mili-bars
mm	Millimeter
Nm	Newton meter
Pa	Pascal
Re	Reynold's Number
RMSE	Root Mean Square Error
tar	Tape archive(file format)

Acknowledgment

I want to thank my supervisor for his patience, timely feedback, immersed support and importantly his wisdom for carrying me through to the very end of writing this thesis. Also, I could not have completed this work without the support of my family, especially my spouse and son who stood by me emotionally and spiritually.

I am grateful to my manager at work who from time to time showed his concern about me and my studies, as well as my colleagues who helped in language translation and proofreading. Lastly, to my parents, especially to my late mum who pass on to Glory during my studies, the very last time we spoke and the very last words, “you will succeed in Jesus Name”.

Abstract

Interpolation methods are widely used in various scientific fields to estimate values between discrete data points. In the laboratory, interpolation methods are used to predict values for unmeasured variables based on existing data. However, field data collection often occurs under highly uncertain conditions, which can lead to incomplete or sparse datasets. In this thesis, the author explores the use of interpolation methods for lab data to interpret field data collected under highly uncertain conditions by reviewing the different types of interpolation methods and their strengths and weaknesses. This work also evaluates the effectiveness of various interpolation methods in accurately predicting missing data points and estimating the overall trends in the data as well as analyze the impact of different levels of uncertainty on the accuracy of the interpolation results. It was found that spline methods performed well in predicting missing data points, while kriging methods for surface interpolation is more effective in capturing the overall trends in the data as it takes into account the spatial correlation of data and provide a smooth continuous surface.

Keywords: Interpolation methods; underwater data analysis; freshwater fish; downstream passage; barotrauma detection sensor; skin friction; pressure

List of Figures

List of Tables

Chapter 1

Introduction

Over 58,700 single and multiple purpose dams have been built worldwide as of 2020 [1], which include flood control, fish farming, hydropower, irrigation, water supply, recreation, navigation, tailing, and others, with hydropower accounting for about 16% of the world's source of power supply. There has been a well-established center around the extreme physical issue dangers to fish because of hydroelectric turbines, wounds and passings brought about by entrainment and impingement during downstream entry through hydropower dams can possibly hurt fish populaces [2] [3], as well as prevent them from moving between feeding and breeding grounds, disrupting their lives and limiting their reproductive capabilities. Mechanical contact with the turbine, share force turbulence, and pressure changes in water flow can injure or kill fish [4]. Shear caused by rapid changes in water velocity can cause physical damage to fish, including hemorrhaging, tissue damage, and scale loss, which can lead to mortality in extreme cases [5]. Pressure-related fish injury and mortality can occur due to pressure nadir, changes in pressure along with the change in log pressure ratio and to determine whether fish can pass and survive during downstream migration, barotrauma sensors for hydropower turbines have been developed [5]. The barotrauma detection system (BDS) sensors are fault-tolerant sensors that can travel through hydropower turbines. They've also been used successfully in glaciers, pressure pipelines, waterfalls, over weirs, and under gates. They are tough and built to record pressure and inertial data in the harshest underwater conditions. Key features include fault-tolerant pressure sensors with triple modular redundancy to ensure

the quality of each individual measurement, absolute orientation of the sensor relative to gravity and magnetic North, self-calibrating pressure sensors (atmospheric) and inertial sensors (accelerometer, magnetometer and rate gyro).

1.1 Problem Statement

Fish populations may be negatively impacted by injury and mortality brought on by entrainment or/at the time of passing downstream through/over hydropower infrastructure [2]. Pressure, shear, and impacts are three known categories of physical mechanisms that harm and kill fish. There do not exist measurement devices for the direct measurement of fluid shear under the extreme states of being capable during downstream turbine section. One of the goals of this thesis will be to take flow information from underwater measurements and visualize them in a unique way. The main objectives of this thesis were to address the following two research questions:

(1) Which is interpolation method is the best performing for the skin friction and pressure dataset?

(2) Which interpolation methods are suitable for creating surfaces to estimate the skin friction and pressure difference for Reynolds numbers without direct measurement data?

1.2 Objectives and Contributions

In this study, pressure difference measurements from the BDS in the field were converted into estimates of skin friction by using data from a well-known laboratory experiment that linked pressure differences to shear (skin friction) and this experimental results were digitized and subjected to interpolation methods.

Secondly, data from the experiment were visualized in a unique way that will aid fluid dynamics researchers in understanding turbulent flow and as well help to understand migratory pathways of fish around the hydropower dams. Natural flows are frequently very different from those observed in the laboratory. The research looks into how those differences affect biological organisms, particularly fish. Many aquatic animals have evolve

sophisticated sensory systems that function in turbulent flows. Turbulence includes both fast and slow vortices, as well as large and small vortices, and comparing laboratory and natural flows which is difficult using standard methods. The main contribution of this work is to test and develop an interpolation method which allows for the BDS pressure difference data to be converted into skin friction (shear).

1.3 Thesis Outline

The first chapter of this work covers the introduction, start-of-the-art and the thesis objectives. Chapter 2 described the architectural design and functionalities of the existing barautrauma detection system (BDS). It also highlight how the BDS structural designed was modified for the purpose of this research, the data extraction process and digitization of the conveyance of nearby strain and skin erosion around a roundabout chamber in cross-stream up to Reynolds Number $Re = 10^6$, the spline interpolation theory and validation of the extracted data and interpolated data and also finding the RMS errors. Chapter 3 described how the data acquisition process using BDS in the field and laboratory respectively, data analysis and visualization, result analysis and comparison. Python programming language was used in the developing program that transformed the raw data into a user-friendly application. Chapter 4 contains the surface interpolation with the goal for smooth surface creation and the surface that is able to pass through the sample points, allowing for the estimation of values at other locations on the surface and the last Chapter is summary of the activities carried out in the thesis, the result and the conclusion drawn based on the analysis carried out and outlines the recommended future work.

Chapter 2

State of the Art

The worries about the impacts of manmade dams on the population of fish have been developing alongside the quantity of dams worked across the globe. Dams might work as a hindrance to resident (i.e., those that finish their life cycle inside a supply or section of the stream) and transient (i.e., anadromous, catadromous, and potamodromous) fish, fragmenting rivers and harming ecosystems [2]. The detrimental effects of dams on diadromous fish migration upstream are well known, and several fishways have been installed to make up- stream transit easier [6]. However, it is still difficult for fish to migrate downstream near dams [7,8] depending on the stage of their life cycle, migrating fish may need to cross dams and continue downstream in order to reach their respective spawning grounds (for catadromous species) or raising and feeding areas (for iteroparous species). For breeding, rearing, and foraging purposes, within a river system, local species may cover large distances, or they may merely move within reservoirs in which they could travel via wetlands and estuaries. Fish populations may suffer significant effects as a result of entrainment or impingement(which happens when fish become stuck against infrastructure) and fish passage, which happens when fish (non-)voluntarily migrate through hydroelectric equipment, both of which are connected to hydroelectric plants [9].

To quantify mortality across turbine types and fish species, Radinger et al. assembled and analyzed a world- wide dataset of turbine fish-mortality evaluations encompassing >275,000 individual fish of 75 species [4] thinking about the typical mistakes related with observational evaluations, the rate of death of fish in general is brought about by

hydroelectric turbines was 22.3% (95% CI 17.5-26.7%). Estimates of mortality varied considerably between and even within individual turbine designs, research approaches, and taxonomic groups [5]. The demand for renewable energy and the conservation of fish biodiversity may be reconciled if hydroelectric turbines could be configured technically in a way that would significantly minimize fish mortality, and if this method became the worldwide norm. With the utilization of latent sensors that record the tension, speed increase, and pace of turn or rotation angle, Pauwels et al. looked at the danger of damage and mortality for many species of fish as they passed downstream through a huge Archimedes hydrodynamic screw [5].

Several novel measures, including as impact event timing and duration, translational and rotational kinetic energy, and pressure gradients are proposed in this study for evaluating downstream passage. The outcomes of the study described that bream had the highest mortality rate (37%) of the three species studied, followed by roach (19%), and eel (3%) on average. For just a few species-specific injury and fatality rates did the operating scenario prove to be statistically significant [5]. An information based on sensor showed extremely tumultuous actual conditions in the screw for Archimedes hydrodynamics in contrast to research with Kaplan turbines, where there was no variance in physical metrics across operating situations. Pollution, over fishing, global warming, coastal growth, and other human activities all pose significant risks to the world's fish stocks. A comprehensive review was conducted to investigate the impacts of entertaining the fish and impingement at hydroelectric dams on freshwater fish and their production in mild environments, along with the role of type of site, type of mediation and viewpoints life cycle.

The review looked at both mainstream and alternative sources of information (commercially published and grey literature) in accordance with the collaboration for environmental evidence's standards [2]. In total, 87 publications included 264 studies that were evaluated critically and synthesized in narrative form. The majority of research (93%) focused on species within the Salmonidae family, and the majority of studies (86%). In terms of minimizing fish damage and death, their analysis reveals that bypasses are the "fish friendliest" transit option as they looked at how freshwater fish may be harmed or killed as they made their way through standard hydroelectric facilities [2]. various kind of investigative work and researches on the frameworks beyond North America, on species

other than salmonids and sportfish, and on the impacts of entrainment and impingement on the population of fish is expected to fill in the holes in our comprehension. For downstream section at projects in the US, government controllers and organizations answerable for hydropower oversight frequently request evaluation studies and moderation to address adverse consequences, with the primary target of keeping away from fish impingement and turbine capture and death. Evaluations of entrainment and impingement rates, all out downstream entry endurance, turbine endurance, spillway door section, and Oberymeyer door section were made for an exceptional little water based projects or hydro-projects on the Mississippi Waterway to assess the downstream passage's impact on fish populations. The garbage racks were designed such that only 15% of the fish that would normally swim downstream would be physically blocked by the narrow bar spacing with a bypass rate of 10% for the Obermeyer gates and a bypass rate of 90% for the gates, the anticipated overall project survival rates are 77.3% and 96.6%, respectively, when 55% of the river flow reaches the turbine intake channel [10].

Another research looked at the tolerance of a surface bypass with varying aperture diameters and the damage sustained by fish during the passage. Bypass's overall acceptability was lower than that of the turbine passage, the quantity of fish swimming downstream did not change much whether the bypass apertures were modest or big. The quick death of any fish species was not seen, amputations and other severely damaging injuries were recognized seldom and weakly at best. The most frequent injuries were lost scales, fin rips and hemorrhages, and skin lesions on the body, with notable species-specific variances [11]. Offering a bottom bypass as a secondary option to the current surface bypass is likely to be helpful in increasing bypass efficiency. By making structural changes to the bypass, such as concealing projecting components, the risk of bypass-related injuries might be substantially mitigated. Alves et al. outlined a technique for determining how hydroelectric dams affect the survival of fish larvae as they pass through the dams' hydraulic components, filling in knowledge gaps that currently exist [12]. Presented and addressed are potential options to aid in mortality reduction and hydraulic structure management, inferences on the efficacy of fish relocation programmes, and a sampling and analytic strategy suited to measure the impacts of passage by the larvae via dams in situ. In situ evaluation of downstream ichthyoplankton transit via dams is made possible with

the suggested method [12] to determine how downstream transit via dams operating under different regimes correlates with fish injuries and deaths, the suggested sample approach is simple, readily conducted, and economical in comparison to alternatives requiring more complex technology or that combine field and lab investigations.

The fast growth of the hydroelectric industry, particularly in the major tropical basins, highlights the significance of its widespread adoption, dams are being built at a rapid rate all around the globe, and this has sparked growing worries about the impact on fish populations. Damage to fish populations can result from three main causes at hydroelectric facilities: migratory fish passing through the facilities on their way downstream, fish impinging on screens and trash racks, and resident fish becoming entrained. Given the importance of both resident and migratory fish to overall fish production, it is imperative that we assess the effects of fish damage and death due to entrainment and impingement at hydroelectric dams. As a result, knowing how fish impingement and entrainment related to hydroelectric dams affect fish productivity is important for ensuring the long-term viability of fish species that rely on our freshwater ecosystems [13]. It was suggested by Mueller et al. to use a dataset of 52,250 fish to conduct a complete evaluation of traditional and novel hydropower [14]. Kaplan turbine locations had the highest fish mortality (83%) due to hydropower. While innovative hydropower is frequently hailed as "fish-friendly," a recent study found that 64 percent of fish died as a result and according to our results, the number of turbine blades, the amount of turbulence at the turbine outputs, and the runner's peripheral speed are the most crucial variables [14]. Optimal turbine technologies and operating modes need to take site-specific variables like head drop, bypass possibilities, and river-specific species composition into account more thoroughly to lessen the effect of hydropower on fish collision, fast decompression, and fluid shear are the three most prevalent stresses that fish face while passing through hydroelectric plants.

Ninety-nine biological reaction models have been created using specialized equipment to simulate the effects of blade hit, fast decompression, or fluid shear and there are 31 distinct species of fish included in the models, and they all have different expected outcomes [15]. Significant species-to-species diversity in vulnerability to the stressors has been identified across these models, and the sensitivity of one species to one stressor does not always predict equivalent susceptibility to another. While the responses of a number of species

to various stressors have been studied, it is still unknown how many additional species, which may have distinct physical features, may react. These models may and have been implemented in a number of scenarios, to better comprehend the risk of injury or death to fish during transit at hydropower facilities, Hydropower Biological Evaluation Toolset (HBET) and Biological Performance Assessment (BioPA) are often used. These models have also been used. This encompasses a wide range of applications, some of which include the replacement of turbines, the installation of new turbines, and the change of the operations of turbines that are already in existence. Tools like BioPA and HBET, when combined with integrated biological response models, may help engineers design hydropower systems that have minimal ecological impact.

2.1 Barotrauma Detection Sensor

The Barotrauma detection center is a multi-modular submerged sensor that screens outright direction (roll, pitch and yaw points), unbiased lightness, attractive field strength, rate of rotation, straight speed increase, and complete water pressure. In the structure of the European Association's H2020 FITHydro project, the Tallinn College of Innovation (Tal-Tech) Biorobotic community made the sensors [5]. Exposure to events like decompression, collisions, and extreme turbulence may be determined by analysis of data collected by the BDS sensor unit. In section 2.1, we get an overview of the various sensors. The BDS's neutral buoyancy may be adjusted in the field by increasing or decreasing the volume of the device by rotation of the flat end cap. Three identical digital total pressure transducers [5] are installed within the hemispherical end cap to measure the combined atmospheric, hydrostatic, and hydrodynamic pressures. Laboratory tests confirmed the precision are installed within the hemispherical end cap to measure the combined atmospheric, hydrostatic, and hydrodynamic pressures. According to Pauwels et al, the laboratory tests confirmed the precision each BDS is pressure tested in a barochamber up to $550kPa$, or 2.75 times the sensor's maximum rated pressure, and compared to a commercial pressure sensor. The triple modular redundancy provided by the extra sensors makes up for the vulnerability of a single sensor. There are three pressure transducers available, and each may be set to measure pressures between 200 and $3000kPa$. The two AAA

batteries and the Barotrauma detection sensor are housed in a bespoke waterproof casing that is $140mm$ long, $40mm$ in diameter, and has two machined polyoxymethylene end caps for a total dry mass of $147g$ [5]. The reference value for the BDS sensors is set to $1000mbar$ and they are designed to automatically compensate for the local atmospheric pressure. To change the pressure measurements there is no reason especially to a standard air pressure datum in post-handling since this is the only data processing that happens during sensor deployment. Overview of the BDS sensors with labels which describe the various part of the BDS and what they represent Figure 2.1.

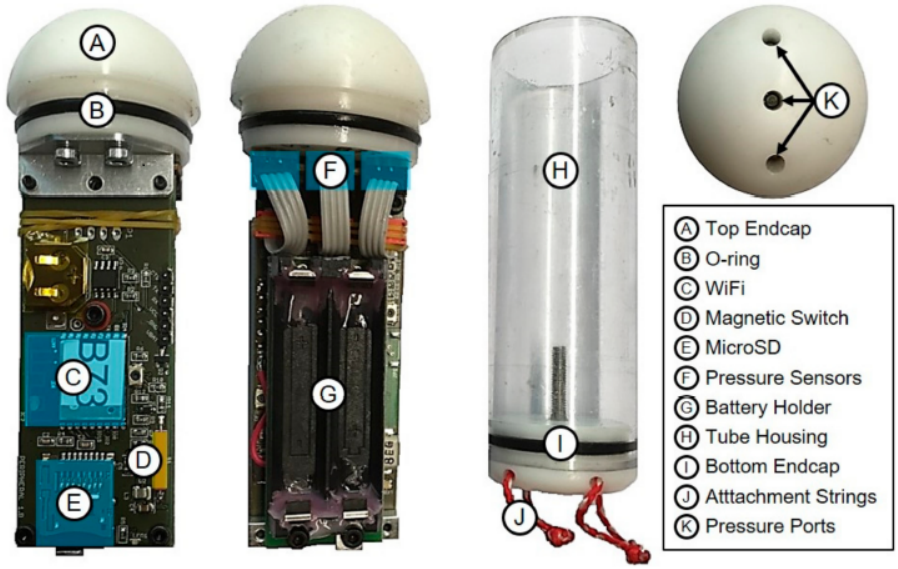


Figure 2.1: Barotrauma Detection Sensor ($image\ source$, Pauwels et al). The top end-cap (A,B) –contains three pressure transducers–(F,K). Below there are two electronics boards containing the WiFi module–(C), magnetic switch–(D), microSD storage– (E), AAA battery holder–(G). The sensor and electronics payload (A–G) is screwed by hand onto the bottom end-cap (I), which also includes two rugged nylon attachment strings (J) for the balloon tags to bring the neutrally buoyant sensor back to the water surface.

2.2 Distribution of local pressure and skin friction around a circular cylinder in cross-flow up to $Re = 5 \times 10^6$

In 1968, E. Achenbach conducted research on skin friction and the local pressure distribution around the cylinder that is circular in shape in cross-flow up to $Re = 5 \times 10^6$ at the Institute für Reaktorbauelemente, KFA-Jülich, Germany, and obtained some experimental

results that were plotted on graphs as shown Figure 2.2, 2.3 and 2.4. Different Reynolds values and $\phi = 5^\circ$ increments around the cylinder's circumference ($\phi = 360^\circ$) were used to find out the distribution of skin friction and pressure. All three diagrams show the skin friction and pressure distribution as a function of the cylinder's peripheral angle [16]. Localizing point of separation is achieved by analyzing the skin friction distribution. It is possible to distinguish three distinct flow regimes based on these measurements: the subcritical flow, characterized by bubble separation and subsequent turbulent reattachment; the supercritical flow, characterized by an unexpected change from the laminar to the tempestuous limit layer at a basic separation from the stagnation point; and the progress between the two.

2.2.1 Skin friction Distribution

The skin rubbing around a roundabout chamber in cross-stream is likewise portrayed by a dimensionless boundary called the coefficient of friction of skin. The coefficient of friction of skin is characterized as the powerful strain of the stream partitioned by the extraneous power per unit region following up on the outer layer of the chamber. The approach and the Reynolds number influence the skin grating coefficient. The coefficient of friction of skin remains largely constant as one moves around a cylinder at low Reynolds numbers, while at higher Reynolds numbers, it decreases along the forward-facing part of the cylinder and increases along the rear-facing part of the cylinder [16]. The equation (2.1), shows how $Achenbach$ [16] determined the skin contact from the encompassing of the stagnation point from the conveyance of the strain around the chamber. As per Achenbach, the condition was fetched by a layered examination in a limit layer for a test boundary. he considered the viscosity η of the fluid and the length, h or the height of the edge of the cylinder.

$$\frac{\triangle p}{\tau_0} = f\left(\frac{\triangle p h^2 \rho}{\eta^2}\right) \tag{2.1}$$

where:

$\triangle p$ is the pressure difference (Pa)

η is the viscosity of the fluid (Nsm^{-2})

h is the length (m)

ρ is the density (Kgm^{-3})

p is pressure (Pa)

$$T = \frac{\tau_0}{\rho U_\infty^2}\sqrt{Re} \tag{2.2}$$

T is temperature $({}^0C)$, τ_0 is the wall-shear stress (Pa) and $U_\infty$is the undisturbed velocity (ms^{-1})

2.2.2 Pressure Distribution

A pressure coefficient describes the pressure distribution around a roundabout chamber subject to cross-stream. Separating between the powerful strain of the stream and the local pressure yields the pressure coefficient. The pressure coefficient changes depending on the flow's Reynolds number and angle of attack. As the Reynolds number decreases, the pressure distribution around the cylinder becomes more symmetrical, and the stagnation point experiences its maximum pressure coefficient (where the flow velocity is zero) [16].

$$P = \frac{p - p_\infty}{(1/2\rho)\, U_\infty^2} \tag{2.3}$$

where:

P is the pressure coefficient(Pa)

p_∞ the static pressure of the infinite flow (Pa)

ρ is the density (Kgm^{-3})

p is pressure (Pa)

U is the undisturbed velocity, (ms^{-1})

Increases in Reynolds number cause an uneven distribution of pressure, with the maximum value of the pressure coefficient occurring at an angle of approximately 80-90 degrees from the stagnation point.

2.2.3 Achenbach Pressure and Skin Friction Distribution Results

Different Reynolds values and $\phi = 5°$ increments around the cylinder's circumference ($\phi = 360°$) were used to determine the pressure and skin friction distributions. All three diagrams show a relationship between the cylinder's peripheral ϕ and the friction o f skin and strain conveyance [16]. Figure 2.2 – 2.4 shows the strain dissemination or distribution of pressure and friction of skin as a function of the cylinder's peripheral angle. Subcritical flow at $Re = 10^5$ is seen in Figure 2.2, where the boundary layer begins to laminarly divide at $\phi = 78^o$ ($\phi = 282^o$) before reaching the main cross-section. The loss of skin friction is a sign of detachment.

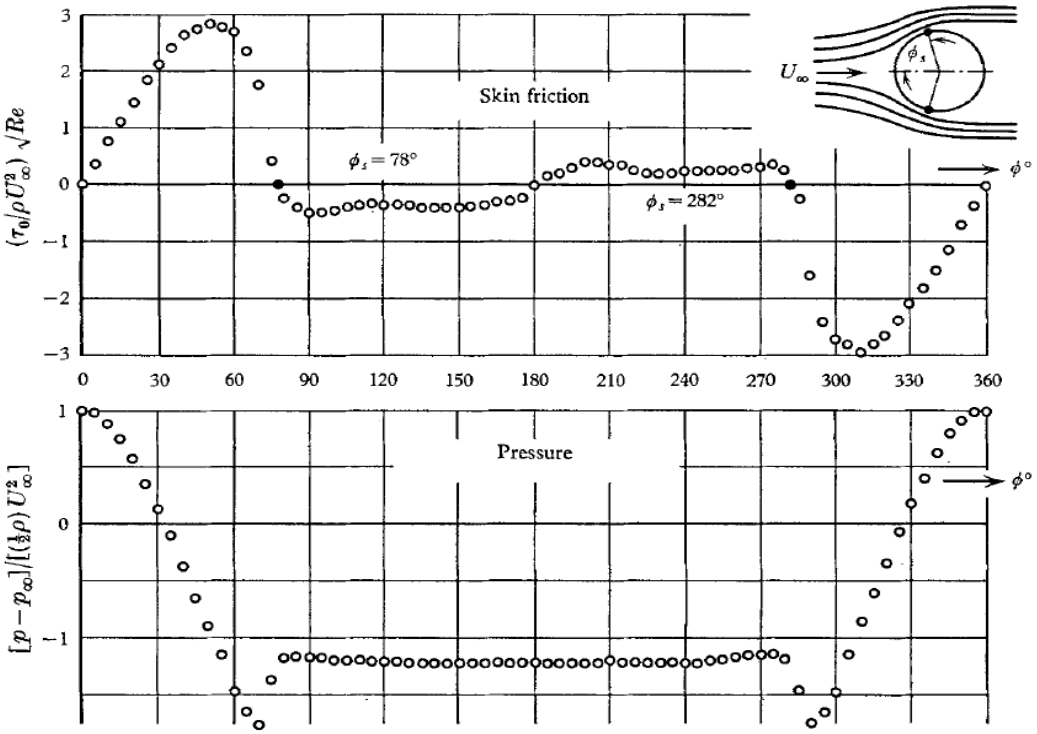


Figure 2.2: Circular cylinder: skin friction and pressure distribution. Re= 10^5

$Re = 2.6 \times 10^5$, the flow behavior is seen in Figure 2.3; at $Re = 3 \times 10^5$, the transition

into the critical zone occurs. Even at an angle of $\phi = 94^o$ ($\phi = 266^o$), the boundary layer separates laminarly. When compared to the flow at $Re = 10^5$, both the amplitude and location of the minimum have shifted. The drag coefficient decreases as the dimensionless pressure at the rear of the cylinder increases [16].

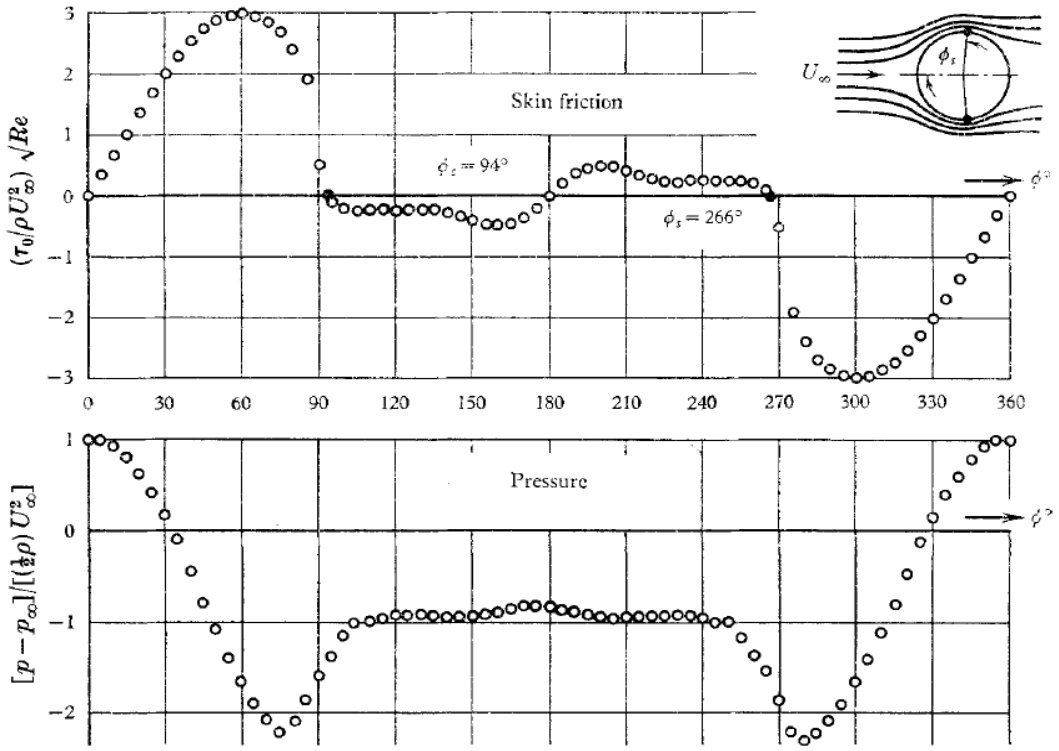


Figure 2.3: Circular cylinder:skin friction and pressure distribution. $Re = 2.6 \times 10^5$

The normal distribution of skin friction in the crucial area is seen in Figure 2.4. There is not complete separation at $\phi = 105^o (\phi = 225^o)$, location of separation. This implies that the wall shear strains should theoretically disappear in the transition zone between laminar detachment and turbulent reattachment. The skin friction then rapidly increases downstream, reaching levels that are often higher than the laminar maximum. The boundary layer is likely turbulent at an angle of $\phi = 147^o (\phi = 220^o)$, when the two layers begin to physically separate [16].

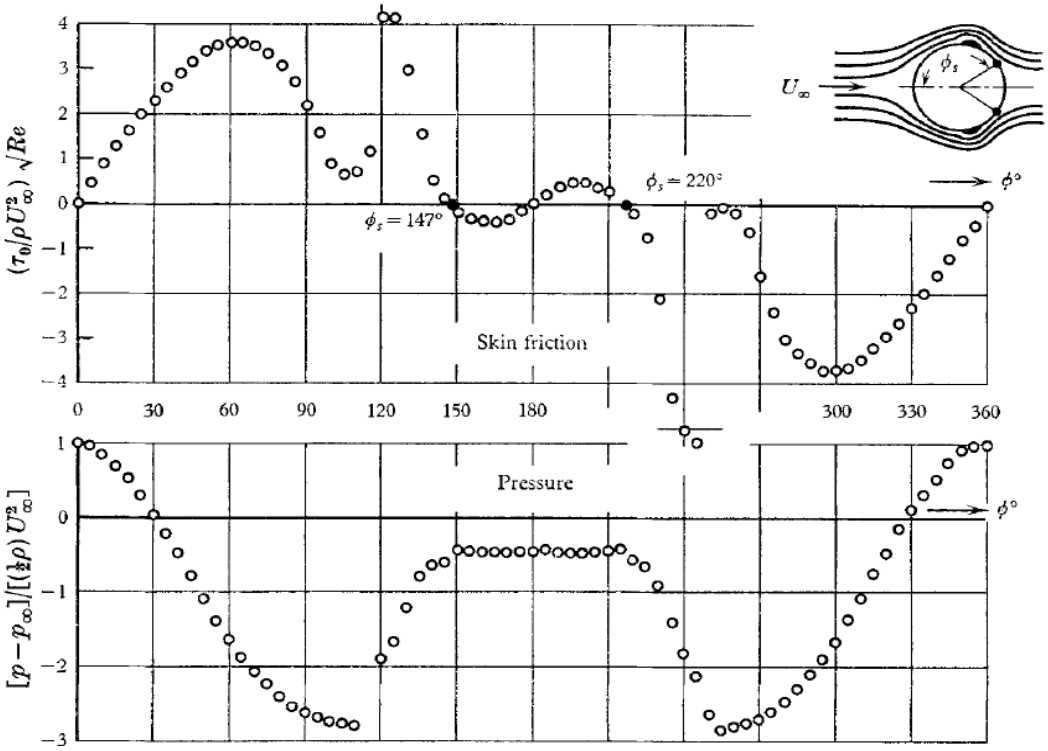
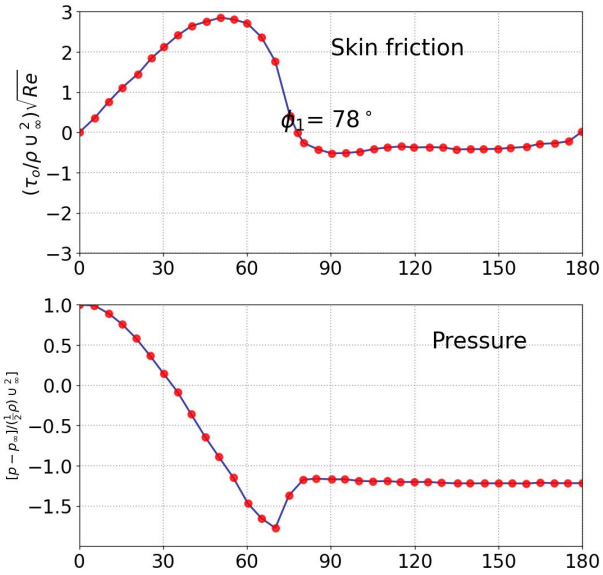


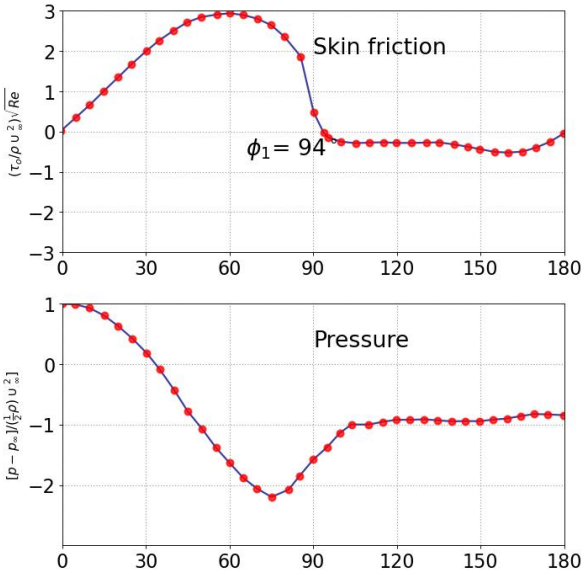
Figure 2.4: Circular cylinder:skin friction and pressure distribution. $Re = 8.5{\times}10^5$

2.2.4 Digitization of Achenbach Pressure and Skin Friction Distribution Graphs

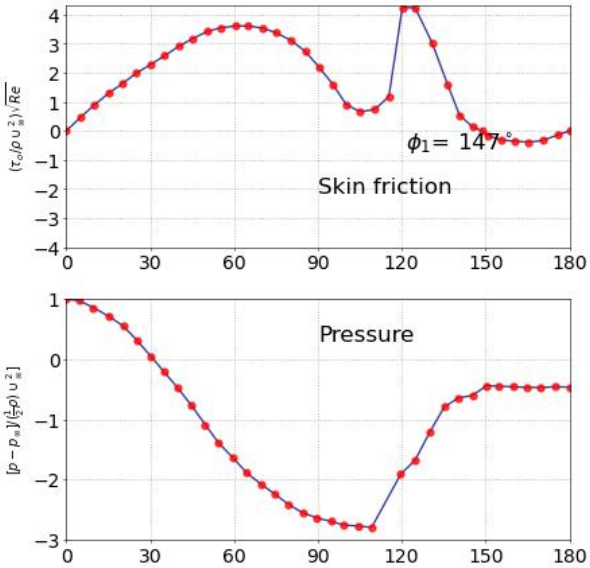
$Method$: The data was extracted using a special software called WebPlotDigitizer [17], the numerical data was carefully captured in the range of 0 to $180°$ in selecting the data points and a desired datasets were obtained for all three Figure 2.5–the data was extracted at the same vertical scale but horizontal scale 0 - 180° peripheral angle ϕ of the cylinder for each graph. The data extraction/digitization and visualization methods are described further below. The WebPlotDigitizer [17] was used to extract data from the Achenbach experimental result, which aided in reverse-engineering the graphical images in Figure 2.2 – 2.4 and allowed the underlying numerical data to be extracted. It allows for precise adjustment of every data point between the X and Y axes to produce the correct list of datasets (see Appendix C).



(a) Skin friction and pressure $Re = 10^5$



(b) Skin friction and pressure $Re = 2.6 \times 10^5$



(c) Skin friction and pressure $Re = 8.5 \times 10^5$

Figure 2.5: Graphs after digitization of the Achenbach experimental result

This extracted dataset has not been processed and does not need to be cleaned. The project file containing this data is contained in the $.tar$ file in https://github.com/abkisssb/Distribution-of-Local-Pressure-and-Skin-Friction-Around-A–Circular-Cylinder-In-A-Cross-Flow-.git , and the dataset was exported as a $.csv$ for use in matplotlib visualization, which will be discussed next.

$Visualization$: The captured datasets were saved in a usable data format, $.csv$, with no

data cleaning required. Matplotlib, a comprehensive Python library for creating interactive visualizations, was used to re-plot a new graph using these datasets. The Jupyter notebook IDE was used to create the code for plotting the various datasets obtained (see Appendix A) are the graphs for the three different $Reynolds$ numbers after digitization which were plotted on a horizontal scale of 0 – 360^o for each of the graphs, and $0 - 180^o$ for the newly digitized graphs, Figure 2.5. The dataset derived and how it was used to generated the graphs can be accessed through the link https://github.com/abkisssb/Distribution-of-Local-Pressure-and-Skin-Friction-Around-A–Circular-Cylinder-In-A-Cross-Flow-.git.There are three folders in the link, in each folder is a $.ipynb$ file, when clicked opens the code which generated the graphs as shown in Figure 2.5.

Chapter 3

Methods

3.1 Interpolation

Mathematically, the process of interpolation of creating new data points that fall inside the bounds of a discrete set of already established data points [18]. It also involves estimating a function that passes through a known data point and can be used to calculate the value of say y for any new value of x within the range of the known data [18]. Generally, interpolation problem statement involves having a known data point $(x_1, y_1), (x_2, y_2) \ldots, (x_n, y_n)$ Figure 3.1, where each x_i and y_i represents a known data values so we can find the value of y for a new value of x which is not part of the original set of data points [18]. The issue explanation in this setting is the dataset extricated from Achenbach research on the conveyance of neighborhood strain and skin contact around a round barrel shaped in crossflow up to $Re = 5 \times 10^6$ to predict y, given x, where x is the angle ϕ, and y is the vertical value on either the pressure or skin friction.

3.2 2D-Interpolation Methods

The second part of the experiment involves interpolation of the regenerated or extracted datasets for the three Reynolds numbers ($Re = 10^5, Re = 2.6 \times 10^5, Re = 8.5 \times 10^5$), as the estimated values between the observed data points varies. The main reason

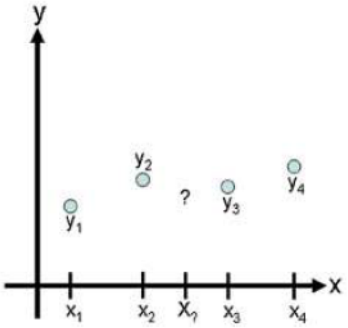


Figure 3.1: Illustration of the interpolation problem: estimate the value of a function in-between data points.

for interpolation was because the points were plotted by painstaking tracing each data points which subjected the dataset to some level roughness or inconsistency and required interpolation to smoothen out the irregularities in the plots making it easier to analyze and visualized. Another aim of applying interpolation was to re-sample the data at different resolution then the original data plotted from the Achenbach results Figure $2.2 - 2.4$, as document containing the original data had to be zoomed from very low resolution which allowed for plotting between two points easier. There are various kinds of techniques for interpolation that are regularly employed, but the method utilized is always determined by the nature of the data and the intended application of the interpolated values. With the objective of smoothing out the data that is noisy, three different types were chosen according to the types of data utilized in this study.

3.2.1 Linear Interpolation:

This is the simplest of all the three methods chose for this analysis and visualization, in which a straight line is used to connect two adjacent data points, where the value of the unknown point is determined by calculating the slope of the line and finding the $y - intercept$ [18].

$$y\left(x\right) = y_i + \frac{\left(y_{i+1} - y_i\right)\left(x - x_i\right)}{\left(x_{+1} - x_i\right)} \tag{3.1}$$

Wherey_i and y_{i+1} are the values of the two known y-values data points and x_i andx_{i+1} are the values of the two $x - values$ data points, x is the unknown estimated value and y is the estimated value, both x and y are directly proportional, that is any changes in y will cause a change in x.

```
def linear_interpolation(x, y):
    """
    Linear interpolation function.
    x: original x values (list or numpy array)
    y: original y values (list or numpy array)
    """
    # Find the index of the x value just smaller than x_new
    x_ = np.linspace(0, 180, 181)
    y_ = np.array([])
    for x_new in x_:
        i = np.searchsorted(x, x_new) - 1
        # Check if x_new is out of range
        if i == len(x) - 1:
            y_ = np.append(y_, [y[i]])
        elif i == -1:
            y_ = np.append(y_, [y[0]])
        # Perform linear interpolation
        else:
            y_ = np.append(y_, [y[i] + (y[i + 1] - y[i]) * ((x_new - x[i]) / (x[i + 1] - x[i]))])
    return x_, y_
```

Figure 3.2: Python function for linear interpolation (see Appendix for full code)

3.2.2 Polynomial Interpolation:

This is another type of interpolation method of finding a polynomial function that passes through a given set of data points. Say we have a polynomial of degree $n - 1$, and number of data points are denoted by n to fits the data as closely as possible [19, 20]. From (2) show the equations for polynomial of degree n-1 with data points in the form $(x_1, y_1), (x_2, y_2) \ldots, (x_{n+1}, y_{n+1})$ where are the coefficients.

$$p\left(x\right) = a_{n+1}x^n + a_n x^{n-1} + \ldots + a_3 x^2 + a_2 x + a_1 \tag{3.2}$$

$$y_1 = a_0 + a_1 x^1 + a_2 x_1^2 + \ldots + a_{n-1} x_1^{n-1}$$

$$y_2 = a_0 + a_1 x^2 + a_2 x_2^2 + ... + a_{n-1} x_2^{n-1}$$

$$y_n = a_0 + a_1 x^n + a_2 x_n^2 + ... + a_{n-1} x_n^{n-1}$$

Using linear algebra techniques like Gaussian elimination, it's possible to solve n equations of the form where the coefficients of the n equations can be found, and that polynomial can be used to interpolate the values at any point along the data range [19]. There is a tendency for polynomial interpolations to behave erratically outside the range of the data, known as overfitting [19]. The subsequent chapter explains the problem that was encountered and how it was fixed. [19].

```
def polynomial_interpolation(x, y):
    """
    Polynomial interpolation function.
    x: original x values (list or numpy array)
    y: original y values (list or numpy array)
    degree: degree of the polynomial (int)
    """
    # Fit a polynomial to the data
    coefficients = np.polyfit(x, y, 50)
    # Create a polynomial function
    polynomial = np.poly1d(coefficients)
    x_ = np.linspace(0, 180, 181)
    y_ = np.array([])
    for x_new in x_:
        # Calculate y_new
        y_new = polynomial(x_new)
        y_ = np.append(y_, [y_new])
    return x_, y_
```

Figure 3.3: Python function for polynomial interpolation (see Appendix A for full code)

3.2.3 Spline Interpolation:

Spline is the third method used in the investigation. It is a piecewise-defined function that is fitted to a set of data points. It is defined by a set of polynomial functions that are smoothly joined together at the data points [18, 19]. Interpolation between data points is performed using a second-degree polynomial using quadratic splines [21] . Take the example in Figure 3.4, where there are $n+1$ data points $(x_1, y_1), \ldots, (x_{n+1}, y_{n+1})$,leading to n intervals and n quadratic polynomials. The form of each quadratic polynomial is:

$$S_i\,(x) = a_i x^2 + b_i x_i + c_{i....} i = 1, 2, ..., n \tag{3.3}$$

where $ai, bi, ci(i = 1, 2, \ldots, n)$ are unknown constants to be calculated. The total number of unknown constants is $3n$ because there are n such polynomials, each with three unknowns.

$$S_1\,(x_1) = y_1$$

$$S_n\,(x_{n+1}) = y_{n+1}$$

$$a_1 x_1^2 + b_1 x_1 + c_1 = y_1$$

$$a_n x_{n+1}^2 + b_n x_{n+1} + c_n = y_{n+1}$$

Therefore, exactly 3n equations the first polynomial $S_1(x)$ must go through (x_1, y_1) and the last polynomial $S_n(x)$ must go through (x_{n+1}, y_{n+1}):

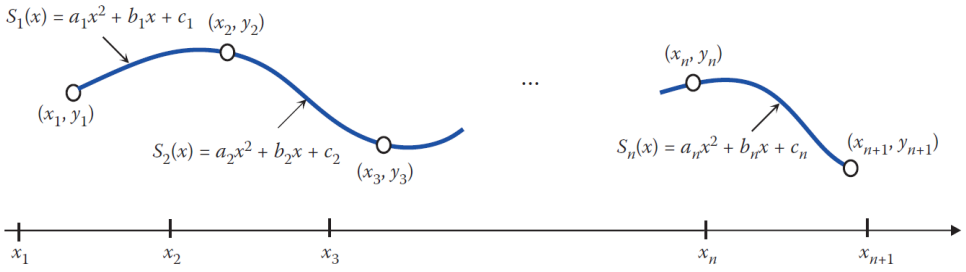


Figure 3.4: Quadratic spline equations

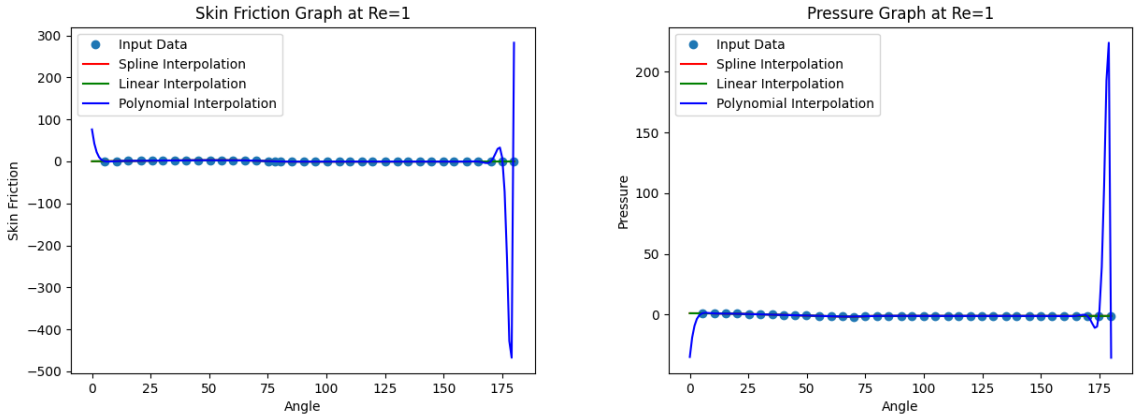
```python
def spline_interpolation(x, y):
    # Create an InterpolatedUnivariateSpline object
    sorted_idx = np.argsort(x)
    sorted_x = x[sorted_idx]
    sorted_y = y[sorted_idx]
    spline = InterpolatedUnivariateSpline(sorted_x, sorted_y)

    x_ = np.linspace(0, 180, 181)
    y_ = spline(np.array(x_))
    return x_, y_
```

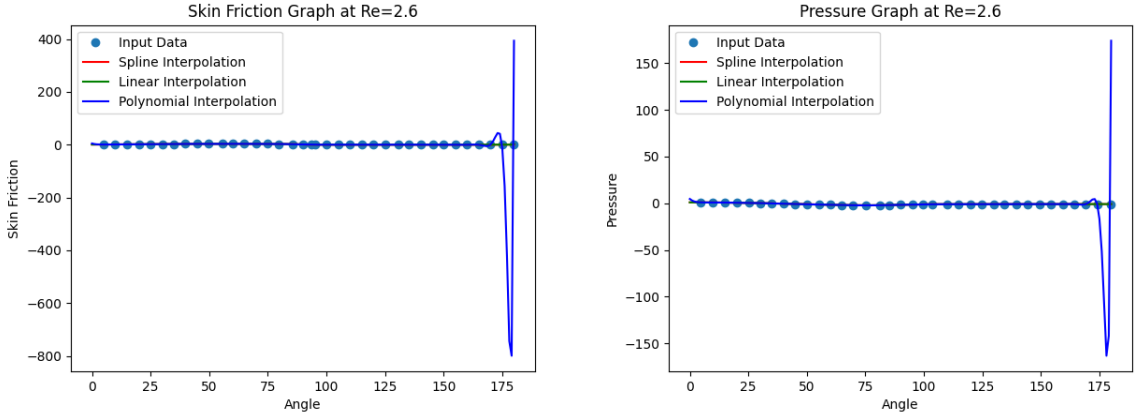
Figure 3.5: Python function for spline interpolation (see Appendix A)

3.3 Testing of Interpolation Methods

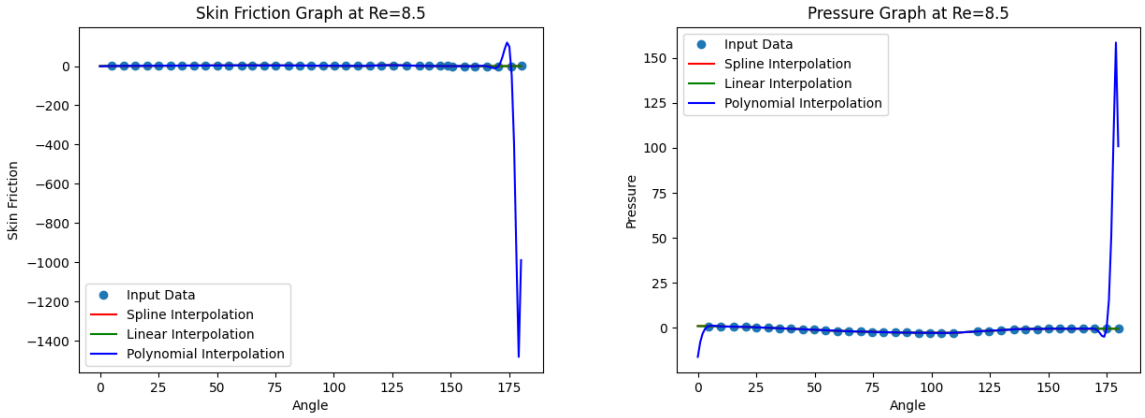
The three datasets for the different Reynold's numbers 10^5, 2.6×10^5 and 8.5×10^5 were interpolated using linear, polynomial and spline interpolation methods and their graphical behaviors were analyzed for skin friction and pressure respectively. Theoretically, of the three interpolation methods, polynomial methods are expected to erratically behave as depicted in Figure 3.6. The performance of the spline, linear and polynomial of which spline shows to be a best fit of all, linear shows a minor deviation and polynomial over-fit or outliers. The error of the interpolated value is composed of two parts, one part which is due to measurement error of the digitized document or graphs from $Achenbach$ results, and another part which is the error of the interpolation itself such as the polynomial interpolation.



(a) $Re = 10^5$



(b) $Re = 2.5 \times 10^5$



(c) $Re = 8.5 \times 10^5$

Figure 3.6: Spline(fitted), Linear(deviation) and Polynomial (oscillating) for $Re = 10^5, 2.6 \times 10^5, 8.5 \times 10^5$

3.3.1 Interpolation Error

After plotting the graphs of the three interpolated datasets, the performance of each method was observed where the spline method provides the best fitting, while the linear shows some deviation at the beginning of the curve and the polynomial method was oscillating as shown in Figure 3.6 (a), (b) and (c). In further investigation and studies of interpolation shows that polynomial interpolation can result in end values that appear to be outliers because of the nature of polynomial functions. Polynomial functions can oscillate wildly as they extend beyond the range of the data points used to fit the curve. This is known as the "Runge's phenomenon" and can lead to polynomial functions that exhibit large oscillations, particularly at the ends of the interval where they are being extrapolated [19]. The function must pass through each data point when fitting a polynomial function to a series of data points using polynomial interpolation. Nevertheless, a polynomial generated by such an approach can swing a great deal if its degree is too high or if the data points are not well-behaved. Employing a greater degree of polynomial or including additional data points close to the ends are two strategies for reducing the impact of oscillations at the endpoints of the interval, however, this may not always be possible or desirable [19]. Another approach is to use alternative interpolation techniques, such as spline interpolation, which are better suited for handling oscillatory data.

3.3.2 Interpolation Error Parameters

3.3.2.1 Root Mean Square Error (RMSE)

The RMSE is commonly used as a metric to evaluate the accuracy of interpolation methods because it provides a measure of how well the interpolation method approximates the actual values at the observed data points. In this thesis, the goal is to estimate the values of a function at points that are not observed based on the values of the function at a set of observed data points [22]. RMSE measures the difference between the $predicted$-values (yp_new_spline) from the interpolation method and the actual-values ($y\_resized$) at the observed data points in Figure 3.7. It provides a quantitative measure of how well the interpolation method fits the observed data points. When MAE and RMSE are compared,

it's easy to see which is more intuitive and provides a simple way to compare the performance of the three types of interpolation methods. Furthermore, RMSE can be used to optimize the parameters used in interpolation methods, such as polynomial fit or radial basis function size. RMSE for interpolation can be calculated by estimating the values at the observed points using the interpolation method. Based on these estimated values, you would calculate the difference between the actual values and the estimated values. Finally, you would calculate the square of these differences, average them, and take the square root to get the RMSE [23]. To calculate the root-mean-square error (RMS error), square the differences between known (actual-value) and unknown (predicted-value) points, sum these squares, divide by the total number of test points, and then square root the result

RMSE Equation:

$$RMSE = \sqrt{\frac{1}{n}\sum((predicted - actual)^2} \tag{3.4}$$

where:

$n-$ is the number of observed data points.

$predicted-$ is the estimated value from the interpolation method.

$actual-$ is the actual value at the observed data point.

3.3.2.2 Mean Average Error (MAE)

The MAE is also used to evaluate the accuracy of interpolation methods, and it has some advantages over Root Mean Square Error (RMSE) in certain situations [24]. Like RMSE, MAE measures the difference between the $predicted$-values (yp_new_spline) from the interpolation method and the $actual$-values $(y\_resized)$ at the observed data points Figure 3.7. However, unlike RMSE, MAE does not square the differences, which makes it less sensitive to outliers [22] . In other words, if there are a few data points that are far

from the other points, these outliers will have a greater impact on RMSE than on MAE. MAE is also easier to interpret than RMSE because it is in the same units as the data.

The formula for calculating MAE is:

$$MAE = (\frac{1}{n}) \times \sum(|predicted - actual|) \tag{3.5}$$

where:

$n-$ is the number of observed data points.

$predicted-$ is the estimated value from the interpolation method.

$actual-$ is the actual value at the observed data point.

The interpolation procedure is more accurate the lower the MAE. As with RMSE, it is essential to observe that MAE alone may not provide a complete picture of the interpolation method's performance and should be used in conjunction with other evaluation metrics. MAE provides an easily interpretable measure of the interpolation method's accuracy and is a suitable alternative to RMSE when working with datasets containing outliers.

3.3.3 Python Implementation of Interpolation Methods

This Python code implements interpolation functions for processing CSV-formatted input data and producing interpolated output data. The code contains three interpolation functions: spline, linear, and polynomial. The data in the CSV file are organized in two columns, with the first column representing the angle (Theta) values and the second column representing either pressure or skin friction. The code can be applied simultaneously to multiple "$Input$" folder input files; for each file, it calculates the interpolated values using the three interpolation functions and prints the RMSE and MAE values for each function. Importing the necessary libraries (os, $numpy$, $pandas$, and $InterpolatedUnivariateSpline$ from $scipy.interpolate$) is the first step of the code. Then, three functions are defined for the three interpolation methods. Upon receiving

two arrays, x and y, the spline_interpolation function returns the interpolated values for x and y. It utilizes the $InterpolatedUnivariateSpline$ function to interpolate the spline. The linear_interpolation function accepts two arrays,x and y, and returns the linearly interpolated values for x and y (Appendix A, Fig. A.1).

The polynomial_interpolation function accepts three arguments: the x and y arrays, as well as the degree of the polynomial to be fit. It returns the values interpolated using polynomial interpolation for x and y. The program then retrieves the input file paths from the "$Input$" folder that contains the dataset and stores them in a list (Appendix B, Fig. B.1—B.3). If there is no input file, the code prints a message and exits. The code scans each CSV file into a pandas DataFrame and stores the angle values in the x array and the data values in the values array for each file in the list. Using the three interpolation functions, it then calculates the interpolated values for x and values and stores them in three distinct arrays. The degree of the polynomial to be fitted by the polynomial_interpolation function is determined by the filename and data type (pressure or skin friction). Finally, the code outputs the RMSE and MAE values for each interpolation function using the calculated interpolated values for the pressure data. The RMSE is computed using $numpy.sqrt$ and $numpy.mean$, while the MAE is computed using $numpy.abs$ and $numpy.mean$. The RMSE and MAE values for each interpolation function and input file are displayed (Appendix A, Fig. A.1—A.5).

3.3.4 Improving The Polynomial Interpolation Methods

For the deviation in the linear method according to the graphs, it was found that the issue was in the python script while reading the input data from the files. Specifically, the code was skipping one row from each dataset. To resolve the issue, the algorithm was modified to include that first row of data. This fixed the deviation issue with the linear method and the graphs were plotted again as shown in Figure 3.8. After applying this fix, it was found that the interpolation techniques produced more accurate results and better represented the underlying data.

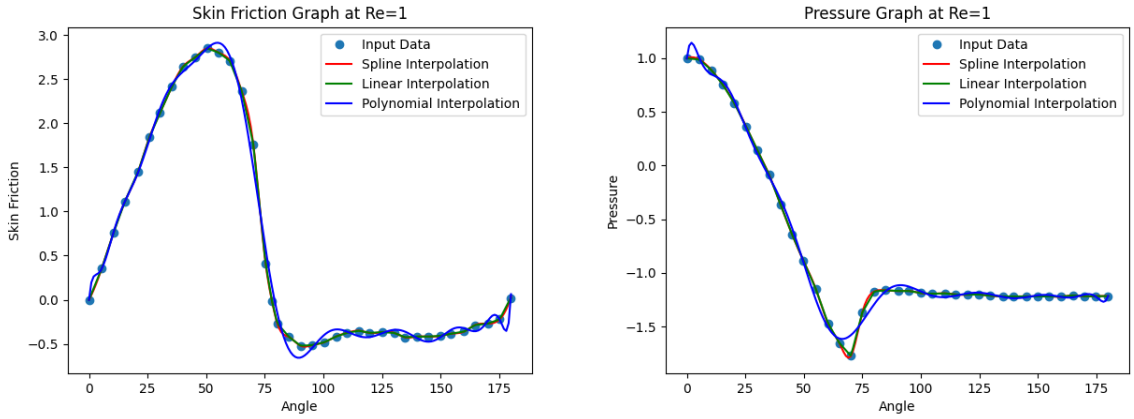
Table 3.1: RMSE and MAE Results for 2D-Spline, Linear and Polynomial Interpolation. Polynomial Degree for Pressure [$Re = 10^5$ (15 degree) $Re = 2.6 \times 10^5$ (19 degree) $Re = 8.5 \times 10^5$(19 degree). Polynomial Degree for Skin friction [$Re = 10^5$ (17 degree) $Re = 2.6 \times 10^5$ (13 degree), $Re = 8.5 \times 10^5$ (25 degree)

		Pressure			Skin Friction		
	Re	1×10^5	2.6×10^5	8.5×10^5	1×10^5	2.6×10^5	8.5×10^5
	Methods						
RMSE	Spline	1.018	1.189	1.613	1.018	1.856	1.974
	Linear	1.018	1.184	1.608	1.798	1.868	1.966
	Polynomial	1.022	1.185	1.615	1.816	1.862	1.975
MAE	Spline	0.683	0.879	1.267	1.377	1.418	1.579
	Linear	0.681	0.876	1.264	1.374	1.433	1.572
	Polynomial	0.685	0.878	1.274	1.378	1.449	1.589

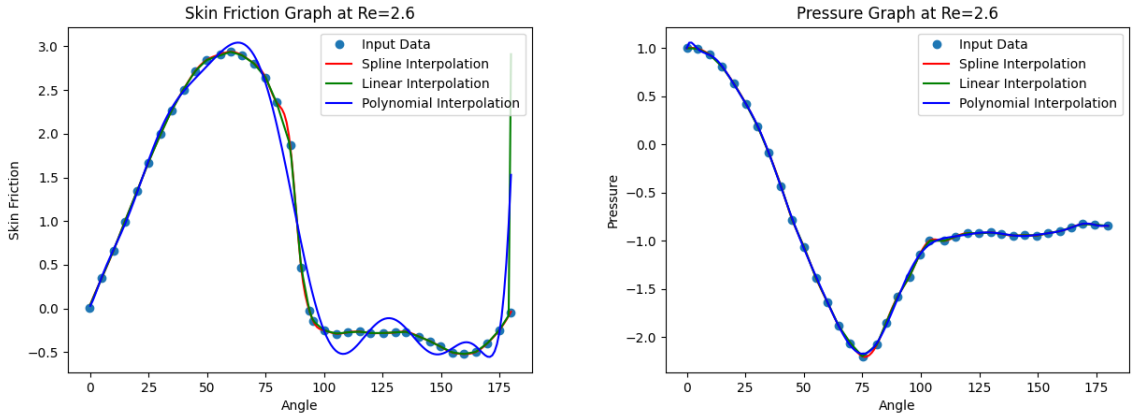
3.3.5 Result and Comparative Analysis

This part explains the result and comparison for the two-dimensional interpolation and visualization of the datasets with different interpolation methods. Three interpolation methods as introduced theoretically in the beginning of this chapter were tested on the datasets which are from the same source. This study intends to find the most appropriate or suitable interpolation method in terms of RMSE and MAE for these datasets Table 3.1 . In order to analyze and verify the results of accuracy assessment, the relative performance of three interpolation methods have been examined, both mathematical studies and visual comparisons was performed. The graphs produced by the three interpolation techniques—linear, polynomial, and spline—are shown in Figure 3.6 represent the first set of result analyses for $Re = 10^5$, $Re = 2.6 \times 10^5$ and $Re = 8.5 \times 10^5$ (skin friction and pressure) and in terms of RMSE and MAE in Table 3.1. According to the results of the error analyses in Figure 3.6, there is high variations in this instance especially for polynomial compared to Figure 3.8, the second set which has a low variations. In both instances it is clear that spline provides the best outcomes, while linear comes close behind and the worst outcomes were once more produced by polynomial. However, even though there is no much difference between linear and spline, spline consistently outper-

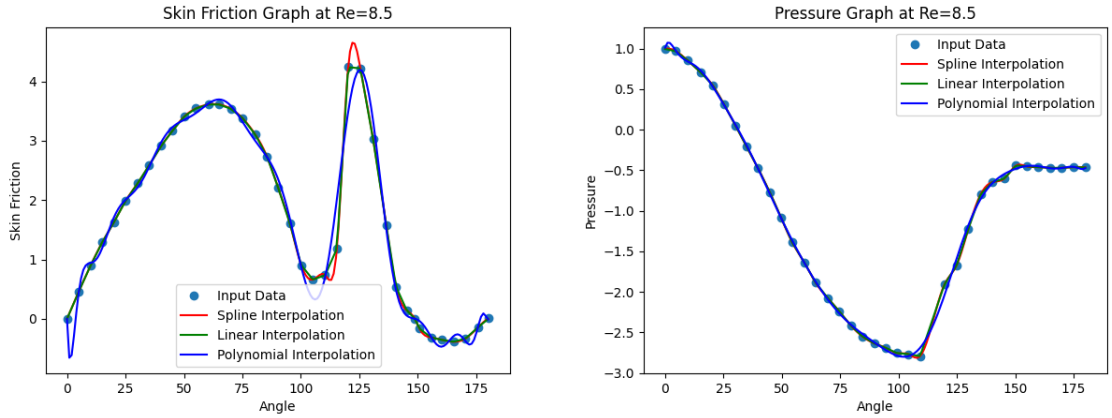
forms linear in terms of performance. Figure 3.7 and Table 3.1 shows the best-performing interpolation technique by comparing the RMSE and MAE and based on the result, for Pressure—Spline has the lowest MAE values of 0.683 $[0.683, 0.879, 1.267]$. In terms of RMSE, spline also has the lowest values of 1.018 $[1.018, 1.189, 1.613]$. Also for Skin friction, Spline has the lowest MAE values of 1.377 $[1.377, 1.418, 1.579]$ and RMSE values for 1.803 $[\ 1.803, 1.856, 1.974]$. In comparison to linear interpolation which shows MAE values of 0.681 $[0.681, 0.876, 1.264]$ and RMSE, 1.018 $[1.018, 1.184, 1.608]$ and for Skin friction shows MAE values of 1.374 $[1.374, 1.433, 1.572]$ and RMSE values for 1.798 $[1.798, 1.868, 1.966]$



(a) $Re = 10^5$



(b) $Re = 2.6 \times 10^5$



(c) $Re = 8.5 \times 10^5$

Figure 3.7: Spline(fitted), Linear(fitted) and Polynomial (fitted) for $Re = 10^5, 2.6 \times 10^5, 8.5 \times 10^5$. Polynomial Degree for Pressure [$Re = 10^5$ (15 degree) $Re = 2.6 \times 10^5$ (19 degree) $Re = 8.5 \times 10^5$(19 degree). Polynomial Degree for Skin friction [$Re = 10^5$ (17 degree) $Re = 2.6 \times 10^5$ (13 degree), $Re = 8.5 \times 10^5$ (25 degree)

Chapter 4

Surface Interpolation

Surface interpolation refers to the process of estimating values for points on a surface based on known values at discrete sample points. Using these sample points, we create a smooth continuous surface, allowing the estimation of values at other locations. The techniques for surface interpolation include polynomial interpolation, spline interpolation, nearest neighbor, and kriging. They vary in complexity, computational requirements, and accuracy. The data type determines the interpolation method, level of accuracy, and computational resources available, but for this thesis, nearest neighbor, spline, and kriging were used.

The reason for the surface interpolation is to make a surface that includes all three Reynolds Numbers (Re) and can be used to interpolate values of pressure and skin friction for values of Re which lie in between the datasets. By providing accurate estimates of pressure and skin friction values at any point on a surface, surface interpolation can aid in simulations, design optimizations, and in identifying regions of high pressure or skin friction gradients that may indicate flow separation or boundary layer transition.

4.1 3D-Interpolation Methods

4.1.1 Kriging Interpolation:

Kriging is a geostatistical interpolation technique used to gauge the worth of a variable at an unsampled area in view of the values of neighboring sampled locations [25]. This technique is often used in spatial analysis to create continuous surface models from discrete data points [26]. Kriging is a powerful tool that uses the spatial correlation of data to produce accurate estimates with quantified uncertainty. The basic principle of kriging is to model the spatial correlation of data using a mathematical function called a variogram [27]. The variogram measures how the variance of the data changes as the distance between points increases. By analyzing the variogram, kriging calculates a set of weights for each sampled location based on the distance to the unsampled area and the spatial connection of the information. At the unsampled area, these special kind of loads are utilized to measure the variable worth [27]. Kriging is a process that can be expressed mathematically as:

$$Z(S_0) = \sum_{i=1}^{N} \lambda_i Z(S_i) \tag{4.1}$$

where :

$Z(Si)-$ is the i-th position of the measured value;

$i-$ is the i-th position measurement values of the unknown weight;

S_0- is the predicted position;

$N-$ is the number of measurements.

Simple, Ordinary and Universal are three special kinds of kriging that are used. The data is known and constant is the theory that comes under simple kriging [28]. The mean estimation from the information given is known as ordinary kriging [28] [27]. The covariates are included in the universal kriging. These universal covariates include temperature elevation, interpolation accuracy etc. there are various flaws and benefits of each type of kriging as well. our choice is kriging is generally based on special characteristics of the under analysis data [27] [28].

One of the key advantages of kriging is that it produces estimates with quantified uncertainty. This is because kriging takes into account the spatial correlation of the data, which provides information about how reliable the interpolation is likely to be [29]. Nearest neighbor and spline interpolation, for example, don't provide you quite as much detail. However, kriging is not without its drawbacks. It may not work well with huge datasets or non-stationary data since it demands more processing resources than alternative interpolation methods. In addition, kriging's interpolation precision is susceptible to outliers, and it's all reliant on how well the variogram model works.

In conclusion, kriging is a powerful geostatistical interpolation technique that produces accurate estimates with quantified uncertainty. The interpolation's accuracy is improved because spatial correlations between data are included. While effective for some data sets, kriging typically calls for more processing power. The decision to utilize kriging, as with any analytical method, is contextual, based on the nature of the data and the desired outcomes of the study.

4.1.2 Spline Interpolation:

Spline surface interpolation is a method used to construct a smooth surface that passes through a set of given points in three-dimensional space [29]. It involves the use of piecewise polynomial functions, called splines, to approximate the surface. The process begins by selecting a set of control points that define the shape of the surface. The spline function is then constructed by fitting a polynomial curve to each section of the surface between adjacent control points. The degree of the polynomial used for each curve is typically chosen to be cubic or higher to ensure smoothness. The spline function is chosen such that it satisfies certain continuity conditions across the boundaries between adjacent polynomial curves. This ensures that the resulting surface is smooth and does not have any sharp discontinuities. Once the spline function has been constructed, it can be evaluated at any point in three-dimensional space to obtain an approximation of the surface. The general formula for spline surface interpolation can be expressed as follows:

Given a set of n sample points

$$\{(x_i, y_i, z_i)\} \tag{4.2}$$

where x_i , y_i , and z_i represent the x,y , and z coordinates , the goal is to estimate z at any arbitrary point(x, y) on the surface. Triangulating sample coordinates creates a mesh (Appendix A, Fig. A4). After that, the mesh is broken into smaller sub-triangles and a polynomial function is fitted to each using the sample points at its vertices. Most polynomials are bivariate quadratic or cubic spline. Then, the value of z at any surface point (x, y) can be estimated by locating the sub-triangle containing the point and evaluating the polynomial function for that sub-triangle at the point.

4.1.3 Nearest Neighbor Interpolation:

Nearest Neighbor Surface Interpolation (NNSI) is a method of surface interpolation used to estimate the value of a function at unsampled points based on the values of the function at nearby sampled points [30] . NNSI assumes that the function being interpolated is continuous and that nearby points on the surface of the function have similar values [28]. NNSI is particularly useful when the sampled points are irregularly spaced or when there is a high density of data points. It is a simple and fast method that can quickly approximate the surface of a function. To estimate the value of the function at an unsampled point, NNSI finds the nearest sampled point to that unsampled point. The value of the function at the unsampled point is then estimated as the value of the function at the nearest sampled point [30].

However, It does not create a smooth surface and produces findings that are sensitive to the sampled point distribution. NNSI also ignores spatial relationships between sampling sites, which can be essential. NNSI is useful in many situations that require simple and quick surface interpolation despite these constraints. It can enhance other surface interpolation algorithms.

From the observations from the other two methods above, the NNSI was found to be particularly useful because the method is simple and fast, NNSI works by assuming that

the function being interpolated is continuous and that nearby points on the surface of the function have similar values. When estimating the value of the function at an unsampled point, NNSI finds the nearest sampled point to that unsampled point and estimates the value of the function at that point.

Table 4.1: RMSE and MAE Results for 3D-Nearest, Spline and Kriging Interpolation

		Pressure			Skin Friction		
	Re	1×10^5	2.6×10^5	8.5×10^5	1×10^5	2.6×10^5	8.5×10^5
	Methods						
RMSE	Nearest	1.018	1.189	1.613	1.803	1.856	1.974
	Spline	1.018	1.184	1.608	1.798	1.868	1.966
	Kriging	1.022	1.185	1.615	1.810	1.862	1.975
MAE	Nearest	0.683	0.879	1.267	1.377	1.418	1.579
	Spline	0.681	0.876	1.264	1.374	1.433	1.572
	Kriging	0.685	0.878	1.274	1.378	1.449	1.589

4.1.4 Griddata—Python Implementation of Surface Interpolation Methods

This Python code incorporates commonly used libraries for scientific computing and data visualization, such as $math$, os, $numpy$, $pandas$, and $matplotlib.pyplot$. Additionally, $pykrige$ and $scipy.interpolate$ are imported for interpolation purposes (see Appendix B, Fig. B.1). This code is designed to interpolate pressure or skin friction data at various angles and Reynolds numbers. The code requests two user inputs using the input function. Choose between pressure and skin friction data as the first input. The second input consists of choosing a Reynolds number from the three options provided. The program then locates the file in the Input directory that corresponds to the selected inputs. It filters the list of files according to the specified Reynolds number and assigns the first file to the variable "$file$." Additionally, it extracts this file into a pandas dataframe and stores the angle, pressure, or skin friction data in numpy arrays (see Appendix B, Fig. B.1).

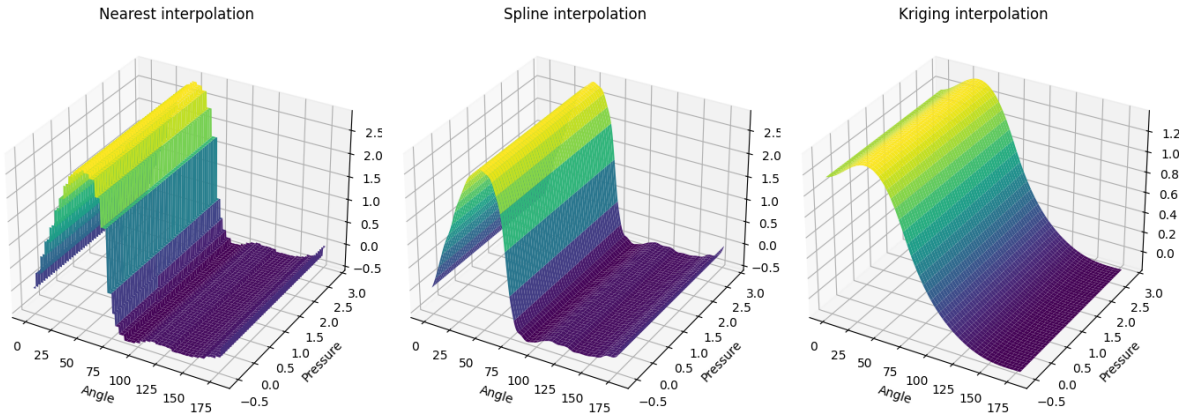
The code subsequently specifies three interpolation techniques: nearest, spline, and kriging. It generates a mesh grid of pressure and angle values and interpolates using the specified methodologies. Utilizing the $matplotlib$ library, the interpolated values are plotted on a 3D surface. For interpolation using kriging, the $pykrige$ library is utilized. The $interp2d$ function from the $scipy.interpolate$ library is used for spline interpolation. The griddata function from the same library is used for additional interpolation methods. The code computes and stores in $rmse\_dict$ and mae_dict, respectively (see Appendix B, Fig. B.1— B.4), the root mean squared error (RMSE) and the mean absolute error (MAE) for each interpolation method. The results and 3D surface plots are printed at the conclusion of the code. The surface plots display interpolated data for pressure or skin friction at various angles and Reynolds numbers. The RMSE and MAE values indicate the precision of each method's interpolated data.

Overall, this code is beneficial for interpolating pressure or skin friction data at varying Reynolds numbers and angles. Additionally, the code is adaptable and permits simple customization of interpolation methods and plotting options. The RMSE and MAE calculations provide a quantitative measure of the interpolation methods' accuracy, making them valuable for validation.

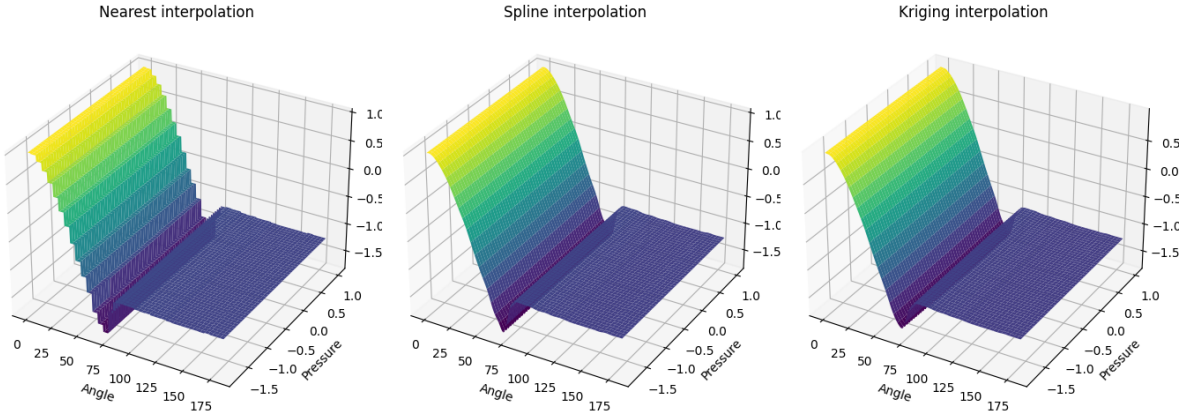
4.2 Results and Comparative Analysis

This section explains the result and comparison for the three-dimensional interpolation and visualization of the datasets with different interpolation methods: spline, nearest neighbor, and kriging, in the context of surface interpolation. These methods were applied to these datasets for $Re = 10^5$, $Re = 2.6 \times 10^5$ and $Re = 8.5 \times 10^5$ (skin friction and pressure) and estimate the RMSE and MAE with missing values in order to predict the values at these locations and create a continuous surface. After conducting a comparative analysis of the results in Table 4.1, it was found that for the kriging shows the lowest error of 0.685 in terms of MAE for pressure $(0.685, 0.878, 1.274)$ and 1.378 $[1.378, 1.449, 1.589]$ for skin friction. The same is true for RMSE Table 4.1. Kriging outperformed the other two methods in terms of accuracy and provided the most realistic

and continuous surface Figure 4.1—4.3. The results of these methods highlight the importance of selecting an appropriate interpolation method when working with spatial data. While spline and nearest-neighbor methods are commonly used, they may not always be the most suitable choice for surface interpolation. Kriging, with its ability to incorporate spatial correlation and variability into the interpolation process, can be a powerful tool for accurately predicting values and creating continuous surfaces.



(a) Skin Friction

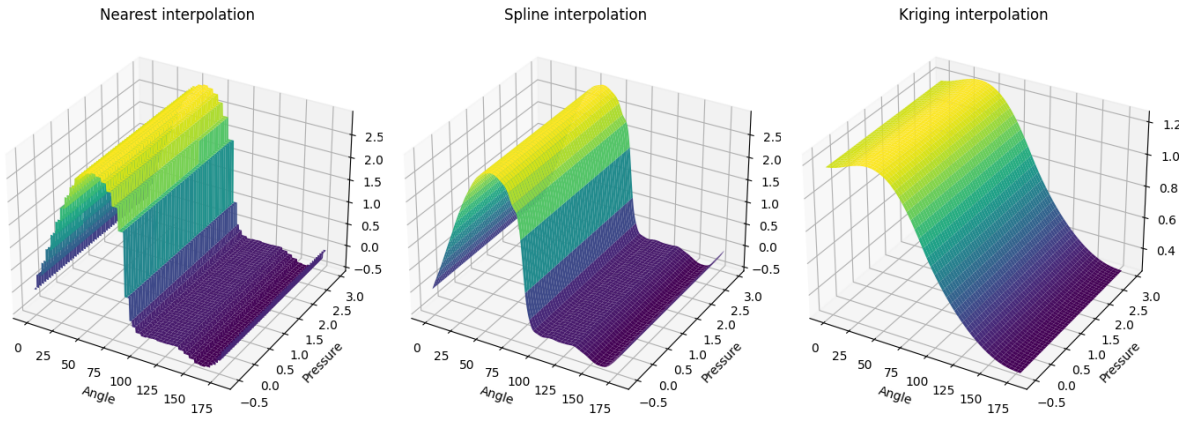


(b) Pressure

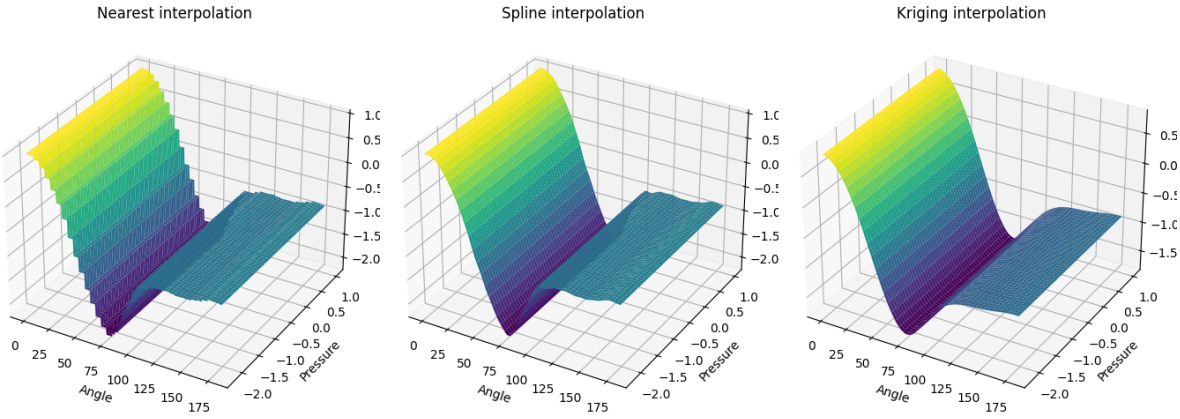
Figure 4.1: Griddata Results For Nearest, Spline and Kriging for $Re = 10^5$

This study also demonstrated the significance of considering the spatial characteristics of the data when selecting an interpolation technique. By taking into consideration the underlying spatial variability and correlation, kriging was able to generate a more accurate and continuous surface, which is crucial for many spatial applications. This paper shows how kriging can be used for surface interpolation and provides vital insights into interpolation algorithms. This work hopes to expand the use of kriging in surface interpolation

and improve spatial forecasts and decision-making across disciplines.



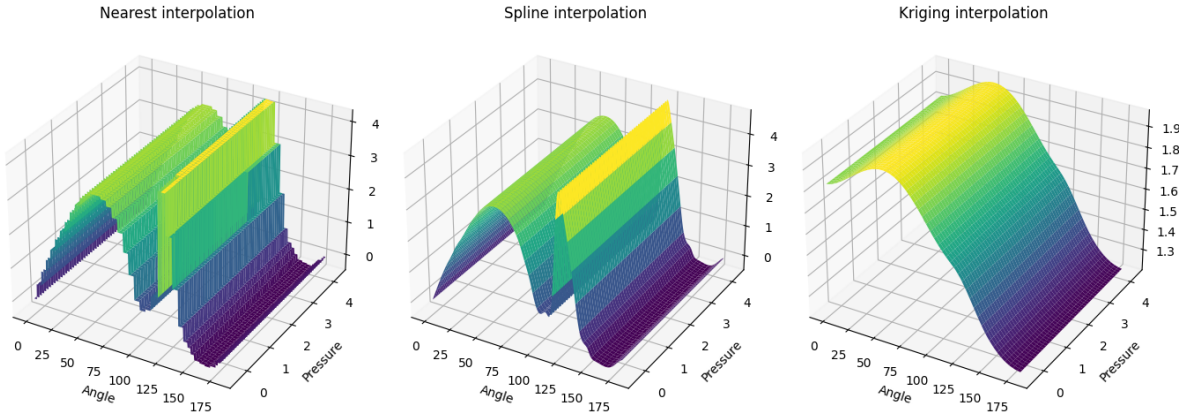
(a) Skin Friction



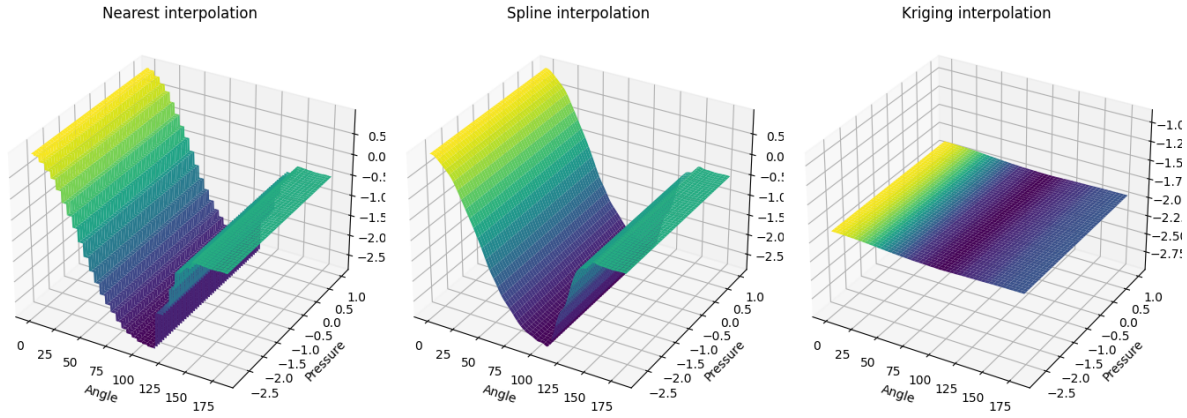
(b) Pressure

Figure 4.2: Griddata Results For Nearest, Spline and Kriging for $Re = 2.6 \times 10^5$

In each instance, it is evident that kriging yields the best results, followed by nearest neighbor and spline, which produced the worst results. Despite the fact that there is little difference between kriging and nearest neighbor, kriging consistently outperforms the nearest neighbor. In addition, error analyses were conducted Table 4.1, and according to the mean absolute error (MAE), the interpolation technique that performed the best was kriging, while spline performed the worst compare to root mean square error (RMSE).



(a) Skin Friction



(b) Pressure

Figure 4.3: Griddata Results For Nearest, Spline and Kriging for $Re = 8.5 \times 10^5$

4.3 Summary

The process of transforming physical data into digital representations that can be quickly evaluated, displayed, and changed using software tools is known as data extraction and digitization. Dealing with incomplete or incorrect data points, which might happen because of things like low data quality, human mistakes, or the constraints of the digitization process itself, can make this procedure difficult. In order to approximate missing data points or values inside a given dataset, interpolation techniques are frequently employed in data processing and analysis. The practice of guessing a function's value at a point within a specified range using the values of nearby points is known as interpolation.

This thesis investigated the most appropriate or suitable interpolation method for the datasets extracted and digitized (Figure 2.5) from E.A Achenbach research [16] which

formed the basis of the entire research. In order to analyze and verify the results of the accuracy assessment, the relative performance of three interpolation methods has been examined, and both mathematical studies and visual comparisons were performed. The study further demonstrated the importance of considering the spatial characteristics of the data when selecting an interpolation technique. The spline interpolation method was found to be the best fit for the two-dimensional interpolation, while for the three- dimension, the kriging interpolation method was able to generate a more accurate and continuous surface by taking into account the underlying spatial variability and correlation. These methods have been widely studied and compared in terms of their performance and both can provide accurate and reliable results, but their performance depends on the characteristics and data distribution. Overall, spline and kriging methods have been found to have the best performance for spatial interpolation for the datasets used in this research.

Conclusions

Three interpolation methods as introduced theoretically in chapter three and four respectfully, were applied to the extracted and digitized datasets from the Achenbach experiment Figure 2.5 which is half of the entire data points from the Achenbach experiment Figure 2.2 – 2.4. The datasets contains 50% of the data points from each figure, which means extraction occur between 0 and 180 degrees angle for $Re = 10^5$, $Re = 2.6 \times 10^5$ and $Re = 8.5 \times 10^5$ (skin friction and pressure) and a total of 180 sample data points (0 to 180) was used, for example (see Appendix C, Datasets, C.1—C.3). Referring to Figure 3.8 which shows the generated graphs by the three interpolation methods in two-dimension, presents the visibility comparisons on the datasets for the three interpolation methods.

This study intends to find the most appropriate or suitable interpolation method for these datasets. In order to analyze and verify the results of accuracy assessment, the relative performance of three interpolation methods have been examined, both mathematical studies and visual comparisons was performed. In Chapter three, the graphs produced by the three interpolation techniques—Spline, Linear, and Polynomial—are shown in Figure 3.8, represent the first set of result analyses for $Re = 10^5$, $Re = 2.6 \times 10^5$ and $Re = 8.5 \times 10^5$ (skin friction and pressure) in two-dimension. In all the instances, it is clear that spline provides the best outcomes, while linear comes close behind and the worst outcomes were once more produced by polynomial. Additionally, some error analyses were carried out (Figure 3.7 and Table 3.1) and the best-performing interpolation technique by comparing the RMSE and MAE and based on the result, for pressure, spline has the lowest MAE values of $0.683, 0.879, 1.267$ respectively. In terms of RMSE, spline also has the lowest values of $1.018, 1.189, 1.613$, respectively. Also for Skin friction, spline has the lowest MAE values of $1.377, 1.418, 1.579$ and RMSE values for $1.803, 1.856, 1.974$ respectfully.

In Chapter four, the performance of three different interpolation methods: Spline, Nearest Neighbor, and Kriging in the context of surface interpolation in three-dimension was investigated. These methods were applied to the three-dimensional datasets (see Appendix C, C.4—C.5) in order to predict the values at these locations and create a continuous surface Figure 41—4.3. After conducting a comparative analysis of the results (Table 4.1 and Figure 41—4.3), for $Re = 10^5$, $Re = 2.6 \times 10^5$ and $Re = 8.5 \times 10^5$ (skin friction and pressure), it was found that kriging outperformed the other two methods in terms of accuracy and precision and provided the most realistic and continuous surface. Based on the result, for pressure, kriging performs best for both RMSE and MAE for all three Reynolds Numbers, it has the lowest MAE error of 0.685 $(0.685, 0.878, 1.274)$ and 1.378 for skin friction $(1.378, 1.449, 1.589)$. In terms of RMSE, kriging has 1.022 which is the lowest values of $(1.022, 1.185, 1.615)$ for pressure and for skin friction has the lowest value of 1.810 $(1.810, 1.862, 1.975)$ Table 4.1.

Finally, this investigation showed the need of considering data spatial properties while choosing an interpolation method. Spline interpolation is more flexible and accurate, especially for irregular data points. Spline interpolation also creates a smooth curve that better represents the function. For many spatial applications, kriging's allowance for spatial variability and correlation created a more realistic and continuous surface. This study sheds light on surface interpolation methods and shows the efficacy of kriging and spline. This work hopes to increase the use of these interpolation approaches to improve spatial predictions and decision-making.

Future Work

The study is hindered by the fact that it only looked at a small number of datasets. It would have been The study's limited datasets impede it. A large collection would have made results more universal. . The study also didn't look into the parameter space of each approximation method as much as it could have. It would be helpful to look at a bigger range of parameter values and assumptions to learn more about how each method works. The metrics used are another drawback. More metrics, like the mean absolute percentage error (MAPE) or the coefficient of determination (R-squared), could be used to give a complete picture of how well each approximation method works.

The first suggestion for future work is to investigate the impact of different parameters or assumptions used in each interpolation method: Many interpolation methods have various parameters and assumptions that can affect their performance. Polynomial interpolation accuracy depends on the degree of the polynomial, the range of the kriging neighborhood, and the kind of spline basis function. Each interpolation approach has downsides. Therefore, future research might create new interpolation methods or alter existing methods to better meet specific issues or restrictions, such as non-stationarity or big datasets.

Second, using interpolation methods with regression or machine learning to construct hybrid models that combine their strengths: Hybrid models can combine interpolation methods with regression or machine learning to take use of their capabilities. Hybrid models frequently yield more accurate and robust findings than single methods.

Bibliography

[1] ICOLD, "World register of damsgeneral synthesis." [Online]. Available: http://www.icold-cigb.net/GB/Worldregister/general_synthesis.asp

[2] D. A. Algera, T. Rytwinski, J. J. Taylor, J. R. Bennett, K. E. Smokorowski, P. M. Harrison, K. D. Clarke, E. C. Enders, M. Power, M. S. Bevelhimer *et al.*, "What are the relative risks of mortality and injury for fish during downstream passage at hydroelectric dams in temperate regions? a systematic review," *Environmental Evidence*, vol. 9, no. 1, pp. 1–36, 2020.

[3] T. Kober, H.-W. Schiffer, M. Densing, and E. Panos, "Global energy perspectives to 2060–wec's world energy scenarios 2019," *Energy Strategy Reviews*, vol. 31, p. 100523, 2020.

[4] J. Radinger, R. van Treeck, and C. Wolter, "Evident but context-dependent mortality of fish passing hydroelectric turbines," *Conservation Biology*, p. e13870, 2022.

[5] I. S. Pauwels, R. Baeyens, G. Toming, M. Schneider, D. Buysse, J. Coeck, and J. A. Tuhtan, "Multi-species assessment of injury, mortality, and physical conditions during downstream passage through a large archimedes hydrodynamic screw (albert canal, belgium)," *Sustainability*, vol. 12, no. 20, p. 8722, 2020.

[6] C. Bunt, T. Castro-Santos, and A. Haro, "Performance of fish passage structures at upstream barriers to migration," *River Research and Applications*, vol. 28, no. 4, pp. 457–478, 2012.

[7] O. Calles, S. Karlsson, M. Hebrand, and C. Comoglio, "Evaluating technical improvements for downstream migrating diadromous fish at a hydroelectric plant," *Ecological Engineering*, vol. 48, pp. 30–37, 2012.

[8] D. Buysse, A. Mouton, R. Baeyens, and J. Coeck, "Evaluation of downstream migration mitigation actions for eel at an archimedes screw pump pumping station," *Fisheries management and ecology*, vol. 22, no. 4, pp. 286–294, 2015.

[9] G. F. Čada, "The development of advanced hydroelectric turbines to improve fish passage survival," *Fisheries*, vol. 26, no. 9, pp. 14–23, 2001.

[10] S. V. Amaral, B. S. Coleman, J. L. Rackovan, K. Withers, and B. Mater, "Survival of fish passing downstream at a small hydropower facility," *Marine and Freshwater Research*, vol. 69, no. 12, pp. 1870–1881, 2018.

[11] J. Knott, M. Mueller, J. Pander, and J. Geist, "Fish passage and injury risk at a surface bypass of a small-scale hydropower plant," *Sustainability*, vol. 11, no. 21, p. 6037, 2019.

[12] D. C. Alves, L. P. Vasconcelos, L. F. da Câmara, L. Hahn, and A. A. Agostinho, "Protocol for the assessment of mortality and injuries in fish larvae associated with their downstream passage through hydropower dams," *Reviews in Fish Biology and Fisheries*, vol. 29, no. 2, pp. 501–512, 2019.

[13] T. Rytwinski, D. A. Algera, J. J. Taylor, K. E. Smokorowski, J. R. Bennett, P. M. Harrison, and S. J. Cooke, "What are the consequences of fish entrainment and impingement associated with hydroelectric dams on fish productivity? a systematic review protocol," *Environmental Evidence*, vol. 6, no. 1, pp. 1–9, 2017.

[14] M. Mueller, J. Knott, J. Pander, and J. Geist, "Experimental comparison of fish mortality and injuries at innovative and conventional small hydropower plants," 2022.

[15] B. D. Pflugrath, R. Saylor, K. M. Engbrecht, R. P. Mueller, J. R. Stephenson, M. Bevelhimer, B. M. Pracheil, and A. H. Colotelo, "Biological response models: Predicting injury and mortality of fish during downstream passage through

hydropower facilities," Pacific Northwest National Lab.(PNNL), Richland, WA (United States), Tech. Rep., 2021.

[16] E. Achenbach, "Distribution of local pressure and skin friction around a circular cylinder in cross-flow up to re= 5× 106," *Journal of Fluid Mechanics*, vol. 34, no. 4, pp. 625–639, 1968.

[17] A. Rohatgi, "Webplotdigitizer: Version 4.6," 2022. [Online]. Available: https://automeris.io/WebPlotDigitizer

[18] Q. Kong, T. Siauw, and A. Bayen, *Python programming and numerical methods: A guide for engineers and scientists*. Academic Press, 2020.

[19] J. Kiusalaas, *Numerical methods in engineering with Python 3*. Cambridge university press, 2013.

[20] R. L. Burden, J. D. Faires, and A. M. Burden, *Numerical analysis*, 10th ed. Cengage learning, 2015.

[21] M. K. Autar Kaw, "Holistic numerical methods," p. chapt. 5, 2009. [Online]. Available: https://nm.mathforcollege.com/chapter-05-05-spline-method/

[22] Y. Chen, X. Shan, X. Jin, T. Yang, F. Dai, and D. Yang, "A comparative study of spatial interpolation methods for determining fishery resources density in the yellow sea," *Acta Oceanologica Sinica*, vol. 35, pp. 65–72, 2016.

[23] Y. Sun, S. Kang, F. Li, and L. Zhang, "Comparison of interpolation methods for depth to groundwater and its temporal and spatial variations in the minqin oasis of northwest china," *Environmental Modelling & Software*, vol. 24, no. 10, pp. 1163–1170, 2009.

[24] M. Mardikis, D. Kalivas, and V. Kollias, "Comparison of interpolation methods for the prediction of reference evapotranspirationan application in greece," *Water Resources Management*, vol. 19, pp. 251–278, 2005.

[25] R. M. Pokhrel, J. Kuwano, and S. Tachibana, "A kriging method of interpolation used to map liquefaction potential over alluvial ground," *Engineering*

Geology, vol. 152, no. 1, pp. 26–37, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0013795212002888

[26] M. A. Oliver and R. Webster, "Kriging: a method of interpolation for geographical information systems," *International Journal of Geographical Information System*, vol. 4, no. 3, pp. 313–332, 1990.

[27] Q. Tan and X. Xu, "Comparative analysis of spatial interpolation methods: an experimental study," *Sensors & Transducers*, vol. 165, no. 2, p. 155, 2014.

[28] H. M. Yilmaz, "The effect of interpolation methods in surface definition: an experimental study," *Earth Surface Processes and Landforms*, vol. 32, no. 9, pp. 1346–1361, 2007. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/esp.1473

[29] B. Ajvazi and K. Czimber, "A comparative analysis of different dem interpolation methods in gis: case study of rahovec, kosovo," *Geodesy and cartography*, vol. 45, no. 1, pp. 43–48, 2019.

[30] P. Arun, "A comparative analysis of different dem interpolation methods," *The Egyptian Journal of Remote Sensing and Space Science*, vol. 16, no. 2, pp. 133–139, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110982313000276

Appendix A

Non-exclusive licence for reproduction and publication of a graduation thesis

I Abiola Taofeek Rashidi

(1). Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "Investigation of Interpolation Methods for Local Pressure and Skin Friction Around A Circular Cylinder", supervised by Associate Professor, Jeffrey Andrew Tuhtan

1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;

1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.

(2). I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.

(3). I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

Appendix B

Python code for 2D-Interpolation

```python
import os
import numpy as np
from scipy.interpolate import InterpolatedUnivariateSpline
import pandas as pd


# function to calculate interpolated values
def spline_interpolation(x, y):
    # Create an InterpolatedUnivariateSpline object
    sorted_idx = np.argsort(x)
    sorted_x = x[sorted_idx]
    sorted_y = y[sorted_idx]
    spline = InterpolatedUnivariateSpline(sorted_x, sorted_y)

    x_ = np.linspace(0, 180, 181)
    y_ = spline(np.array(x_))
    return x_, y_


def linear_interpolation(x, y):
    """
    Linear interpolation function.
    x: original x values (list or numpy array)
    y: original y values (list or numpy array)
    """
    # Find the index of the x value just smaller than x_new
    x_ = np.linspace(0, 180, 181)
    y_ = np.array([])
    for x_new in x_:
        i = np.searchsorted(x, x_new) - 1
```

Figure B.1: Code for linear interpolation function

```
        # Check if x_new is out of range
        if i == len(x) - 1:
            y_ = np.append(y_, [y[i]])
        elif i == -1:
            y_ = np.append(y_, [y[0]])
        # Perform linear interpolation
        else:
            y_ = np.append(y_, [y[i] + (y[i + 1] - y[i]) *
                                ((x_new - x[i]) / (x[i + 1] - x[i]))])
    return x_, y_

def polynomial_interpolation(x, y,degree):
    """
    Polynomial interpolation function.
    x: original x values (list or numpy array)
    y: original y values (list or numpy array)
    degree: degree of the polynomial (int)
    """
    # Fit a polynomial to the data
    y= np.append(y, y[-1])
    x= np.append(x, 180)
    coefficients = np.polyfit(x, y, degree)
    # Create a polynomial function
    polynomial = np.poly1d(coefficients)
    x_ = np.linspace(0, 180, 181)
    y_ = np.array([])
    for x_new in x_:
        # Calculate y_new
        y_new = polynomial(x_new)
        y_ = np.append(y_, [y_new])
```

Figure B.2: Polynomial interpolation function ..(continuation from A.1)

```
62
63
64  # get file list from input folder
65  file_list = [os.path.join(root, file) for root, dirs, files in
66               os.walk('Input') for file in files if
67               file.endswith('.csv') and '~$' not in file.lower()]
68  if not file_list:
69      print('No input file found')
70      exit(1)  # Exit if no input file found
71
72  # iterate on each file and calulate its interpolated values
73  for file in file_list:
74      print('Processing : {}'.format(file))
75      if 'pressure' in os.path.basename(file).lower():
76          data_type = 'Pressure'
77          degree=15 if 'Re=1' in file else 19
78      else:
79          data_type = 'Skin Friction'
80          if 'Re=1' in file:
81              degree=17
82          elif 'Re=2.6' in file:
83              degree=13
84          elif 'Re=8.5' in file:
85              degree=25
86          else:
87              degree=14
88
89
90      df = pd.read_csv(file, names=['Angle', data_type]) # read csv file
```

Figure B.3: Code to get files from input folder ..(continuation from A.2)

```
91
92      x = np.array(df['Angle'])  # store angle values in a variable
93      values = np.array(df[data_type])
94
95      # pressure values interpolation
96      # find interpolated values
97      xp_new_spline, yp_new_spline = spline_interpolation(x, values)
98      # find interpolated values
99      xp_new_linear, yp_new_linear = linear_interpolation(x, values)
100     # find interpolated values
101     xp_new_polynomial, yp_new_polynomial = polynomial_interpolation(x, values,degree)
102
103     # calculate RMSE Pressure
104     y_resized = np.resize(values, yp_new_spline.shape)
105     rmse = np.sqrt(np.mean((yp_new_spline - y_resized) ** 2))
106     print('RMSE(Pressure) for {} file using spline interpolation is : {:.3f}'
107                                             .format(file, rmse))  # print RMSE
108
109     y_resized = np.resize(values, yp_new_linear.shape)
110     rmse = np.sqrt(np.mean((yp_new_linear - y_resized) ** 2))
111     print('RMSE(Pressure) for {} file using linear interpolation is : {:.3f}'
112                                             .format(file, rmse))  # print RMSE
113
114     y_resized = np.resize(values, yp_new_polynomial.shape)
115     rmse = np.sqrt(np.mean((yp_new_polynomial - y_resized) ** 2))
116     print('RMSE(Pressure) for {} file using polynomial interpolation is : {:.3f}'
117                                               .format(file, rmse))  # print RMSE
118
119
```

Figure B.4: Code for RMSE Calculation....(continuation from A.3)

```
118
119     # calculate MAE Pressure
120     y_resized = np.resize(values, yp_new_spline.shape)
121     MAE = np.abs(yp_new_spline - y_resized).mean()
122     print('MAE(Pressure) for {} file using spline interpolation is : {:.3f}'
123                                             .format(file, MAE))  # print MAE
124
125     y_resized = np.resize(values, yp_new_linear.shape)
126     MAE = np.abs(yp_new_linear - y_resized).mean()
127     print('MAE(Pressure) for {} file using linear interpolation is : {:.3f}'
128                                              .format(file, MAE))  # print MAE
129
130     y_resized = np.resize(values, yp_new_polynomial.shape)
131     MAE = np.abs(yp_new_polynomial - y_resized).mean()
132     print('MAE(Pressure) for {} file using polynomial interpolation is : {:.3f}'
133                                                 .format(file, MAE))  # print MAE
134
135     import matplotlib.pyplot as plt
136
137     # plt.subplot(2, 1, 1)  # define a sub plot
138     plt.plot(x, values, 'o', label='Input Data')
139     plt.plot(xp_new_spline, yp_new_spline, 'r', label='Spline Interpolation')
140     plt.plot(xp_new_linear, yp_new_linear, 'g', label='Linear Interpolation')
141     plt.plot(xp_new_polynomial, yp_new_polynomial, 'b', label='Polynomial Interpolation')
142
143     plt.xlabel('Angle')
144     plt.ylabel(data_type)
145     plt.title('{} Graph at {}'.format(data_type,os.path.basename(os.path.dirname(file))))
146     plt.legend()
147     plt.show()
```

Figure B.5: Code for MAE Calculation...(continuation from A.3)

Appendix C

Python code for 3D-Interpolation

```
import math
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from pykrige import OrdinaryKriging
from scipy.interpolate import griddata, interp2d

# take input1
data_path = 'Input'
while True:
    option = input('Select \n1-Pressure\n2-Skin Friction\n')
    if option == '1':
        files_list = [os.path.join(root, file) for root, dirs, files
                      in os.walk(data_path) for file in files if
                      'pressure' in file.lower() and '~$' not in
                       file.lower() and file.endswith('.csv')]
        break
    elif option == '2':
        files_list = [os.path.join(root, file) for root, dirs, files
                      in os.walk(data_path) for file in files if
                      'skin' in file.lower() and '~$' not in file.lower()]
        break
    else:
        print('Invalid Input !', end=' ')

```

Figure C.1: Code for griddata surface interpolation

```
26
27  # take input 2
28  while True:
29      option = input('\nSelect\n1-Re=8.5*10^5\n2-Re=2.6*10^5\n3-Re=10^5\n')
30      if option == '1':
31          Re = 'Re=1'
32          files_list = [file for file in files_list if os.path
33                        .basename(os.path.dirname(file)) == Re]
34          break
35      elif option == '2':
36          Re = 'Re=2.6'
37          files_list = [file for file in files_list if os.path
38                        .basename(os.path.dirname(file)) == Re]
39          break
40      elif option == '3':
41          Re = 'Re=8.5'
42          files_list = [file for file in files_list if os.path
43                        .basename(os.path.dirname(file)) == Re]
44          break
45      else:
46          print('Invalid input!', end=' ')
47
48  # processing
49  file = files_list[0]
50  if 'pressure' in os.path.basename(file).lower():
51      data_type = 'Pressure'
52  else:
53      data_type = 'Skin Friction'
54  df = pd.read_csv(file, names=['Angle', data_type])
55  angle = np.array(df['Angle'].tolist())
```

Figure C.2: Code for griddata surface interpolation.....continuation from Fig. A.1

```
value = np.array(df[data_type].tolist())

# Define the interpolation methods
interpolations = ['nearest',  'spline', 'kriging']

# Create a meshgrid of angle and pressure values
theta_mesh, p_mesh = np.meshgrid(np.linspace(angle.min(),
angle.max(), 500),np.linspace(value.min(), value.max(), 500))

# Interpolate using different methods
fig, axs = plt.subplots(1, 3, figsize=(15, 10),
                        subplot_kw={'projection': '3d'})
rmse_dict = {}
mae_dict = {}
for i, interp in enumerate(interpolations):
    if interp == 'kriging':
        # Create a kriging object and fit the data
        OK = OrdinaryKriging(angle, value, value, variogram_model='linear')
        # Interpolate the pressure values using kriging
        p_interp, _ = OK.execute('grid', np.linspace(angle.min(),
            angle.max(), 500), np.linspace(value.min(), value.max(), 500))
    elif interp == 'polynomial':
        # Interpolate the pressure values using polynomial interpolation
        # Interpolate the pressure values using polynomial interpolation
        coeffs = np.polyfit(angle, value, 3)
        poly = np.poly1d(coeffs)
        p_interp = poly(np.linspace(angle.min(), angle.max(), 500))
    elif interp == 'spline':
        # Interpolate the pressure values using spline interpolation
        f = interp2d(angle, value, value, kind='cubic')
```

Figure C.3: Code for griddata surface interpolation.....continuation from Fig. B.2

```
        p_interp = f(np.linspace(angle.min(), angle.max(), 500),
                     np.linspace(value.min(), value.max(), 500))
    else:
        # Interpolate the pressure values using the given method
        p_interp = griddata((angle, value), value, (theta_mesh, p_mesh),
                                                    method=interp)
    value_resized = np.resize(value, p_interp.shape)
    print(p_interp.shape,value.shape)
    # Calculate RMSE and MAE
    rmse = math.sqrt(((p_interp - value_resized) ** 2).mean())
    mae = np.abs(p_interp - value_resized).mean()
    # Store the RMSE and MAE for this method
    rmse_dict[interp] = rmse
    mae_dict[interp] = mae
    # Plot the results
    axs[i].plot_surface(theta_mesh, p_mesh, p_interp, cmap='viridis')
    axs[i].set_title(interp.capitalize() + ' interpolation')
    axs[i].set_xlabel('Angle')
    axs[i].set_ylabel('Pressure')
    axs[i].set_zlabel('Re')
print("RMSE:")
for key, value in rmse_dict.items():
    print(key + ": " + str(value))

print("\nMAE:")
for key, value in mae_dict.items():
    print(key + ": " + str(value))

plt.tight_layout()
plt.show()
```

Figure C.4: Code for griddata surface interpolation.....continuation from Fig. B.3

Appendix D

Datasets

Angle	Skin friction
0.006664	0
5.418582	0.353177
10.43176	0.755688
15.24889	1.108892
20.87906	1.445622
25.89224	1.848132
30.31507	2.119163
35.34268	2.414824
40.17648	2.64474
45.43234	2.75135
50.48661	2.849749
55.35817	2.800212
60.03258	2.709587
65.14683	2.364148
70.09281	1.763922
75.34367	0.407518
78.03547	-0.01178
80.29964	-0.26668
85.38835	-0.42308
90.26657	-0.52193
95.33306	-0.51394
100.3962	-0.4813
105.4549	-0.41577
110.1116	-0.37489
115.1758	-0.35046
119.6387	-0.37533
125.3133	-0.36737
130.1793	-0.37581
134.8481	-0.42534
139.9146	-0.41735
144.5768	-0.41756
149.846	-0.40958
154.3021	-0.38513
159.9745	-0.36073
164.6267	-0.28697
170.0975	-0.27078
175.1584	-0.2217
179.7884	0.016448

Angle	Pressure
0	1
5.347194	0.988873
10.52365	0.889814
15.32623	0.756328
20.32932	0.580806
25.34204	0.363235
30.16476	0.14183
35.1801	-0.08721
40.01508	-0.36213
45.04268	-0.64468
49.87066	-0.88902
55.08389	-1.14863
60.31112	-1.4694
65.12509	-1.65258
70.11592	-1.77458
74.98592	-1.36908
80.09495	-1.1738
84.67338	-1.16204
90.40146	-1.16932
94.98251	-1.16903
99.9497	-1.18782
104.9143	-1.19515
110.258	-1.19099
114.8416	-1.20216
119.8669	-1.20312
124.7673	-1.20153
129.5418	-1.21269
135.2699	-1.21998
139.6601	-1.2197
145.0046	-1.21935
149.9675	-1.21904
154.7394	-1.21873
159.894	-1.22223
164.8541	-1.21045
169.8187	-1.21777
174.7815	-1.21746
179.9352	-1.21713

Figure D.1: Input data for $Re = 10^5$

Angle	Skin friction
0.577786	0.007665
4.926903	0.352698
9.86265	0.659393
14.7925	0.996779
20.10361	1.349494
24.84431	1.671543
30.15984	2.001239
34.72022	2.261914
39.85987	2.50722
44.81477	2.71417
49.78442	2.844392
60.04381	2.938584
64.95032	2.897454
69.96271	2.805169
74.98837	2.64383
79.6534	2.359743
85.51086	1.868445
90.19751	0.471826
93.75119	-0.02705
95.31013	-0.1422
99.94126	-0.24982
105.3275	-0.28841
110.1272	-0.27327
115.3125	-0.26582
119.926	-0.28136
125.3049	-0.28159
130.6823	-0.27415
135.0992	-0.26666
140.4885	-0.3206
145.3014	-0.37451
149.7316	-0.43608
155.1238	-0.50537
159.9293	-0.52092
164.9196	-0.49811
169.8951	-0.39858
174.8618	-0.25301
179.8167	-0.04606
55.68338	2.911239

Angle	Pressure
0	1
4.617404	0.993699
9.82983	0.930931
15.06195	0.806725
20.11815	0.631369
25.1842	0.425294
30.2601	0.1885
34.96283	-0.08403
40.26715	-0.43352
44.99286	-0.77773
50.08517	-1.06572
55.18732	-1.38443
60.07581	-1.63654
64.96267	-1.88352
69.82655	-2.05883
75.06359	-2.1984
81.17856	-2.07709
85.1435	-1.84773
90.05694	-1.57766
94.99171	-1.37414
99.53199	-1.13981
103.7172	-0.99754
109.8715	-0.99911
114.8572	-0.95431
119.8461	-0.91976
124.6542	-0.92099
129.8436	-0.91208
134.4643	-0.92862
139.662	-0.94531
144.4684	-0.94142
149.6611	-0.94275
154.461	-0.91838
159.4549	-0.89918
164.2498	-0.85945
169.2387	-0.82489
174.0485	-0.83124
179.6291	-0.84291

Figure D.2: Input for $Re = 2.6 \times 10^6$

Angle	Skin friction
0	0
5.016722	0.457416
10.03344	0.893612
15.05017	1.287368
20.06689	1.628074
25.08361	1.989999
30.301	2.288311
35.11706	2.586529
40.33445	2.916672
45.1505	3.172449
50.36789	3.417711
55.18395	3.546168
60.40134	3.611058
65.01672	3.612147
70.03344	3.539059
75.05017	3.381091
80.46823	3.106506
85.48495	2.736336
90.301	2.206968
95.31773	1.603376
100.1338	0.904246
105.1505	0.661397
110.3679	0.736898
115.3846	1.183704
120.2007	4.240542
125.4181	4.220552
130.8361	3.033501
136.4548	1.581245
140.6689	0.542451
145.2843	0.140356
148.6957	0.003229
150.7023	-0.16606
155.7191	-0.31342
160.3344	-0.35477
165.3512	-0.38542
170.3679	-0.33118
175.786	-0.13892
180.4013	0.010705

Angle	Pressure
-0.15009	0.995225
4.62569	0.966424
9.687375	0.851595
15.34157	0.703296
20.39683	0.545456
25.14263	0.315939
30.18147	0.048183
34.92299	-0.21001
39.6645	-0.4682
44.84701	-0.77419
49.72933	-1.08973
54.46443	-1.39093
59.6555	-1.63958
64.39916	-1.88343
69.74746	-2.07951
74.50397	-2.23734
79.55637	-2.4143
84.46582	-2.54824
89.53179	-2.6344
94.60204	-2.69188
99.22345	-2.75413
104.3001	-2.7686
109.226	-2.79263
119.6651	-1.90879
124.7782	-1.67954
129.9255	-1.22089
135.2172	-0.7957
140.1688	-0.64769
145.2541	-0.60481
150.3571	-0.44246
154.837	-0.45214
159.9144	-0.46183
164.9918	-0.47153
169.7719	-0.47165
175.0015	-0.46223
180.3784	-0.46715
-0.15009	0.995225

Figure D.3: Input Data $Re = 8.5 \times 10^5$

Theta	P	Re	Theta	P	Re	Theta	P	Re
0	1	1.00E+06	0	1	2.60E+05	-0.15009	0.99523	8.50E+05
5.34719	0.98887	1.00E+06	4.6174	0.9937	2.60E+05	4.62569	0.96642	8.50E+05
10.5236	0.88981	1.00E+06	9.82983	0.93093	2.60E+05	9.68738	0.85159	8.50E+05
15.3262	0.75633	1.00E+06	15.0619	0.80672	2.60E+05	15.3416	0.7033	8.50E+05
20.3293	0.58081	1.00E+06	20.1182	0.63137	2.60E+05	20.3968	0.54546	8.50E+05
25.342	0.36324	1.00E+06	25.1842	0.42529	2.60E+05	25.1426	0.31594	8.50E+05
30.1648	0.14183	1.00E+06	30.2601	0.1885	2.60E+05	30.1815	0.04818	8.50E+05
35.1801	-0.08721	1.00E+06	34.9628	-0.08403	2.60E+05	34.923	-0.21001	8.50E+05
40.0151	-0.36213	1.00E+06	40.2672	-0.43352	2.60E+05	39.6645	-0.4682	8.50E+05
45.0427	-0.64468	1.00E+06	44.9929	-0.77773	2.60E+05	44.847	-0.77419	8.50E+05
49.8707	-0.88902	1.00E+06	50.0852	-1.06572	2.60E+05	49.7293	-1.08973	8.50E+05
55.0839	-1.14863	1.00E+06	55.1873	-1.38443	2.60E+05	54.4644	-1.39093	8.50E+05
60.3111	-1.4694	1.00E+06	60.0758	-1.63654	2.60E+05	59.6555	-1.63958	8.50E+05
65.1251	-1.65258	1.00E+06	64.9627	-1.88352	2.60E+05	64.3992	-1.88343	8.50E+05
70.1159	-1.77458	1.00E+06	69.8265	-2.05883	2.60E+05	69.7475	-2.07951	8.50E+05
74.9859	-1.36908	1.00E+06	75.0636	-2.1984	2.60E+05	74.504	-2.23734	8.50E+05
80.095	-1.1738	1.00E+06	81.1786	-2.07709	2.60E+05	79.5564	-2.4143	8.50E+05
84.6734	-1.16204	1.00E+06	85.1435	-1.84773	2.60E+05	84.4658	-2.54824	8.50E+05
90.4015	-1.16932	1.00E+06	90.0569	-1.57766	2.60E+05	89.5318	-2.6344	8.50E+05
94.9825	-1.16903	1.00E+06	94.9917	-1.37414	2.60E+05	94.602	-2.69188	8.50E+05
99.9497	-1.18782	1.00E+06	99.532	-1.13981	2.60E+05	99.2234	-2.75413	8.50E+05
104.914	-1.19515	1.00E+06	103.717	-0.99754	2.60E+05	104.3	-2.7686	8.50E+05
110.258	-1.19099	1.00E+06	109.872	-0.99911	2.60E+05	109.226	-2.79263	8.50E+05
114.842	-1.20216	1.00E+06	114.857	-0.95431	2.60E+05	119.665	-1.90879	8.50E+05
119.867	-1.20312	1.00E+06	119.846	-0.91976	2.60E+05	124.778	-1.67954	8.50E+05
124.767	-1.20153	1.00E+06	124.654	-0.92099	2.60E+05	129.926	-1.22089	8.50E+05
129.542	-1.21269	1.00E+06	129.844	-0.91208	2.60E+05	135.217	-0.7957	8.50E+05
135.27	-1.21998	1.00E+06	134.464	-0.92862	2.60E+05	140.169	-0.64769	8.50E+05
139.66	-1.2197	1.00E+06	139.662	-0.94531	2.60E+05	145.254	-0.60481	8.50E+05
145.005	-1.21935	1.00E+06	144.468	-0.94142	2.60E+05	150.357	-0.44246	8.50E+05
149.967	-1.21904	1.00E+06	149.661	-0.94275	2.60E+05	154.837	-0.45214	8.50E+05
154.739	-1.21873	1.00E+06	154.461	-0.91838	2.60E+05	159.914	-0.46183	8.50E+05
159.894	-1.22223	1.00E+06	159.455	-0.89918	2.60E+05	164.992	-0.47153	8.50E+05
164.854	-1.21045	1.00E+06	164.25	-0.85945	2.60E+05	169.772	-0.47165	8.50E+05
169.819	-1.21777	1.00E+06	169.239	-0.82489	2.60E+05	175.001	-0.46223	8.50E+05
174.782	-1.21746	1.00E+06	174.048	-0.83124	2.60E+05	180.378	-0.46715	8.50E+05
179.935	-1.21713	1.00E+06	179.629	-0.84291	2.60E+05			

Figure D.4: Pressure-Data for Griddata for $Re = 10^5$,$Re = 2.6\times10^5$, $Re = 8.5\times10^5$

Theta	S	Re	Theta	S	Re	Theta	S	Re
0.00666	0	1.00E+06	-0.57779	0.00766	2.60E+05	0	0	8.50E+05
5.41858	0.35318	1.00E+06	4.9269	0.3527	2.60E+05	5.01672	0.45742	8.50E+05
10.4318	0.75569	1.00E+06	9.86265	0.65939	2.60E+05	10.0334	0.89361	8.50E+05
15.2489	1.10889	1.00E+06	14.7925	0.99678	2.60E+05	15.0502	1.28737	8.50E+05
20.8791	1.44562	1.00E+06	20.1036	1.34949	2.60E+05	20.0669	1.62807	8.50E+05
25.8922	1.84813	1.00E+06	24.8443	1.67154	2.60E+05	25.0836	1.99	8.50E+05
30.3151	2.11916	1.00E+06	30.1598	2.00124	2.60E+05	30.301	2.28831	8.50E+05
35.3427	2.41482	1.00E+06	34.7202	2.26191	2.60E+05	35.1171	2.58653	8.50E+05
40.1765	2.64474	1.00E+06	39.8599	2.50722	2.60E+05	40.3344	2.91667	8.50E+05
45.4323	2.75135	1.00E+06	44.8148	2.71417	2.60E+05	45.1505	3.17245	8.50E+05
50.4866	2.84975	1.00E+06	49.7844	2.84439	2.60E+05	50.3679	3.41771	8.50E+05
55.3582	2.80021	1.00E+06	60.0438	2.93858	2.60E+05	55.1839	3.54617	8.50E+05
60.0326	2.70959	1.00E+06	64.9503	2.89745	2.60E+05	60.4013	3.61106	8.50E+05
65.1468	2.36415	1.00E+06	69.9627	2.80517	2.60E+05	65.0167	3.61215	8.50E+05
70.0928	1.76392	1.00E+06	74.9884	2.64383	2.60E+05	70.0334	3.53906	8.50E+05
75.3437	0.40752	1.00E+06	79.6534	2.35974	2.60E+05	75.0502	3.38109	8.50E+05
78.0355	-0.01178	1.00E+06	85.5109	1.86845	2.60E+05	80.4682	3.10651	8.50E+05
80.2996	-0.26668	1.00E+06	90.1975	0.47183	2.60E+05	85.4849	2.73634	8.50E+05
85.3883	-0.42308	1.00E+06	93.7512	-0.02705	2.60E+05	90.301	2.20697	8.50E+05
90.2666	-0.52193	1.00E+06	95.3101	-0.1422	2.60E+05	95.3177	1.60338	8.50E+05
95.3331	-0.51394	1.00E+06	99.9413	-0.24982	2.60E+05	100.134	0.90425	8.50E+05
100.396	-0.4813	1.00E+06	105.328	-0.28841	2.60E+05	105.151	0.6614	8.50E+05
105.455	-0.41577	1.00E+06	110.127	-0.27327	2.60E+05	110.368	0.7369	8.50E+05
110.112	-0.37489	1.00E+06	115.313	-0.26582	2.60E+05	115.385	1.1837	8.50E+05
115.176	-0.35046	1.00E+06	119.926	-0.28136	2.60E+05	120.201	4.24054	8.50E+05
119.639	-0.37533	1.00E+06	125.305	-0.28159	2.60E+05	125.418	4.22055	8.50E+05
125.313	-0.36737	1.00E+06	130.682	-0.27415	2.60E+05	130.836	3.0335	8.50E+05
130.179	-0.37581	1.00E+06	135.099	-0.26666	2.60E+05	136.455	1.58124	8.50E+05
134.848	-0.42534	1.00E+06	140.488	-0.3206	2.60E+05	140.669	0.54245	8.50E+05
139.915	-0.41735	1.00E+06	145.301	-0.37451	2.60E+05	145.284	0.14036	8.50E+05
144.577	-0.41756	1.00E+06	149.732	-0.43608	2.60E+05	148.696	0.00323	8.50E+05
149.846	-0.40958	1.00E+06	155.124	-0.50537	2.60E+05	150.702	-0.16606	8.50E+05
154.302	-0.38513	1.00E+06	159.929	-0.52092	2.60E+05	155.719	-0.31342	8.50E+05
159.975	-0.36073	1.00E+06	164.92	-0.49811	2.60E+05	160.334	-0.35477	8.50E+05
164.627	-0.28697	1.00E+06	169.895	-0.39858	2.60E+05	165.351	-0.38542	8.50E+05
170.097	-0.27078	1.00E+06	174.862	-0.25301	2.60E+05	170.368	-0.33118	8.50E+05
175.158	-0.2217	1.00E+06	179.817	-0.04606	2.60E+05	175.786	-0.13892	8.50E+05
179.788	0.01645	1.00E+06	55.6834	2.91124	2.60E+05	180.401	0.0107	8.50E+05

Figure D.5: Skin friction-Data for Griddata for $Re = 10^5$, $Re = 2.6 \times 10^5$, $Re = 8.5 \times 10^5$