DISS. ETH NO. 24625

Real-Time 3D Flow Visualization Technique with Large Scale Capability

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZÜRICH
(Dr. sc. ETH Zürich)

presented by

Andreas Müller

MSc ETH Masch.-Ing., ETH Zürich
born September 23, 1984
citizen of Switzerland

accepted on the recommendation of

Prof. Dr. Thomas Rösgen, examiner
Prof. Dr.-Ing. Jochen Wiedemann, co-examiner
Dr. Andrin Landolt, co-examiner

2017

Real-Time 3D Flow Visualization Technique with Large Scale Capability

Published and distributed by:

Institute of Fluiddynamics
ETH Zürich
ETH Zentrum, ML H35
CH-8092 Zürich
Switzerland

http://www.ifd.mavt.ethz.ch

Printed in Switzerland by:

Druckzentrum ETH Zentrum
ETH Zentrum, HG E 39
CH-8092 Zürich
Switzerland

Abstract

The use of quantitative flow visualization techniques like particle image velocimetry (PIV), particle tracking velocimetry (PTV) or probe traversing in commercial wind tunnel testing is rather the exception than the rule, often hindered by the comparatively complex and time-consuming setup. In practice, qualitative visualization methods like mini-tufts and smoke flow visualization are often the only applicable option, especially when measuring in or around complex geometries.

In this work, a novel quantitative flow visualization method (ProCap) is developed with the goal to enhance the productivity of aerodynamic testing. To this end, the orientation and position of a hand-held probe (e.g. multi-hole pressure probe, thermoelectric anemometer) are optically tracked using a motion capture camera system with sub-millimeter accuracy. The simultaneous recording of the probe's output signal allows interpolating the flow data onto a regular grid. To provide some real-time feedback to the operator, a GPU-accelerated interpolation and rendering scheme is applied. By continuously scanning the probe around the test article the reconstructed field eventually converges to the time-averaged flow field. The interactive, real-time visualization of the measured flow is crucial for the efficiency of the method as it allows the operator to focus specifically on regions with complex flow structures. To date, ProCap supports the following visualization features: Contour slices, vector slices, streamlines, and isosurfaces.

The scanning approach offers great flexibility as with different probes the volumetric distribution of virtually any flow quantity can be recorded. So far, the system has been tested mostly with a five-hole probe, a device that allows one to measure the static pressure, the flow velocity magnitude and the flow direction, albeit in a limited range of angles.

To enhance the consistency and quality of the measured velocity field, the applied interpolation method offers the possibility to account for physical constraints such as mass conservation.

The visualization method was successfully tested on flows around different aerodynamic models. Comparisons with measurements from a 3-axis traversing system reveal that the setup and measurement time is reduced by a factor of about 20 with acceptable losses in accuracy. This circumstance underlines the method's potential to fill the gap between traditional flow visualization and high-precision, high-complexity measurement techniques.

Zusammenfassung

Der Einsatz von quantitativen Strömungsvisulisierungsmethoden in kommerziellen Windkanalmessungen ist eher die Ausnahme als die Regel, meist aufgrund des komplexen und aufwendigen Aufbaus. In der Praxis kommen qualitative Visualisierungstechniken wie Wollfäden oder Visualisierungen mit Rauch häufiger zum Einsatz, dies gilt insbesondere bei Messungen in und um komplexe Geometrien.

In dieser Arbeit wird eine neue quantitative Strömungsvisualisierungsmethode (ProCap) entwickelt mit dem Ziel, die Produktivität von aerodynamischen Untersuchungen zu verbessern. Zu diesem Zweck wird mit Hilfe eines optischen Motion Capture Systems, welches submillimeter Genauigkeit hat, die Orientierung und die Position einer handgeführten Sonde (z.B. Mehrlochsonde, thermoelektrische Anemometer) bestimmt. Das gleichzeitige Aufnehmen des Sondensignals erlaubt es, die gemessenen Strömungsdaten auf ein reguläres Gitter zu interpolieren. Um dem Anwender ein Feedback in Echtzeit bereitzustellen, wird für die Interpolation und räumliche Darstellung eine programmierbare Grafikkarte (GPU) eingesetzt. Mit fortlaufender Messdauer konvergiert die Rekonstruktion zum zeitlich gemittelten Strömungsfeld. Die interaktive Visualisierung der gemessenen Strömung in Echzeit ist für die Effizienz der Methode von entscheidender Bedeutung. Sie ermöglicht dem Anwender, sich auf Gebiete mit interessanten Strömungsstrukturen zu fokussieren. Zurzeit unterstützt ProCap die folgenden Visualisierungsfunktionen: Kontourflächen, Vektorflächen, Stromlinien und Isoflächen.

Das Messkonzept von ProCap zeichnet sich nicht nur durch seine Effizienz aus, sondern bietet auch diverse Einsatzmöglichkeiten. Unter anderem kann durch den Einsatz verschiedener Sonden die räumliche Verteilung beliebiger Strömungsgrössen gemessen werden. Bisher wurden die meisten Messungen mit einer sogenannten Fünflochsonde durchgeführt. Diese druckbasierte Messtechnik eignet sich für Strömungen mit einer bekannten Hauptströmungsrichtung und ermöglicht das gleichzeitige Messen von Druck, Geschwindigkeit und Geschwindigkeitsrichtung.

Zur Verbesserung der Messqualität wurde das Interpolationsschema für die Geschwindigkeit so angepasst, dass die Massenerhaltung der Strömung inhärent erfüllt ist.

Verschiedene Strömungen wurden mit der neuen Messmethode untersucht. Vergleiche mit Traversierungen belegen die Effizienz und das Leistungsvermögen von ProCap. Die Messzeit verkürzt sich um zirka das 20-fache. Allerdings ist die räumliche Auflösung der komplexen Strömungsstrukturen etwas schlechter. Nichtsdestotrotz besitzt die Methode das Potential, die bestehende Lücke zwischen komplexen, quantitativen und einfachen, rein-qualitativen Messsystemen zu schliessen.

Acknowledgment

Many people have contributed either directly or indirectly to this project for which I am very grateful. In particular, I would like to thank Prof. Dr. Thomas Rösgen for his help and advice I received throughout my work at IFD. Without his support and ideas, this thesis would not have been possible. Also, I would like to thank Dr. Andrin Landolt who initiated and supervised the project. His numerous ideas and suggestions as well as his enthusiasm for the project were of great help. I also wish to thank Prof. Dr.-Ing. Jochen Wiedemann for acting as a co-examiner, reviewing my thesis and his interest in the project.

Moreover, I am grateful to Rene Holliger and Pius Stachel for technical assistance and their ideas and suggestions on technical aspects of the project. Specials thanks are due to Bianca Maspero and Sonia Atkison for their help in administrative matters.

I also wish to thank Martin Viertel, Lukas Kadek, Christoph Loy, Gabriela Fisch, Yves Gerster and Dominik Rehbock for working on various student projects which helped make this work a success. They deserve great credit for the effort they put into this project.

Furthermore, I would like to express my sincere thanks to my colleagues at the institute (Oliver Häuselmann, David Borer, Martin Ehrensperger, Dr. Tim Grünberg, Philipp Bühlmann, Hanna Berning, Bernhard Vennemann, Markus Schmidt, Dr. Lukas Prochazka and Dr. Alexander Meier) not only for their valuable comments on the project but also for being good friends with whom to work with was always a pleasure.

Thanks are also due to all the people at Yacht Research Unit, Auckland who made the tests in the TFWT possible, namely Prof. Richard Flay, David Le Pelley and Dr. Nick Velychko.

Finally, I would like to add personal thanks to my parents and my family for supporting me throughout my education. And last but not least, a special thanks goes to Carmen Röthlin, my girlfriend, for her encouragements and support during this very intense time at ETH.

Contents

Nomenclature

Abbreviations

A/D	analog-digital
API	application programming interface
CAD	computer-aided design
CFD	computational fluid dynamics
CMOS	complementary metal-oxide-semiconductor
CPU	central processing unit
DAQ	data acquisition
DEM	diffuse element method
ED	eigenvalue decomposition
ETH	Eidgenössische Technische Hochschule
eRTI	extended region to interpolate
FOV	field of view
GPGPU	general purpose computation on GPU
GPU	graphics processing unit
GUI	graphical user interface
I/O	input-output
IR	infra-red
LED	light-emitting diode
LPR	local polynomial regression
MLS	moving least squares
MoCap	motion capture
NACA	national advisory committee for aeronautics
NC	normalized convolution
PC	personal computer
PIV	particle image velocimetry
ProCap	probe capture
RBF	radial basis function
RDR	raw-data-region
ROI	region of influence
RT	real-time
RTI	region-to-interpolate

SVD	singular value decomposition
sMLS	solenoidal moving least squares
TFWT	twisted flow wind tunnel
VRPN	virtual-reality peripheral network

Roman

$\boldsymbol{A}$	auto-covariance matrix
$AMSE$	asymptotic mean squared error
$\boldsymbol{a}_i$	relative position vector of tap i
$\boldsymbol{B}$	Vandermonde matrix
$B_{\delta_{\boldsymbol{x}}}$	ball around $\boldsymbol{x}$ and with radius $\delta_{\boldsymbol{x}}$
$\boldsymbol{b}_j$	j-th basis function
$\boldsymbol{C}$	cross-covariance matrix
$\boldsymbol{c}_j$	coefficient of the j-th basis function
$c(\cdot)$	certainty function
c_α	pitch coefficient
c_β	yaw coefficient
c_s	pseudo-static pressure coefficient
c_t	pseudo-total pressure coefficient
c_{p_i}	pressure coefficient of the i-th pressure tap
D	denominator of the calibration pressure coefficients
d	diameter of the probe tip or space dimension
E	mean squared error in the orthogonal Procrustes problem
$e(\boldsymbol{x})$	point-wise error
e_i	measurement error for sample i
$\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3}$	principle axes
f	arbitrary function
$\boldsymbol{G}$	Gram matrix
h	shape parameter
$h_{\mathcal{X},\Omega}$	fill distance
k_i	dimensionless pressure difference between tap i and 0
L	noise level
M	number of sample points

M_Δ	number of samples per histogram bin
$M_\mathcal{T}$	number of samples taken within the period $\mathcal{T}$
M_v	number of samples in a voxel
Ma	Mach number
ME	mean of the error
MSE	mean squared error
n	order of a polynomial
N	number of markers
	or number of basis functions
$p(\cdot)$	arbitrary polynomial
p_{avg}	average pressure
p_{dyn}	dynamic pressure
p_i	pressure measured by the i-th pressure tap
p_s	static pressure
$\tilde{p}_s$	pseudo-static pressure
p_t	total pressure
$\tilde{p}_t$	pseudo-total pressure
$q(\cdot,\cdot)$	separation measure
q_q	rotation quaternion
$q_\mathcal{X}$	separation distance
R	number of evaluation points
$\boldsymbol{R}$	rotation matrix
$\boldsymbol{R}_*$	optimal rotation matrix
Re_d	Reynolds number based on the length d
$RMSD$	root-mean-square deviation
$RMSE$	root-mean-square error
s	certainty threshold
$\boldsymbol{t}$	translation vector
$\boldsymbol{t_*}$	optimal translation vector
Tu	turbulence intensity
$\boldsymbol{u}$	velocity vector
	or sampling vector
$\boldsymbol{u_a}$	apparent velocity vector
$\boldsymbol{u_p}$	probe velocity vector
u	velocity component in x-direction
	or an arbitrary function to be approximated
u_i	i-th sample of u
v	velocity component in y-direction

w	velocity component in z-direction
$w(\cdot, \boldsymbol{x}_0)$	weight function centered at $\boldsymbol{x}_0$
$\boldsymbol{W}$	Weight matrix
x, y, z	Cartesian space variables
$\boldsymbol{x_c}$	reference position of the probe's centroid
$\boldsymbol{x_i}$	reference position of marker i
	or i-th data site
$\boldsymbol{x_{pt}}$	reference position of the probe tip
$\boldsymbol{x_0}$	evaluation point
$\boldsymbol{y_c}$	measured position of the probe's centroid
$\boldsymbol{y_i}$	measured position of marker i
$\boldsymbol{y_{pt}}$	measured position of the probe tip
$\boldsymbol{z_i}$	transformed reference position of marker i

Greek

α	pitch angle
β	yaw angle
$\Delta(\cdot),\ \delta(\cdot)$	deviation of a variable
Δ_{shear}	length scale of the local flow gradient
Δ_{wall}	distance to the next wall
δ_{max}	maximum support radius
$\delta_{\boldsymbol{x}}$	support radius of the weight function centered at $\boldsymbol{x}$
θ_i	angle between the flow direction and tap i
$\lambda,\ \boldsymbol{\lambda}$	Lagrangian multiplier(s)
$\Lambda_{m,\boldsymbol{\gamma}}$	Lebesgue function of the m-th order MLS estimate
	of the $\boldsymbol{\gamma}$-th derivative
$\mu(\cdot)$	linear functional
$\boldsymbol{\chi}$	trajectory of a fluid particle
ρ	density
σ	standard deviation
σ_0	cut-off standard deviation
σ_u	standard deviation of the quantity u
τ_{63}	the 63% rise time
ϕ	arbitrary measured variable

	or rotation angle
	or radial weight function
ψ_i	ith generating function
Ω	domain of interest

Calligraphic

$\mathcal{B}$	linear approximation space
$\mathcal{I}_{\boldsymbol{x}}$	indices collection of the data sites found in $B_{\delta_{\boldsymbol{x}}}$
$\mathcal{P}_m^d$	d-dimensional polynomial space of degree m
$\mathcal{T}$	integral time scale
$\mathcal{X}$, $\mathcal{Y}$	point sets
$\mathcal{X}_*$	set of points located in the support of w

Other symbols, accents and sub- and superscripts

$\langle \cdot, \cdot \rangle$	inner product
$\langle \cdot, \cdot \rangle_{w(\cdot, \boldsymbol{x}_0)}$	weighted semi-inner product
$\lVert \cdot \rVert_{w(\cdot, \boldsymbol{x}_0)}$	norm induced by the semi-inner product
$\widehat{(\cdot)}$	estimate
$\widetilde{(\cdot)}$	variable expressed relative to the body centered coordinate system
$(\cdot)_q$	quaternion
$(\cdot)_\infty$	ambient quantity
$(\cdot)'$	rotation vector
$\widehat{\delta^{\boldsymbol{\alpha}}(\cdot)}$	diffuse derivative in multi-index notation
$\mathrm{E}[\cdot]$	expectation value
$\mathrm{Var}\,[\cdot]$	variance

Chapter 1

Introduction

Despite rapid progress in computational fluid dynamics (CFD) over the last few decades, wind tunnel testing has remained indispensable in the optimization and certification of aerodynamic shapes. As conventional wind tunnel tests focus primarily on the measurement of the force and moment characteristics of a model, their contribution to the understanding of the surrounding fluid flow often remains limited. However, to improve the shape of a model in a deterministic and efficient way, detailed knowledge of the flow topology around the model is required. Hence, there is a growing need for flow visualization techniques that are both efficient and capable of covering large measurement volumes.

One possibility to collect topological information about the flow is the use of traditional flow visualization techniques such as smoke visualization or mini-tufts. The main advantage of these methods is that their implementation and operation is quick and intuitive. On the downside, however, the results are purely qualitative, and in general, the extraction of quantitative data for later comparison is almost impossible.

Alternatively, the flow around a test article can be investigated by non-intrusive, quantitative visualization methods such as particle imaging velocimetry, particle tracking velocimetry or laser Doppler imaging. If properly used, these methods provide high-quality, spatially and temporally resolved flow data. In practice, these techniques are rarely used in wind tunnel environments since their installation and operation remain complex and time-consuming. In particular, optical constraints (reflections, illumination power, seeding) make it almost impossible to cover a large volume in a single measurement run. Another entirely different strategy is to scan the area of interest by a traversing system in conjunction with a point measurement probe (e.g. pressure probe, hot wire or laser Doppler anemometer). Besides being expensive in terms of measurement time, complex mo-

del geometries or different angles of attack make the adaptation and control of such a system more difficult.

While these high-precision quantitative flow visualization methods are the right choice for thorough flow studies, in a design iteration loop (wind tunnel measurement, interpretation of results, design modification) traditional flow visualization tools are more suitable as time plays a crucial role. However, to enhance the productivity, quantitative data, which provides a full picture of the flow, would be of great value. To the best of the author's knowledge, at present no system exists that produces quantitative flow data on the one hand and offers the same efficiency and flexibility (e.g. setup- and measurement time, handling) as traditional flow visualization methods on the other hand (see figure 1.1). The aim of this work is to develop an interactive tool for quantitative flow visualization in and around complex geometries by focusing on the gap between traditional and high-precision, high-complexity flow visualization techniques. The working principle of this method is based on optically tracking the 3D position and orientation of a hand-held probe (e.g. multi-hole pressure probe, thermoelectric anemometer). In parallel with the tracking, the data signal of the probe is recorded, processed and spatially interpolated onto a regular grid. To accelerate the measurement, a feedback loop is created by displaying the processed data in real-time to the probe operator.

To reach its full potential and to facilitate the system's use in commercial wind tunnel campaigns, the development of the novel method has to focus on:

- **Real-time capability:**
  One way of reducing measurement time is to provide an instant feedback of the measurement to the control unit (here, the operator). This requires the extraction of high-level information from the acquired raw data set in real-time.

- **Short setup and measurement time:**
  In many commercial testing scenarios, the long setup and/or measurement time is the main reason why quantitative flow visualization is rather the exception than the rule. Therefore, it is vital for the new system to have a shorter setup and measurement time compared to conventional methods.

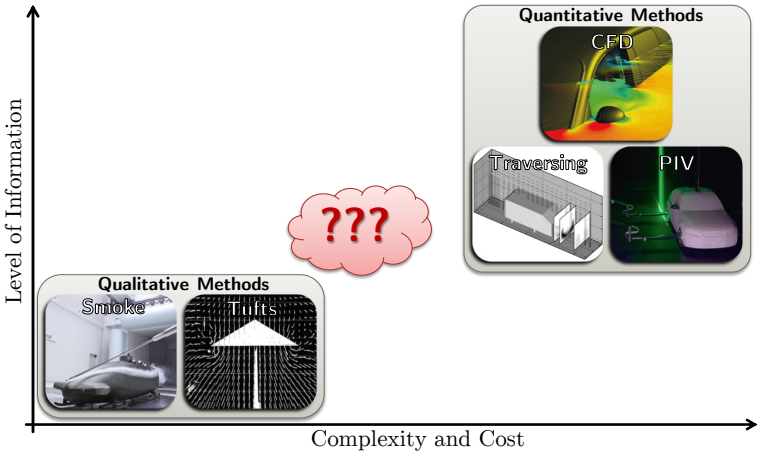


Figure 1.1: This diagram illustrates the gap between traditional flow visualization techniques and existing quantitative flow visualization methods. Image sources: Smoke: BMW Wind tunnel (retrieved June 16, 2017, from `http://www.autobild.de`); Tufts: *Bird and Riley* (1952); CFD: Ansys CFX (retrieved June 16, 2017, from `http://www.ansys.com`); Traversing: *Lienhart et al.* (2002); PIV: RUAG Aerodynamics (retrieved June 16, 2017, from `https://aerodynamics.ruag.com`)

- **Large-scale capability:**
  In general, the cost and complexity of established, quantitative flow visualization methods increase with the size of the area of interest. For wind tunnel measurements, it is quite common that the model under investigation is larger than the maximum volume that can be covered by a single measurement.

- **Volumetric measurement capability:**
  Planar measurements usually provide not all information required to fully understand the 3D flow structures around a wind tunnel model. Therefore, the novel approach shall measure and visualize volumetric flow data.

- **Low-complexity operation:**
  The ease of operation is one of the main reasons why visualization with a smoke probe is still in use. The hand-held approach allows the user to focus on regions of specific interest.

- **Post-processing capability:**
  Similar to CFD, the new method shall offer the possibility to post-process and analyze in detail the measured flow field at a later stage.

This application-driven viewpoint for the novel visualization tool asks for scientific developments in different fields. On the experimental side, the fusion of the data from several sensors poses the main challenge. On the algorithmic and computational side, the difficulties arise in the processing, interpolation, and visualization of the measured data in real-time and with the specific measurement point distribution of free-hand scanning.

The present work is structured as follows: In chapter 2, the new flow visualization system (ProCap) is introduced. It also contains a description of the main components. As the current system is most often used in conjunction with a five-hole probe, chapter 3 addresses issues related to the probe calibration and the integration into the ProCap environment. For a detailed description of the flow field reconstruction method and its implementation into ProCap, the reader is referred to chapter 4. Chapter 5 provides a brief outline of the applied visualization concepts, and in the final chapter, some of the recent measurement results are presented and reviewed.

Chapter 2

System description

2.1 Overview

ProCap stands for probe capture and is a quantitative flow visualization method specifically designed for large-scale wind tunnel measurements (*Landolt et al.* (2016), *Mueller et al.* (2012)). The method essentially combines the measurement capabilities of a traversing system with the flexibility of a hand-operated (smoke) probe. Figure 2.1 provides a general idea of the working principle. The test engineer (operator) scans the domain of interest with a hand-guided flow probe fitted with retro-reflective markers. For reduced interference, the operator stands well away from the flow region being investigated, preferably outside the test section. The instantaneous position of the probe is determined in real-time by means of a pas-

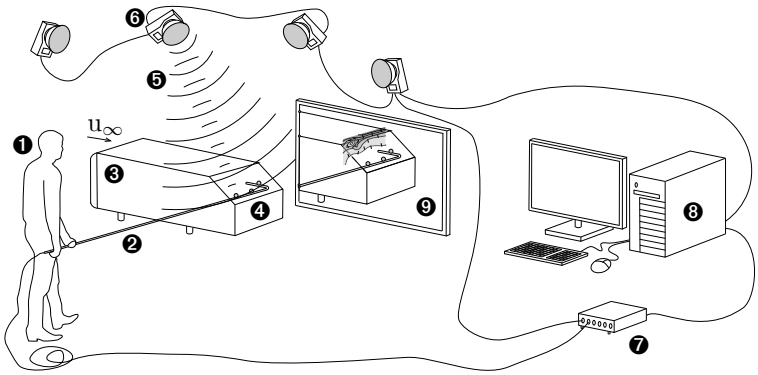


Figure 2.1: Experimental setting: ❶ operator, ❷ measuring probe, ❸ aerodynamic model, ❹ retro-reflective markers, ❺ IR illumination, ❻ cameras, ❼ DAQ-board, ❽ MoCap and ProCap software, ❾ feedback display

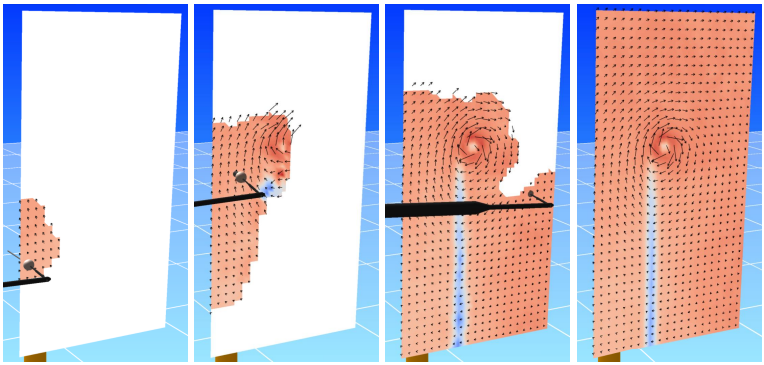


Figure 2.2: Snapshots of the real-time visualization in chronological order.

sive marker-based, commercially available motion capture system. Such a system consists of $n \geq 2$ synchronized cameras. High-power LEDs (contained in the housing of the cameras) emit light flashes in the infra-red range which are reflected by the markers on the probe. This reflected light is captured by the cameras allowing the determination of the markers' positions in each image plane. Triangulation is the process that combines these 2D-Data with the camera parameters so as to compute the 3D position of each marker. Once the marker positions are known, one can easily deduce the position and orientation of the probe.

As the signal of the probe is sampled time-synchronously with the tracking, the two data sets can be fused to form a single, multi-valued point cloud. For visualizing the flow field in real-time, these point data have to be interpolated onto a regular grid. ProCap makes use of a GPU-accelerated implementation of a local, adaptive approximation method called moving least-squares (consult chapter 4 for details). This method is not only efficient, but it can also cope with highly non-uniformly distributed data. The processed data are then displayed in real-time on some display visible to the operator guiding the probe. In this way, a feedback loop is created which is necessary to significantly speed up the measurement process (if compared to a measurement with a traversing system) and to

display high-level results to the users. With the help of this visual feedback, regions containing interesting flow structures are quickly identified, and the probe can be navigated accordingly. Usually, the measurement is continued until the real-time feedback is converged (see figure 2.2), i.e. meaning that the operator no longer detects changes in the field while crossing it with the probe.

ProCap can be used in conjunction with any probe capable of measuring the quantity of interest fast enough (that is commensurate with the probe's motion in the flow field). The probe used here is a conventional five-hole probe that allows one to measure the static pressure, the velocity magnitude and the flow direction, albeit in a limited range of angles.

In the remainder of this chapter, the key components of the system are briefly described. On the hardware-side these are:

- the motion capture system
- the data acquisition board
- the five-hole probe

On the software side, this is the real-time visualization software.

2.2 Hardware

2.2.1 Motion capture system

An integral component of the ProCap measurement system is the positional tracking of the probe. In recent years, tracking systems have found widespread use in fields such as robotics, augmented reality, medical applications, film animations, etc. As a result, numerous tracking systems have become commercially available. ProCap is not restricted to a single system since the device-independent VRPN (Virtual-Reality Peripheral Network) interface is used for communication. There are VRPN drivers available for more than thirty different systems*. However, it is important to mention that not all

*A list of supported trackers can be found on
https://github.com/vrpn/vrpn/wiki/Available-hardware-devices

systems are suitable for ProCap. ProCap pursues two goals: First, it acts as a real-time flow visualization tool. The real-time visualization is crucial for the overall efficiency of the measurement, i.e. the quick localization of the relevant flow features and update about the convergence progress of the interpolated flow field. Second, ProCap is supposed to provide quantitative data for post-processing. Both tasks make demands on the properties of the underlying tracking system:

1. **Latency:** Since in ProCap the operator acts as a controller, the delay of the visualization of the virtual scene should be as small as possible. For augmented reality and virtual reality applications, it is recommended to keep the time lag shorter than $15ms$ in order to provide a realistic experience. As the probe localization is only one part of the whole process chain, the latency of the tracking system should remain well below $15ms$. To enhance the user experience ProCap processes the positional data of the probe in two independent threads. One thread transfers the position of the probe to the rendering pipeline with the least possible delay, while the other combines the positional data with the measurements from the flow sensor. In this way, the time-consuming tasks (e.g. velocity correction, interpolation) do not interfere with the rendered position of the probe.

2. **Update rate:** From film industry, it is known that the human brain requires at least $15fps$ to perceive a scene in motion as natural and continuous.

3. **Accuracy:** The tracking performance is of great importance for the overall quality of the flow measurement. The tracking accuracy should at least match the spatial resolution capability of the probe used.

4. **Working range:** The distance between the sensing device(s) and the region of interest usually depends on the size of the wind tunnel. Typically, the width and height of the test section are on the order of a few meters.

Figure 2.3: Oqus3 camera (retrieved June 16, 2017, from `http://www.qualisys.com`)

Regarding these requirements, optical motion capture systems seem to be the best choice.

The results shown in this thesis are exclusively based on a high-end marker-based optical tracking system from Qualisys AB, Sweden. The system contains four Oqus3 cameras (see figure 2.3). The technical specifications of this CMOS camera model are listed in table 2.1. The cameras are synchronized to each other and can record up to $500 fps$. The position of a marker in the image plane is determined with subpixel accuracy.[†] The cameras are equipped

Table 2.1: Specifications of the Oqus3 camera

pixel pitch	$14.6\ \mu m$
resolution	1280×1024 px (81920×65536 subpx)
max. frame rate	$500\ fps$ (full FOV)
positional noise	typically ± 1 subpx
aperture	$f/2.8 - f/22$
focal length	$25mm$

[†]Qualisys defines the size of a subpixel as 1/64 of a pixel.

with a $25mm$ lens providing approximately a 40° horizontal field-of-view. To make the retro-reflective markers detectable, the lens is surrounded by a LED strobe light emitting in the near-infrared. A cut-off filter is used to increase the sensitivity of the system by blocking reflections from daylight. The aperture, flash duration and exposure time can be adjusted according to the circumstances. Typically, the exposure time is on the order of $100\mu s$. The working distance is limited by the power of the LEDs and the resolving capacity of the cameras. According to Qualisys, the Oqus3 cameras have a maximum capture distance of approximately $22m$.[‡] One advantage of the Qualisys system is that the Oqus cameras provide configurable sync in and out ports for synchronization with external hardware. The internal clock has an accuracy of 1 parts per million. Another strength of the Qualisys system is the comparatively low latency. Normally, the time elapsed between exposure and output is less than $4ms$, even for cases with several hundred markers and more than ten cameras. The triangulation algorithm is highly optimized and allows the system to run on an ordinary laptop without performance losses.

2.2.2 Data acquisition board

Similar to the tracking, ProCap accesses the data signal(s) of the probe through a VRPN connection.[§] In this work, a high-end A/D card from Measurement Computing is used, namely the USB-2533 DAQ-board (see figure 2.4). It features a 16-bit/$1MHz$ A/D converter coupled with either 64 single-ended or 32 differential analog input channels. The range can be set for each channel separately. Four of the analog input channels can be configured to measure data signals from thermocouples. The A/D converter can be triggered by an external clock which simplifies the synchronization with the tracking system considerably. In addition to the analog channels, the USB-2533 board also offers 24 I/O digital channels.

VRPN does not include drivers for Measurement Computing devi-

[‡] based on $16mm$ markers

[§] National Instruments A/D cards are on the list of supported devices

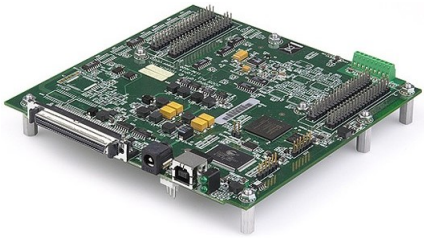


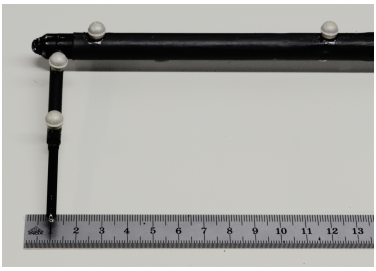
Figure 2.4: Measurement computing USB-2533 board (retrieved June 16, 2017, from `http://www.mccdaq.de`)

ces. This problem is solved by a tailored server application that fills the data recorded by the board into time-stamped VRPN packages of the type `vrpn_Analog`. These packages can then be accessed by the client (ProCap software).

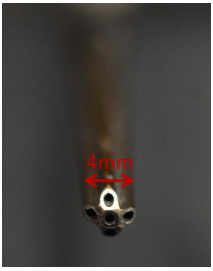
2.2.3 Five-hole probe

With different probe types, ProCap is capable of measuring the volumetric distribution of various spatially distributed flow quantities. In this work, a conventional five-hole probe is used. The working principle of a five-hole probe is based on measuring the pressure distribution across the probe tip by five differently oriented pressure taps. Assuming that the probe is properly calibrated, these five pressure values can be translated into local flow quantities, such as velocity magnitude, flow direction, and static pressure. However, this requires that the angle between the flow and the probe is smaller than 60 degrees. For more details on this subject, see chapter 3.

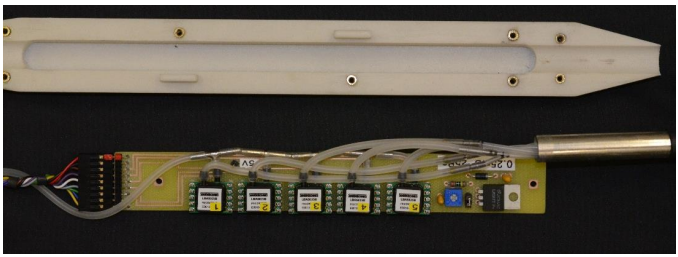
Figure 2.5a shows the probe used for the tests. To determine the instant position and orientation of the probe, its L-shaped head is fitted with four retro-reflective markers ($\varnothing 7mm$). As explained in section 3.3, the position of these markers is of crucial importance for the tracking accuracy. The pyramidal probe tip (see figure 2.5b) has a diameter of 4 mm and is formed by five longitudinally aligned tubes soldered together. The angle between two opposite taps is



(a) For the positional tracking four retro-reflective markers are glued on the L-shaped head.



(b) Tip of the five-hole probe used.



(c) Picture of the sensor unit.

Figure 2.5: Photographs of the five-hole probe

approximately 90 degrees. The tubes which connect the forward facing pressure taps with the pressure transducers have an inner diameter of $0.8mm$.

The distance from the probe tip to the sensor unit is about 30 cm. Basically, the sensor unit consists of a circuit board with five miniature, differential pressure transducers (see figure 2.5c). Each pressure transducer is assigned to one pressure tap. For the reference pressure a soft silicone tube, which extends from the sensors to the ambiance, is used. To cope with different velocity levels, the user can choose from three different sets of pressure transducers (see table 2.2). According to the manufacturer (First Sensor), all pressure sensors are temperature compensated to ensure accurate and precise measurements. While the sensors from the HCLA series

Table 2.2: Tested pressure transducers from First Sensor

	Range	T comp.	Type
HCLA12X5EB	$\pm 1250Pa$ ($\lessapprox$ 35 m/s)	✓	piezores.
HCLA02X5EB	$\pm 250Pa$ ($\lessapprox$ 20 m/s)	✓	piezores.
LBAS025BE	$\pm 25Pa$ ($\lessapprox$ 6 m/s)	✓	thermal

are fast response, piezoresistive pressure transducers, the more sensitive LBA-sensors are based on thermal micro-flow measurements and therefore are comparatively slow. According to the data sheet, the step response time τ_{63} is approximately $1ms$.

2.3 Software design

This section provides an overview of the software that allows the processed data to be displayed in real-time to the operator. It is based on Unity, a software framework (all purpose engine) designed for the creation and development of video games and graphics-focused applications. Of fundamental importance to the usability of the system is the ability to process and display measured data in real-time. Therefore, the primary focus in designing the software was performance followed by functionality and flexibility. Figure 2.6 is a schematic of the software architecture. For efficiency reasons, the workload is split between the CPU and the GPU. On the CPU two threads running concurrently in an asynchronous manner are involved, namely the main thread and the worker thread.

Figuratively speaking, the main thread is the nerve center of the application, where relevant information is pooled and the processes to be executed are coordinated. Below, a list is given of the tasks the main thread is responsible for:

- Setting up the scene (reading project settings, loading CAD-models, initializing probe and configuring scene camera)
- Controlling the worker thread (invocation and abortion)

- Passing measurement data to the GPU memory
- Invoking and scheduling the interpolation of the flow field carried out by the GPU.
- Invoking and scheduling other compute tasks on the GPU (e.g. generation of isosurfaces)
- Coordinating the rendering process
- Processing the user input (GUI)

At the start of the measurement the main thread invokes the worker thread to execute repeatedly the following tasks:

- Accessing data from the probe and the motion capture system via VRPN.
- Fusion of the position and probe data with the help of the provided time stamps.
- Transforming the measured raw data into flow quantities by making use of the probe calibration maps

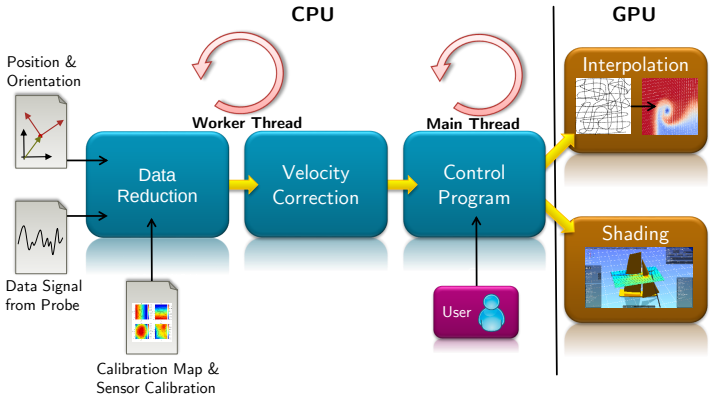


Figure 2.6: Building blocks of ProCap's visualization software.

- If required, applying corrections due to the motion of the probe (see section 3.4)
- Storing the processed data into a dedicated buffer that can be accessed by the main thread.

As the time stamps of the VRPN data packages are expressed in the time frame of the corresponding server, VRPN connections perform "quick syncs" at the rate of $4Hz$. These syncs determine the clock offset between server and client which in turn is required to transform the time stamps of the data packages into the client time frame.

Since the operations carried out by the worker thread are computationally inexpensive, the worker thread can easily run at the same rate as the motion capture system ($100Hz$ or higher). The update rate of the main thread, however, is limited to $60fps$ mainly to save computational power both on the CPU and GPU.

Per cycle, the GPU performs two tasks. First, the reconstructed flow field is locally updated based on the latest measurements (interpolation). Second, the virtual scene containing flow visualization elements such as contour slices, isosurfaces or vector plots is rendered. The measurement principle of the ProCap system complicates the interpolation in several respects. As the input data change over time, the interpolated flow field has to be refreshed at regular intervals. By taking the locality of the changes into account, the amount of work can be reduced significantly. Another problem to be solved is the irregularity of the data point distribution. More detailed information about this issue and the flow field reconstruction in general is provided in chapter 4. Similarly, the steps involved in the rendering of the measurement scene are taken up in chapter 5.

Chapter 3

Five-hole probe

The next sections take up a number of issues related to five-hole probes and their integration into ProCap. The first section is intended to provide some background information on multi-hole pressure probes. This is followed by a thorough analysis of the five-hole probe calibration process. First, we review the conventional calibration approach, then we introduce a few modifications that help to enlarge the angular range of the probe. The non-trivial question of how accurately an optical tracking system determines the position and orientation of a rigid body probe is tackled in the third section. This information is necessary to determine the overall measurement accuracy. The fourth and final section discusses a simple approach to reducing the measurement errors induced by the movement of the probe.

3.1 Background

Since the invention of the Pitot-static tube in 1732 by Henri Pitot, pressure-based fluid flow instruments have been extensively used both in industry and research. Inserting a conventional Pitot-static tube in the flow allows one to measure the static and total pressure of the flow at the position of the probe tip. By means of these two pressure readings, one can additionally compute the magnitude of the velocity, provided that the density of the fluid is known. It is clear that these measurements are only accurate if the probe is nearly aligned with the local flow direction and if the flow field's non-uniformity is small compared to the size of the probe tip. Thus, for complex flow fields, the usability of a Pitot-static tube is limited as the flow direction at the point of measurement is usually not known. To overcome these limitations, so-called multi-hole pressure probes can be employed. The principle of operation of these pressure

probes relies on the circumstance that the pressure distribution on the probe's head depends not only on the velocity but also on the direction of incoming flow. In other words, by proper calibration, measuring the pressure at a few distinct locations on the probe's surface is often sufficient to determine not only all three components of the velocity vector but also the static pressure. The concept of multi-hole pressure probes is relatively old, in fact, the first probes were tested as early as in the 1950s. Having been unrivaled for years, today, other, more advanced velocity measurement techniques such as hot-wire anemometry or laser Doppler velocimetry are available. Although these newer techniques may provide better temporal and spatial resolution, there are still a number of arguments that support the use of pressure-based probes:

- *Simplicity of operation:* Compared to other velocity sensing techniques, multi-hole pressure probes are relatively simple to operate.

- *Ease of fabrication:* The fabrication of standard pressure probes is rather simple and inexpensive. Traditionally, multi-hole pressure probes are manufactured either by soldering a bundle of longitudinally aligned tubes together or by drilling holes into a cylindrical bar. Both methods require mechanical finishing to shape the tip of the probe. Recently, the development of highly accurate 3D printing systems has not only simplified the fabrication process but also led to a significant reduction of the fabrication costs (*Hall and Povey*, 2017).

- *Wide application range:* Optical access to the point of measurement or seeding the flow with tracers is not required.

- *Reliability and robustness:* Pressure-based flow sensors are very robust flow sensing instruments. They can be operated even in very harsh conditions (e.g. high or low temperatures, corrosive fluids, particle-laden flows) and tolerate mechanical contact with the model. Also, age-related degradation generally affects the pressure transducers only and therefore, recalibration of the probe itself is not required.

- *All-in-one measurement tool:* To the best of the author's knowledge, multi-hole pressure probes are the only flow sensors capable of measuring the velocity vector and the static pressure simultaneously.

On the downside, multi-hole pressure probes must be inserted in the flow, making the technique intrusive. Normally, the flow is disturbed only locally, but in certain situations, the presence of a multi-hole pressure probe can alter the global behavior of the flow. For instance, it is commonly known that small disturbances can trigger flow phenomena such as flow separation or transition. To prevent such worst case scenarios, it is important to design the probe as small as possible. Compared to hot-wires, the temporal resolution of multi-hole pressure probes, which is essentially dictated by the tubing and the pressure transducers, is poor. High-end transducers may be able to measure pressure fluctuations up to $10kHz$. This is, however, still an order of magnitude lower than what a hot-wire can resolve. Another problem of multi-hole pressure probes is related to their calibration: Mathematically speaking, deducing the flow conditions from the pressure readings is an inversion problem.* The aim of the probe calibration is to provide explicit expressions that solve this inversion problem. It is known that the flow around the probe is rather complex and depends on a number of parameters (e.g. Re-number, Ma-number, wall proximity, turbulence intensity, flow history, etc.). The calibration attempts to take care of as many parameters as possible. It is, however, impossible to cover all parameter combinations. Thus, one is forced to make sophisticated assumptions in order to reduce the dimensionality of the inversion problem. Due to the complexity of the problem, the calibration does usually not account for effects induced by turbulence and interference with other bodies (e.g. walls or other probes). It is clear that these assumptions introduce new uncertainties and thus have a negative impact on the measurement accuracy. In short, the measurement quality of a multi-hole pressure probe is in general not as good as that of a hot-wire, except if the flow conditions are almost identical to those the probe is subjected to during the calibration runs (often laminar free stream flow).

*The well-posedness is not necessarily given.

Over the years, a large number of multi-hole probe designs have been proposed, each with different intentions, e.g. ease of use, ease of manufacturing, enlarging the flow angle and/or velocity range, reduction of the probe size, measurement of turbulent flows etc.. Detailed information on different designs and multi-hole pressure probes in general can be found in *Bryer and Pankhurst* (1971) and *Telionis et al.* (2009). In the present work, we will focus on five-hole probes only and their application in ProCap.

3.2 Five-hole probe calibration

The aim of this section is to give a detailed exposition of the five-hole probe calibration process. In the first part, we set up the notation and terminology. In the second part, we develop the theory of the conventional calibration process. In Part 3 we derive a new set of calibration coefficients that cover a broader range of yaw and pitch. Part 4 is devoted to how the calibration data is attained and how this data is parameterized in order to be used during operation. Then we briefly address the non-trivial question of how the flow state affects the accuracy of the measurement. The final part contains a brief discussion of the accuracy of the tested five hole probe.

3.2.1 Notation and principle of operation

As depicted in figure 3.1, the head of a five-hole probe typically consists of a central and four surrounding pressure sensing ports. Usually, the outer pressure taps are symmetrically arranged around the center hole. Five-hole probes can be classified by their tip shape into three different categories: the hemispherical, the conical and the pyramidal five-hole probe. Besides that, the pressure taps are either forward facing or perpendicular to the probe's surface. In the present work, a pyramidal probe with forward-facing pressure taps is used. Moreover, the angle between two opposite holes is approximately 90 degrees. For the numbering of the pressure taps and the definition of the probe-specific coordinate system, the reader is referred to

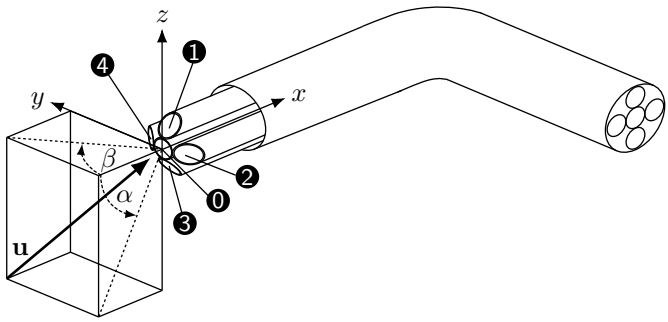


Figure 3.1: Definition of the pitch (α) and yaw (β) angle and numbering of the pressure taps.

figure 3.1. Mathematically, the flow direction relative to the probe can be fully described by two angles. Commonly, the polar and the azimuth angle of the corresponding spherical coordinate system are taken. To shorten the analysis below, we use here a slightly different definition of the flow angles (cf. figure 3.1):

$$\text{pitch: } \tan(\alpha) = \frac{w}{u} \qquad \text{yaw: } \tan(\beta) = \frac{v}{u}, \tag{3.1}$$

where $\boldsymbol{u} = [u, v, w]^T$ denotes the velocity vector relative to the probe-specific coordinate system. Clearly, this angle definition is only valid for a hemisphere, i.e $\alpha \in (-90°, 90°)$ and $\beta \in (-90°, 90°)$.

Five-hole probes can be operated in two different modes: The *null-reading* and the calibrated mode.

- In the null-reading mode, the angular orientation of the probe is adjusted such that the probe is aligned with the flow direction (nulling). For a symmetric tip shape, the probe is considered to be aligned with the flow if the pressure readings of two opposite holes are equal. When the probe is nulled, the orientation of the probe gives the direction of the velocity, whereas the central hole returns the stagnation pressure. The advantage of the null-reading method is that calibration becomes trivial as only a simple calibration map for the dyn-

amic pressure is required. But this comes at a cost, the nulling of the probe is not only very time-consuming (practically only steady flow measurements are possible), it also requires a complex probe traversing system that allows the probe to be rotated. Furthermore, the method relies on the assumptions that the probe is symmetric and that the probe tip is small compared to the length scales of the flow.

- In the calibrated mode, there is no need to change the orientation of the probe. Assuming that the probe is properly calibrated, the five pressure readings provide all information required to determine the flow direction, the flow speed, and the in-field pressure. On the one hand, this approach has the advantage of being more flexible than the nulling method, one the other hand the calibration task is much more involved. As ProCap is, by design, only compatible with the calibrated operation mode, the calibration is addressed in detail in the next subsections.

3.2.2 Conventional calibration process

For each hole $i = 0, 1, \ldots, 4$ one can define a dimensionless pressure coefficient

$$c_{p_i} = \frac{p_i - p_s}{\frac{1}{2}\rho|\boldsymbol{u}|^2}, \quad i = 0, 1, \ldots, 4 \tag{3.2}$$

where p_s stands for the static pressure, $|\boldsymbol{u}|$ for the true velocity, and ρ for the density. In addition, p_i stands for the pressure measured by the i-th pressure tap. Dimensional analysis reveals that these pressure coefficients depend on a number of parameters

$$c_{p_i} = f\left(\alpha, \beta, Re_d, Ma, \frac{\Delta_{\text{wall}}}{d}, \frac{\Delta_{\text{shear}}}{d}, Tu\right) \tag{3.3}$$

α and β are the flow angles as defined above, Re_d is the Reynolds number based on the diameter of the probe tip d, Ma is the local Mach number, Δ_{wall} is the distance to the next wall, Δ_{shear} is the length scale of the local flow gradient ($\|\boldsymbol{\nabla u}\|/|\boldsymbol{u}|$), and Tu

is the turbulence intensity. Provided that the probe is sufficiently small, the effect of the wall vicinity and of the flow gradient can be neglected. Moreover, it is quite common that the dependence on the Reynolds number, on the Mach number and on the turbulence intensity are also not taken into account. Of course, these simplifications are only physically meaningful if the flow conditions are similar to those during the calibration or if the mentioned dependencies are small within the range of application. Due to the hand-guided approach, ProCap allows only measurements at relatively low flow speeds. Thus, the Mach number plays indeed only a minor role and can be ignored without consequences. Regarding turbulence and Reynolds number effects, this may not be the case. For details, the reader is referred to the sections 3.2.5, but for now, we assume that the pressure coefficients depend on the flow angles only, i.e.

$$c_{p_i} = f\left(\alpha, \beta\right), \quad i = 0, 1, \ldots, 4. \tag{3.4}$$

The pressure coefficients cannot be determined by the pressure readings alone as the static and dynamic pressure are unknown. This issue is normally solved by combining the pressure coefficients in such a manner that the static and dynamic pressures drop out. To uniquely determine the flow angles, one has to find two of these combinations that form a bijective, vector-valued function over α and β. Typically, the following coefficients are taken (cf. *Dudzinski and Krause* (1969), *Treaster and Yocum* (1979)):

$$\begin{aligned} c_\alpha &:= \frac{c_{p_1} - c_{p_3}}{c_{p_0} - \frac{1}{4}\sum_1^4 c_{p_i}} = \frac{p_1 - p_3}{p_0 - \frac{1}{4}\sum_1^4 p_i} \quad \text{(pitch coefficient)} \\ c_\beta &:= \frac{c_{p_2} - c_{p_4}}{c_{p_0} - \frac{1}{4}\sum_1^4 c_{p_i}} = \frac{p_2 - p_4}{p_0 - \frac{1}{4}\sum_1^4 p_i} \quad \text{(yaw coefficient)} \end{aligned} \tag{3.5}$$

Although there is no real physical significance in these coefficients, the denominator is occasionally referred to as the pseudo-dynamic pressure. In addition, it is worth noting that the pressure difference $p_1 - p_3$ is mainly sensitive to changes of the pitch angle, while $p_2 - p_4$ senses yaw angle variations. This circumstance allows us to relate these coefficients to the quantities we intend to measure, namely α, β, the static and the dynamic pressure. More precisely, the bijecti-

vity of the function $f : [\alpha, \beta]^T \rightarrow [c_\alpha, c_\beta]^T$ guarantees the existence of the following functions:

$$\alpha\left(c_\alpha, c_\beta\right),\ \beta\left(c_\alpha, c_\beta\right),\ c_s\left(c_\alpha, c_\beta\right),\ c_t\left(c_\alpha, c_\beta\right) \tag{3.6}$$

Here, c_s and c_t denote the pseudo-static and pseudo-total pressure coefficient, which are defined as follows

$$c_s := \frac{\frac{1}{4}\sum_1^4 p_i - p_s}{p_0 - \frac{1}{4}\sum_1^4 p_i}, \quad c_t := \frac{p_0 - p_t}{p_0 - \frac{1}{4}\sum_1^4 p_i}. \tag{3.7}$$

p_t is a label for the total pressure, which is defined by the sum of the static pressure p_s and the dynamic pressure $p_{dyn} = 0.5\rho|\boldsymbol{u}|^{2\dagger}$ The aim of the five-hole probe calibration is to determine the precise relationships (3.6), commonly known as calibration maps. To this end, the probe is inserted into a known flow field while varying the pitch and the yaw angle. The five pressure values are recorded for all angle combinations. Subsequently, this data is used to deduce the calibration maps (3.6). This task is often referred to as data reduction step and many different techniques have been proposed, e.g. look-up tables in combination with local interpolation, various spline interpolation methods, artificial neural networks, etc.. The data reduction method used for this work is explained in section 3.2.4.

3.2.3 Novel set of calibration coefficients

For small flow angles (approx. $\pm 30°$ for a probe with cone angle 90°) the coefficients $c_\alpha, c_\beta, c_s, c_t$ introduced in the previous section work fine. At larger flow angles, however, one is confronted with two problems:

1. The denominator (pseudo dynamic pressure) passes through zero causing singularities of the pressure coefficients.

2. The flow over one or more pressure holes separates causing

[†]This definition is only valid for a subsonic flow.

loss of sensitivity. Furthermore, flow separation is known to be sensitive to changes of the Reynolds number and of the turbulent motion.

Different approaches have been proposed to overcome or to reduce these problems. Two solution categories can be distinguished:

- **Redefinition of the denominator:** A popular attempt to prevent singularities of the pressure coefficients is to apply a different denominator. One common approach is to replace the average pressure $p_{avg} = 1/4(p_1 + p_2 + p_3 + p_4)$ by the smallest pressure reading, i.e. $\min(p_1, p_2, p_3, p_4)$. However, this is problematic as it leads to kinks in the calibration surfaces. A more sophisticated but also a more complex modification of the denominator was proposed by *Pisasale and Ahmed* (2002). They come up with the idea of replacing the denominator by $D = p_0 - p_s + A(p_t - p_s)$, where the constant A is selected such that D is positive for all angle combinations. Since the static and the total pressure are unknowns, D cannot be computed from the five pressure readings alone. Therefore, *Pisasale and Ahmed* (2002) suggest the use of an additional 1D calibration function that relates the pressure values to the newly defined denominator D.

- **Sectioning:** Here, the angle range to be covered is split into different sectors (e.g. *Gallington* (1980), *Zilliac* (1993),*Paul et al.* (2011)). Typically, the pressure reading with the highest value determines the sector number. If properly designed, this kind of approach overcomes the singularity problem. However, it does not necessarily solve the separation problem (although sometimes stated otherwise) as according to *Dominy and Hodson* (1992) separation bubbles may occur even at angles as small as a few degrees. In addition, it leads to a more complicated calibration since continuity of the calibration coefficients at the boundary of two sections is not given and therefore, one has to create separate calibration maps for each zone. On top of that, there is no guarantee that the sectioning is not influenced by the flow state (e.g. turbulence, Re number, etc.). In the worst case, the calibration maps have to be extrapolated

to retrieve the flow properties (velocity, static pressure, etc.), which in turn may lead to unnecessary large errors.

Both approaches do not address the problem caused by flow separation. On the one hand, pressure readings which are affected by flow separation do not provide much information about the flow direction. On the other hand, and this is more problematic, flow separation is known to be sensitive to changes in the flow conditions (e.g. turbulence, Reynolds number). In the worst case, hysteresis effects may be observed. Consequently, reducing the influence of flow separation on the calibration maps is not primarily a matter of how one selects the pressure coefficients but of how the probe tip is designed (shape and position of the taps). For instance, an edged probe tip has the advantage that the flow topology around the probe is less dependent on the flow conditions as the sharp edges provoke separation (*Dominy and Hodson*, 1992).

Below, we derive a new set of pressure coefficients that bypass the singularity problem and thus allow calibration at angles of pitch and yaw much larger than the standard coefficients. But since the pressure coefficients are derived based on potential flow theory, they only perform well as long as the Reynolds number is sufficiently large.[‡] From potential flow theory, the normalized pressure distribution for a stationary flow around a sphere is given by (*Kundu et al.* (2016), eq. (7.106)):

$$\frac{p_i - p_s}{\frac{1}{2}\rho|\boldsymbol{u}|^2} = -\frac{5}{4} + \frac{9}{4}\cos^2(\theta_i) \tag{3.8}$$

where θ_i is the angle between the vector pointing upstream $-\boldsymbol{u}$ and $\boldsymbol{a}_i$, which is the vector from the sphere center to the point on the surface where the pressure p_i is required, i.e.

$$\cos(\theta_i) = -\frac{\boldsymbol{u} \cdot \boldsymbol{a}_i}{|\boldsymbol{u}||\boldsymbol{a}_i|} \tag{3.9}$$

For the next steps, the geometrical difference between a sphere and a forward facing pyramidal probe tip is neglected. If we assume a cone angle of 90°, the corresponding positions of the pressure taps

[‡]The standard coefficients face the same problem.

on the sphere are given by

$$\begin{aligned}
\frac{\boldsymbol{a}_0}{|\boldsymbol{a}_0|} &= [-1,0,0]^T & \frac{\boldsymbol{a}_1}{|\boldsymbol{a}_1|} &= \frac{1}{\sqrt{2}}[-1,0,1]^T \\
\frac{\boldsymbol{a}_2}{|\boldsymbol{a}_2|} &= \frac{-1}{\sqrt{2}}[1,1,0]^T & \frac{\boldsymbol{a}_3}{|\boldsymbol{a}_3|} &= \frac{-1}{\sqrt{2}}[1,0,1]^T \\
\frac{\boldsymbol{a}_4}{|\boldsymbol{a}_4|} &= \frac{1}{\sqrt{2}}[-1,1,0]^T
\end{aligned} \tag{3.10}$$

Furthermore, using the angle definition (3.1), the normalized flow vector can be expressed by

$$\frac{\boldsymbol{u}}{|\boldsymbol{u}|} = \frac{1}{\underbrace{\sqrt{1+\tan^2(\alpha)+\tan^2(\beta)}}_{=:N(\alpha,\beta)}}[1,\tan(\beta),\tan(\alpha)]^T. \tag{3.11}$$

Substituting these expressions into equation (3.8) leads to formulas for the theoretical pressure coefficients, which, as a matter of fact, depend on the flow angles only.

$$\begin{aligned}
c_{p_0} &= -\frac{5}{4} + \frac{9}{4N(\alpha,\beta)} & c_{p_1} &= -\frac{5}{4} + \frac{9\left(1-\tan(\alpha)\right)^2}{8N(\alpha,\beta)} \\
c_{p_2} &= -\frac{5}{4} + \frac{9\left(1+\tan(\beta)\right)^2}{8N(\alpha,\beta)} & c_{p_3} &= -\frac{5}{4} + \frac{9\left(1+\tan(\alpha)\right)^2}{8N(\alpha,\beta)} \\
c_{p_4} &= -\frac{5}{4} + \frac{9\left(1-\tan(\beta)\right)^2}{8N(\alpha,\beta)}
\end{aligned} \tag{3.12}$$

In order to eliminate the static pressure we introduce the following four coefficients

$$\begin{aligned}
k_1 &:= c_{p_1} - c_{p_0} = \frac{p_1-p_0}{\frac{1}{2}\rho|\boldsymbol{u}|^2} = \frac{\tan^2(\alpha)-2\tan(\alpha)-1}{N(\alpha,\beta)} \\
k_2 &:= c_{p_2} - c_{p_0} = \frac{p_2-p_0}{\frac{1}{2}\rho|\boldsymbol{u}|^2} = \frac{\tan^2(\beta)+2\tan(\beta)-1}{N(\alpha,\beta)} \\
k_3 &:= c_{p_3} - c_{p_0} = \frac{p_3-p_0}{\frac{1}{2}\rho|\boldsymbol{u}|^2} = \frac{\tan^2(\alpha)+2\tan(\alpha)-1}{N(\alpha,\beta)} \\
k_4 &:= c_{p_4} - c_{p_0} = \frac{p_4-p_0}{\frac{1}{2}\rho|\boldsymbol{u}|^2} = \frac{\tan^2(\beta)-2\tan(\beta)-1}{N(\alpha,\beta)}
\end{aligned} \tag{3.13}$$

The task is now to find two coefficients which on the one hand are

functions of either α or β only and on the other hand are independent of the dynamic pressure. By trigonometry, we can show that

$$\frac{k_1 - k_3}{k_1 + k_3} = \frac{p_1 - p_3}{p_1 + p_3 - 2p_0} = \tan(2\alpha) \tag{3.14}$$

and

$$\frac{k_4 - k_2}{k_2 + k_4} = \frac{p_4 - p_2}{p_2 + p_4 - 2p_0} = \tan(2\beta). \tag{3.15}$$

If we employ the following trigonometric relationship

$$\tan(\gamma) = \begin{cases} \frac{\tan(2\gamma)}{1+\sqrt{1+\tan^2(2\gamma)}} & \text{if } \gamma \in \left(-\frac{\pi}{4}, \frac{\pi}{4}\right) \\ \frac{\tan(2\gamma)}{1-\sqrt{1+\tan^2(2\gamma)}} & \text{if } \gamma \in \left(-\frac{\pi}{2}, -\frac{\pi}{4}\right) \cup \left(\frac{\pi}{4}, \frac{\pi}{2}\right) \\ 1 & \text{if } \gamma = \frac{\pi}{4} \\ -1 & \text{if } \gamma = \frac{\pi}{4} \end{cases} \tag{3.16}$$

and take into account that

$$\begin{cases} p_1 + p_3 - 2p_0 < 0 & \text{if } \alpha \in \left(-\frac{\pi}{4}, \frac{\pi}{4}\right) \\ p_1 + p_3 - 2p_0 > 0 & \text{if } \alpha \in \left(-\frac{\pi}{2}, -\frac{\pi}{4}\right) \cup \left(\frac{\pi}{4}, \frac{\pi}{2}\right) \\ p_1 + p_3 - 2p_0 = 0 & \text{if } \alpha \in \left\{-\frac{\pi}{4}, \frac{\pi}{4}\right\} \end{cases} \tag{3.17}$$

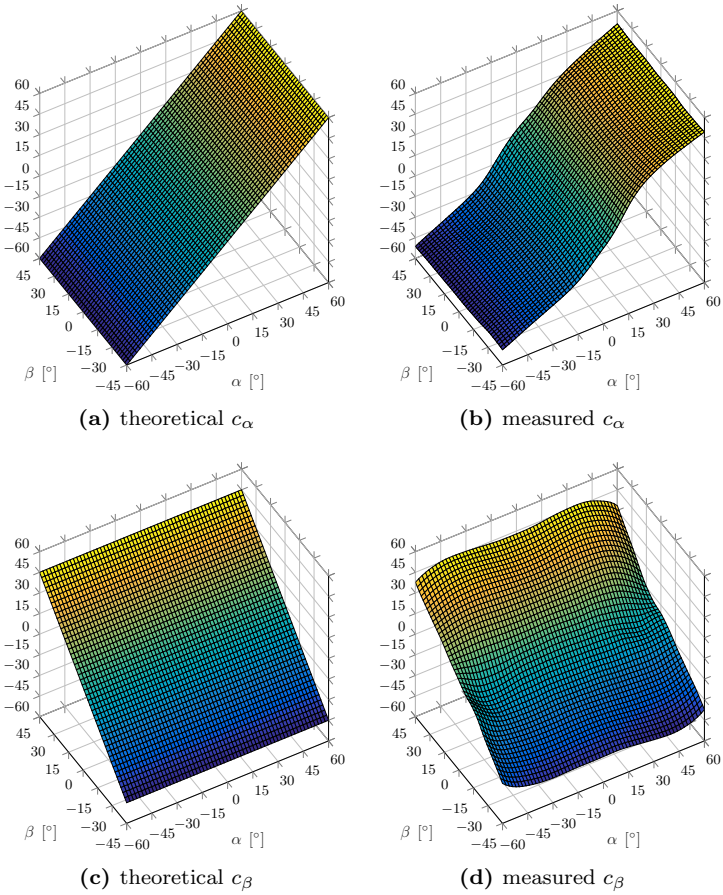
and

$$\begin{cases} p_2 + p_4 - 2p_0 < 0 & \text{if } \beta \in \left(-\frac{\pi}{4}, \frac{\pi}{4}\right) \\ p_2 + p_4 - 2p_0 > 0 & \text{if } \beta \in \left(-\frac{\pi}{2}, -\frac{\pi}{4}\right) \cup \left(\frac{\pi}{4}, \frac{\pi}{2}\right) \\ p_1 + p_3 - 2p_0 = 0 & \text{if } \beta \in \left\{-\frac{\pi}{4}, \frac{\pi}{4}\right\} \end{cases} \tag{3.18}$$

we finally obtain theoretical expressions for α and β, that do not depend on the dynamic pressure:

$$\begin{aligned} \alpha &= \tan^{-1}\left[\frac{p_1 - p_3}{p_1 + p_3 - 2p_0 - \sqrt{(p_1 - p_3)^2 + (p_1 + p_3 - 2p_0)^2}}\right] \\ \beta &= \tan^{-1}\left[\frac{p_4 - p_2}{p_2 + p_4 - 2p_0 - \sqrt{(p_4 - p_2)^2 + (p_2 + p_4 - 2p_0)^2}}\right] \end{aligned} \tag{3.19}$$

Figure 3.2 confirms the correctness of the derived result, both the-



(a) theoretical c_α **(b)** measured c_α

(c) theoretical c_β **(d)** measured c_β

Figure 3.2: Comparison between theory and measurement

oretically (left figure) and experimentally (right figure). While the theory is based on the potential flow solution around a sphere, the experimental data stems from a real flow around the pyramidal five-hole probe. Due to physical limitations, it was only possible to measure within the range $(\alpha, \beta) \in [-60°, 60°] \times [-45°, 45°]$. Despite

this, it is clearly visible that no singularities are present. Deviations between the experimental and the ideal plane are explained by the non-spherical shape of the probe tip, by imperfections in the manufacturing process and by the fact that potential flow is only an idealization of the real flow.

Now, based on the potential flow theory, the two denominators in eq. (3.19), i.e.

$$p_1 + p_3 - 2p_0 - \sqrt{(p_1 - p_3)^2 + (p_1 + p_3 - 2p_0)^2} \tag{3.20}$$

and

$$p_2 + p_4 - 2p_0 - \sqrt{(p_4 - p_2)^2 + (p_2 + p_4 - 2p_0)^2} \tag{3.21}$$

are identical ($= -4/N(\alpha, \beta)$). Consequently, we are allowed to replace these denominators by their arithmetic mean. This modification serves the purpose of reducing the uncertainty of the pitch and yaw coefficient, which we define as follows

$$c_\alpha = \tan^{-1}\left[\frac{\frac{1}{2}(p_3 - p_1)}{D}\right] \tag{3.22}$$

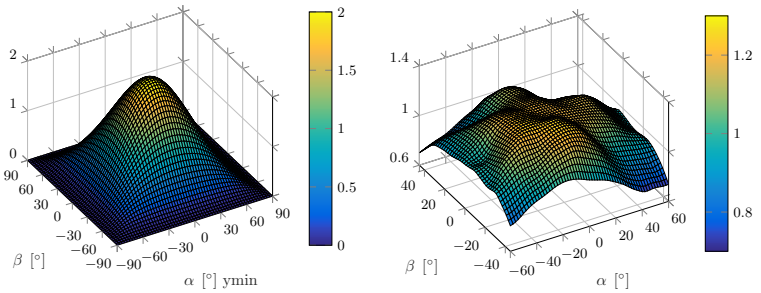
$$c_\beta = \tan^{-1}\left[\frac{\frac{1}{2}(p_2 - p_4)}{D}\right] \tag{3.23}$$

with

$$\begin{aligned} D := \quad & \left.\begin{aligned} & p_0 + \frac{1}{4}\sqrt{(p_1 - p_3)^2 + (p_1 + p_3 - 2p_0)^2} \\ & + \frac{1}{4}\sqrt{(p_2 - p_4)^2 + (p_2 + p_4 - 2p_0)^2} \end{aligned}\right\} =: \tilde{p}_t \\ & - \\ & \left.\frac{1}{4}(p_1 + p_2 + p_3 + p_4)\right\} =: \tilde{p}_s \end{aligned} \tag{3.24}$$

Here, $\tilde{p}_t$ and $\tilde{p}_s$ denote the pseudo-total and the pseudo-static pressure. By means of these new definitions, the static and total pressure coefficients can be redefined as well

$$c_s := \frac{\tilde{p}_s - p_s}{D}, \quad c_t = \frac{\tilde{p}_t - p_t}{D}. \tag{3.25}$$



(a) theoretical value of $\frac{D}{\frac{1}{2}\rho|\boldsymbol{u}|^2}$ **(b)** measured value of $\frac{D}{\frac{1}{2}\rho|\boldsymbol{u}|^2}$

Figure 3.3: Dimensionless denominator as a function of pitch and yaw

To improve the angular range of the probe, one has to guarantee that the denominator D is non-zero over a wider range of angles. As demonstrated in figure 3.3, the theoretical value of D is indeed non-zero for $(\alpha, \beta) \in (-90°, 90°) \times (-90°, 90°)$. The experimentally determined values are plotted in the right figure. Although the function looks less smooth, it certainly satisfies the requirement of not passing zero as shown in figure 3.3.

Besides circumventing singularities, it is also important to minimize the propagation of uncertainty. To this end, we compare the error propagation for the standard coefficients with that for the new set. The assessment below is based on the following assumptions:

1. The flow is adequately described by potential flow theory

2. There are no errors induced by the data reduction, i.e. $\alpha = \alpha\left(c_\alpha, c_\beta\right)$ and $\beta = \beta\left(c_\alpha, c_\beta\right)$ are exact

3. The uncertainties of the pressure readings are independent, have zero mean, are Gaussian distributed and have the same standard deviation δp

4. The uncertainties of the measured angles are negligible

Therefore, one can employ the following formula

$$\delta\phi = \delta p \sqrt{\sum_{i=0}^{4} \left(\frac{\partial \phi}{\partial p_i} \right)^2} = \frac{\delta p}{p_{dyn}} \sqrt{\sum_{i=0}^{4} \left(\frac{\partial \phi}{\partial c_i} \right)^2} \tag{3.26}$$

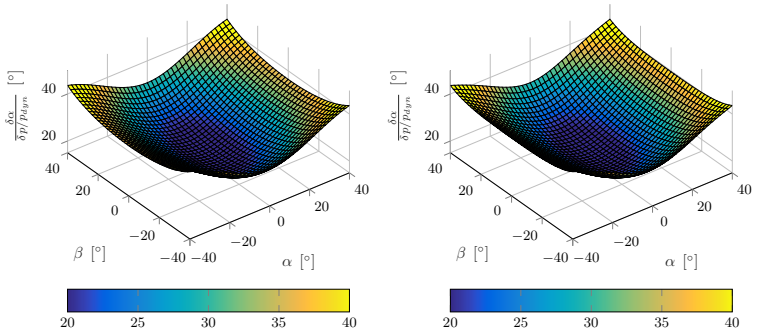
where ϕ is a physical quantity estimated by the pressure readings, e.g. α, β, c_s or c_t. Applying the chain rule on the derivatives $\partial\phi/\partial c_i$ leads to

$$\delta\phi = \frac{\delta p}{p_{dyn}} \sqrt{\sum_{i=0}^{4} \left(\frac{\partial \phi}{\partial c_\alpha} \frac{\partial c_\alpha}{\partial c_i} + \frac{\partial \phi}{\partial c_\beta} \frac{\partial c_\beta}{\partial c_i} \right)^2} \tag{3.27}$$

Using potential flow theory, these terms can be analytically evaluated for both calibration coefficients sets. Figure 3.4 illustrates the error amplification factor for the pitch angle (i.e. $\delta\alpha/(\delta p/p_{dyn})$) as a function of α and β. The domain covers only the angle range where the uniqueness of the standard method is ensured. It is clearly visible that within this domain the two methods perform similarly. However, it is worth pointing out that the assumption, that no error is introduced by the generation of the calibration map, rather applies to the new than to the standard approach. This is explained by the fact that for the standard method $c_\alpha(\alpha, \beta)$ and $c_\beta(\alpha, \beta)$ have a more complex shape than their newly derived counterparts.

Since the new approach aims at extending the probe's angular range, the error amplification at larger flow angles is also of interest. Figure 3.5 shows the amplification factor of the pitch angle for $(\alpha, \beta) \in [-60°, 60°] \times [-60°, 60°]$. Not surprisingly, the error grows with the magnitude of the pitch and yaw angle. At about 60° the amplification is roughly 4 times larger than at the center. For high-quality pressure transducers with a good signal-to-noise ratio, this might still be acceptable.

To sum up, with the presented calibration coefficients we do not face the problem of singularities in the calibration maps. In addition, the pitch and yaw angle coefficients are nearly one-dimensional, linear functions. Thus, it is suggested that the parameterization of the corresponding calibration maps requires fewer degrees of freedom than the conventional setting. Regarding propagation of uncertainty, at small flow angles the new calibration coefficients per-



(a) standard calibration method **(b)** new calibration method

Figure 3.4: Propagation of uncertainty: Comparison of the error amplification factor of the pitch angle.

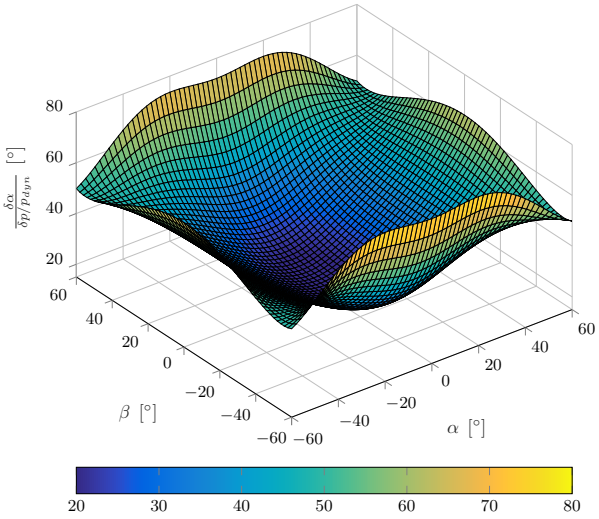


Figure 3.5: Propagation of uncertainty: Error amplification factor of the pitch angle for $(\alpha, \beta) \in [-60°, 60°] \times [-60°, 60°]$.

form similarly to the standard coefficients. At larger flow angles, however, the uncertainty level increases with pitch and yaw angle. Nonetheless, acceptable results are expected for angles less than 60°.

3.2.4 Data acquisition and reduction

To determine the calibration maps $\alpha\left(c_\alpha, c_\beta\right)$, $\beta\left(c_\alpha, c_\beta\right)$, $c_s\left(c_\alpha, c_\beta\right)$ and $c_t\left(c_\alpha, c_\beta\right)$, one has to measure the probe hole pressures for a large number of pitch and yaw angle combinations. Normally, the flow angles are varied by changing the probe orientation rather than the flow direction. This can be achieved by using a dedicated apparatus that allows one to adjust the probe orientation without changing the position of the probe tip (see figure 3.6). In most cases, the probe is positioned in the center of a uniform free jet with known flow properties (i.e. p_s, p_t and ρ). For every combination of pitch and yaw angle, the five pressure values are recorded for later use. One difficulty of this approach lies in the alignment of the probe with the jet. In the case of ProCap, the alignment problem is aggravated as in addition, the optically tracked probe frame has to be mapped to the physical frame of the probe. To bypass this problem, the calibration data for this work was acquired in the empty ETH wind tunnel by varying the probe orientation manually (see figure 3.7). As in the measurement, the probe is tracked optically by the MoCap system. In this way, one has only to map the wind tunnel flow direction to the tracked probe frame. Besides reducing the alignment problem, this approach also has the advantage that no additional mechanical calibration fixture is required. In order to ensure that the measured points cover the complete pitch-yaw range, the scatter plot of the point distribution is displayed to the operator in real-time. It may be criticized that this approach leads to inaccurate calibration data because the probe tip is not kept at a fixed position and since the laminarity of the wind tunnel flow may be insufficient. Regarding the first concern: In the literature, it is often stated that it is essential to keep the position of the probe tip fixed as even the best designed free jet may contain non-uniformities. However, this statement is questionable as in most cases the flow properties are

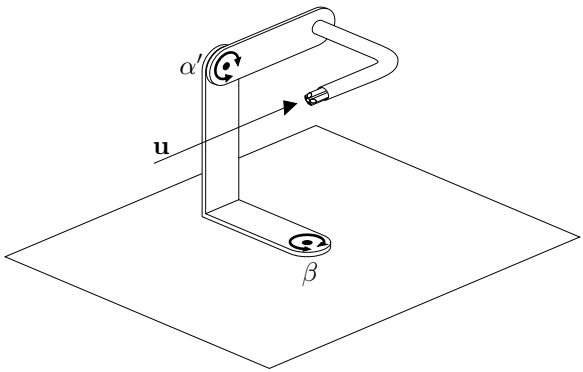


Figure 3.6: Setup for standard calibration

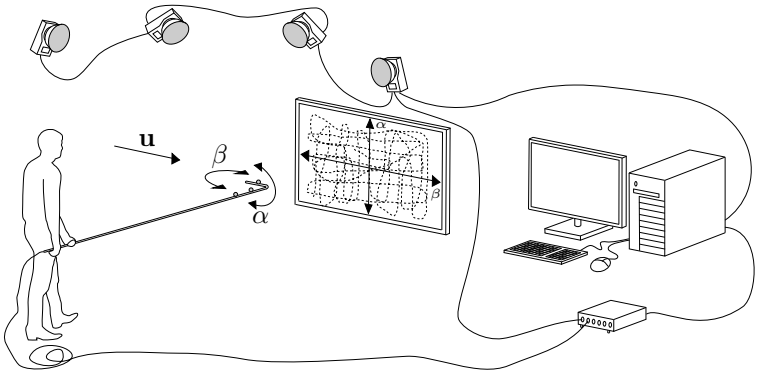


Figure 3.7: Setup for five-Hole probe calibration with ProCap

measured simultaneously at slightly different positions or are measured beforehand. In other words, there will always be a discrepancy between the measured and the true flow properties at the probe's location. Consequently, keeping the probe tip at a fixed position can introduce systematic errors in the calibration, which are of the same

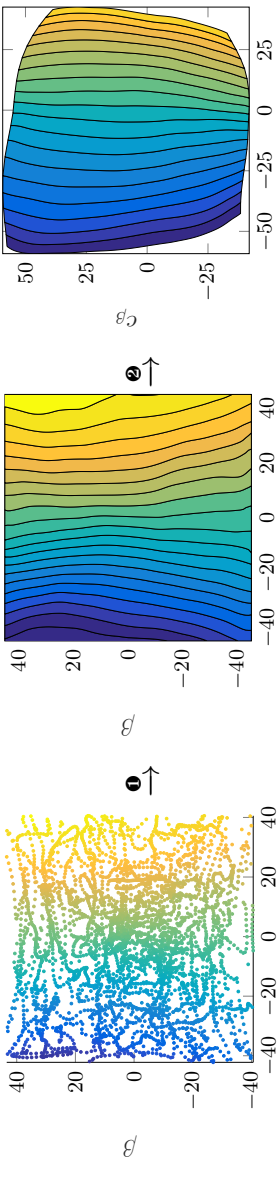
order as the non-uniformities of the jet. In contrast, if the probe tip randomly changes position within the core of the jet there is a chance of averaging out these errors. Regarding the concern about the flow laminarity: As turbulence influences the flow pattern on the probe tip, it is more reasonable to perform the calibration in flow conditions similar to the actual experiment. In order to reduce the variance of the pressure signals, it is suggested that oversampling is the better approach than physically lowering the pressure fluctuations of the flow.

Since the probe is directed by hand, one has to subtract the motion of the probe. Details on how this issue is solved are given in section 3.4.

Following the acquisition of the calibration data, the calibration maps are determined. As depicted in figure 3.8, the process that converts the raw data into calibration maps is divided into two steps:

1. **Interpolation onto a regular grid:** Varying the flow angles manually results in an irregular distribution of the collected data points in the pitch-yaw plane (cf. figure 3.8). The problem with an irregularly distributed point set is that fitting a global parametric model to the data leads to a poor approximation since areas with a high point density are weighted more strongly than others. To this end, the pressure coefficients are first interpolated onto a regular grid using a first-order moving least squares filter. The grid spacing is typically 1°. As later shown in chapter 4, the moving least squares interpolation is less affected by uneven point distributions than standard approximation methods.

2. **Generation of the calibration maps:** In the second step the calibration maps $\alpha(c_\alpha, c_\beta)$, $\beta(c_\alpha, c_\beta)$, $c_t(c_\alpha, c_\beta)$ and $c_s(c_\alpha, c_\beta)$ are computed. In the c_α-c_β-space the interpolation nodes are nearly evenly distributed, since the proposed c_α and c_β depend almost linearly on α and β, respectively. Hence, the calibration maps can be created using a global polynomial least squares



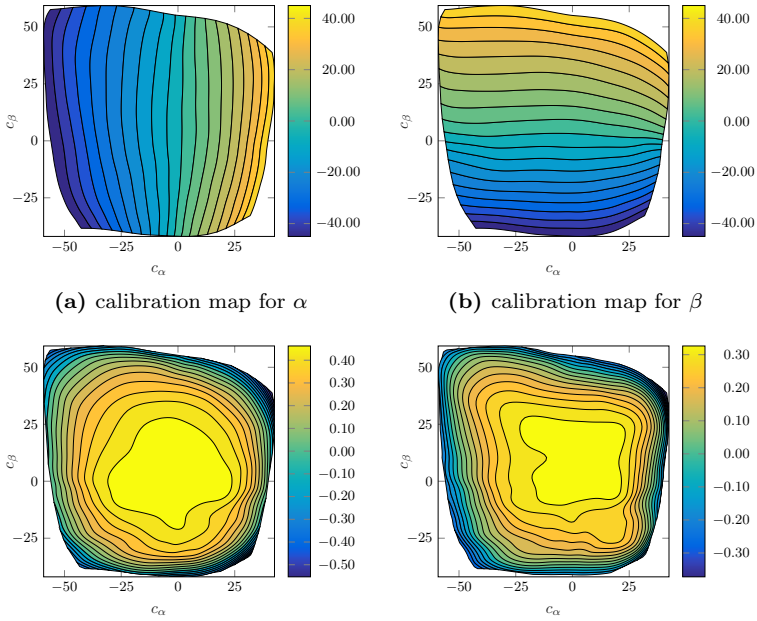
(a) raw data **(b)** interpolated data **(c)** calibration map

Figure 3.8: Schematic to illustrated the steps involved in the calibration process: ❶ interpolation onto a regular grid, ❷ generation of calibration map

fit:

$$\phi\left(c_\alpha, c_\beta\right) = \sum_{i=1}^{n} \sum_{j=1}^{n-i} a_{\phi,i,j} c_\alpha^i c_\beta^j, \quad \phi = \alpha, \beta, c_s, c_t \tag{3.28}$$

It is found that a sixth order polynomial fit (i.e. $n = 6$) meets the accuracy requirements of the application as the mean absolute error of the fit is typically less than $0.2°$ for both the pitch and yaw angles. If higher accuracy is required, a local polynomial spline interpolation is sometimes applied. For completeness, in figure 3.9 a graphic illustration of the calibration maps at an arbitrary Reynolds number ($Re_d = 3.2 \cdot 10^3$) is provided.



(a) calibration map for α

(b) calibration map for β

(c) calibration map for c_t

(d) calibration map for c_s

Figure 3.9: Calibration maps at $Re = 3.2 \cdot 10^3$

3.2.5 Potential error sources

The calibration scheme described above relies on the assumption that the Reynolds number and turbulence have little or no influence on the calibration coefficients. In this section the validity of this idealization is examined.

Reynolds number effects

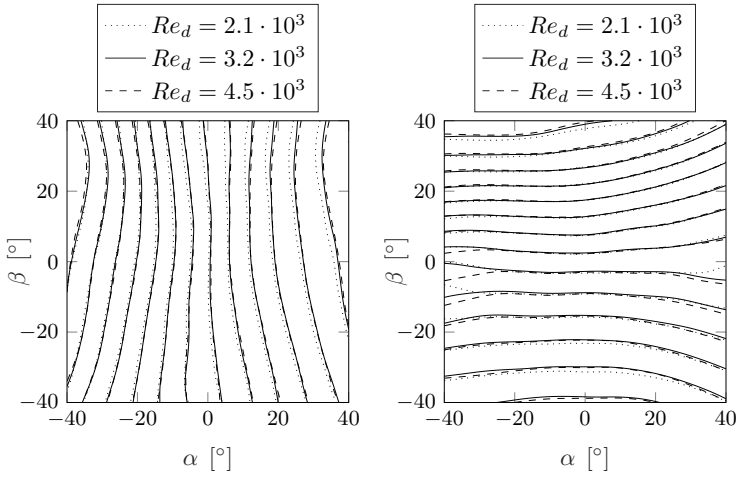
According to *Dominy and Hodson* (1992), two distinct Reynolds number effects influence the probe calibration:

- **Influence on the flow separation**: For a five hole with a cone angle of 90° this effect is observed if $Re_d < 1.5 \cdot 10^4$.
- **Influence on the structure of the flow over the sensing holes**: This effect is present at all Reynolds numbers. Moreover, *Dominy and Hodson* (1992) found that the impact on the dynamic pressure is the largest, while the yaw and pitch coefficients do not change significantly.

Additionally, at very low Reynolds numbers the assumption of inviscid flow does not hold and viscosity strongly affects the flow structure around the probe tip. The transition from an inviscid to a viscosity-dominated flow is gradually and eventually, leads to a flow governed by the laws of creeping flow (Stokes). For instance, if we suppose that the tip of the probe can adequately be described by a sphere, Stokes flow yields the following surface pressure distribution:

$$\frac{p_i - p_s}{\frac{1}{2}\rho|\boldsymbol{u}|} = 6\cos(\theta_i)\,\frac{1}{Re_d}, \tag{3.29}$$

where θ_i is the angle between the upstream flow direction and the vector from the center of the sphere to the point i on the surface. Although Stokes flow is a simplified assumption, it confirms experiments that report an increase in the stagnation pressure when the Reynolds number is lowered. More accurate solutions may be obtained by the theory of Oseen which partly accounts for inertia terms.



(a) pitch coefficient c_α **(b)** yaw coefficient c_β

Figure 3.10: Reynolds number effect

From experiments, it is known that for a Pitot-static tube the increase of the stagnation pressure becomes evident at $Re_d < 10^3$.

For a ProCap measurement Re_d often falls into the range with the highest Reynolds number sensitivity. Figure 3.10 displays the contour lines of c_α and c_β in dependence of the pitch and yaw angle for three slightly different Re_d. This example clearly demonstrates that even small Reynolds number variations can have a significant impact on the probe's accuracy. For instance, at large pitch and small yaw angle, the error of the measured yaw can be as large as 10° if the wrong calibration map is used. To reduce the influence of the Reynolds number and therefore to increase the reliability of the measurement, ProCap allows the user to upload multiple calibration files, which contain the calibration data for different Reynolds numbers. At each time step, ProCap selects the calibration file on the basis of the Reynolds number computed in the previous step (nearest neighbor criterion).

Effect of turbulence

Turbulence can bias the measurements of a five-hole probe significantly. One can distinguish the following effects:

1. Eddies, which are smaller than the tip size, lead to erratic measurements of the flow direction as well as of the static and dynamic pressure. This effect can be reduced by miniaturizing the probe. However, it is important to keep in mind that a smaller probe also implies a smaller Reynolds number and therefore can negatively impact the probe's measurement accuracy.

2. When the pressure over the probe tip is fluctuating in time, the pressure readings of the transducers do not necessarily reflect the actual pressure on the surface. Due to the flexible hoses the amplitude and the phase of the pressure signals are distorted. The distortion is typically a function of the signal's frequency. As theoretical models are imprecise, the exact transfer function has to be obtained experimentally. If this transfer function is known, one can easily deduce the true pressure values. In practical terms, this involves three steps:

   a) A fast Fourier transform is applied to a subset of the data. This subset is usually selected by a sliding box filter.

   b) In the frequency domain, the deconvolution is performed to correct the pressure signal, applying for example a suitable Wiener filter.

   c) Finally, the inverse Fourier transform is applied to the deconvolved data to obtain the corrected signal in time space.

   Since no acoustic calibration system was available, ProCap does not yet support this kind of signal correction. The effect on the time-averaged flow field is considered to be small.

3. Measuring highly fluctuating flows with a five-hole probe is delicate as acceleration effects come into play. This issue may best be explained by making use of the apparent mass model from potential flow theory. The pressure on the surface of

a sphere subjected to an unsteady flow with velocity $\boldsymbol{u}(t)$ is given by (*Kundu et al.* (2016), eq.(7.105))

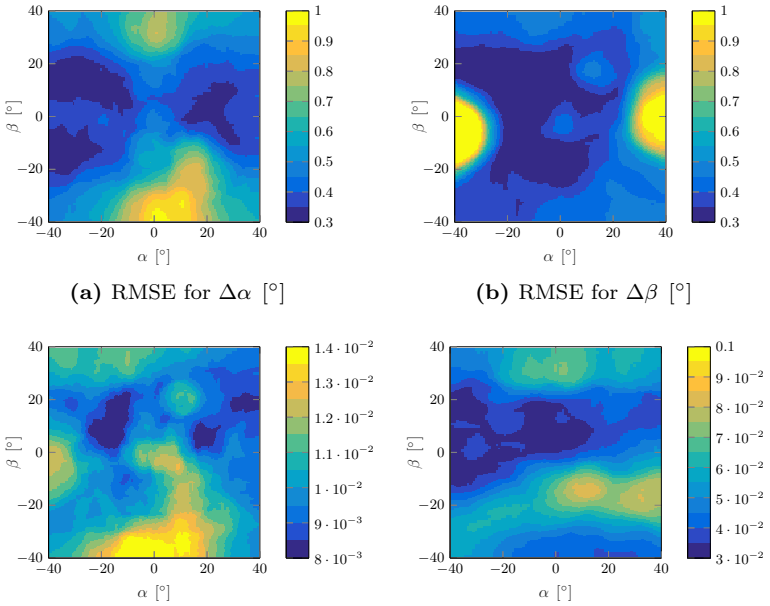
$$\frac{p_i - p_s}{\frac{1}{2}|\boldsymbol{u}|^2} = \underbrace{-\frac{5}{4} + \frac{9}{4}\cos^2(\theta_i)}_{c_{p_i,\text{steady}}} \underbrace{-\frac{d}{2|\boldsymbol{u}|^2}\boldsymbol{a}_i \cdot \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t}}_{c_{p_i,\text{unsteady}}}, \tag{3.30}$$

where d stands for the diameter of the sphere, $\boldsymbol{a}_i$ is the normalized vector from the sphere center to the point i on the surface, and θ_i is the angle between the $\boldsymbol{a}_i$ and $-\boldsymbol{u}$. As the calibration does not account for the unsteady term $c_{p_i,\text{unsteady}}$, highly fluctuating flows result in erroneous measurements. *Telionis et al.* (2009) estimated that for a probe with a tip diameter of 2mm and a flow that fluctuates with an amplitude of 10% of the mean velocity and with a frequency of $2kHz$ errors are caused in the computed dynamic pressure of approximately 6.3%.

4. It is known that turbulent disturbances in the flow can alter the flow separation behavior and the boundary layer of a bluff body. As most calibrations are performed in a laminar free jet, this can have an effect on the probe's performance. Interestingly, no publications were found discussing this particular issue.

3.2.6 Performance of the tested five-hole probe

The intention of this section is to evaluate the performance of the five-hole used for these tests. The accuracy of the probe is examined experimentally by subjecting the calibrated probe to a flow with known properties. The probe was calibrated at three different Reynolds numbers, namely at $Re = 2.1 \cdot 10^3$ ($\widehat{=} 8.5m/s$), $3.2 \cdot 10^3$ ($\widehat{=} 13.0m/s$) and $4.5 \cdot 10^3$ ($\widehat{=} 18.3m/s$). These Re-numbers cover the complete velocity range of the validation runs. The five validation runs are conducted in the ETH wind tunnel using the same setup as for the calibration (see figure 3.7). During these runs the probe is steered by hand and it is attempted to collect data



(a) RMSE for $\Delta\alpha$ [°]

(b) RMSE for $\Delta\beta$ [°]

(c) RMSE for $\Delta|\boldsymbol{u}|/|\boldsymbol{u}|$ [−]

(d) RMSE for $\Delta p_s/(\frac{1}{2}\rho|\boldsymbol{u}|^2)$ [−]

Figure 3.11: *RMSE* distribution

for all flow angle combinations that fall in the calibrated range $(\alpha, \beta) \in (-40°, 40°) \times (-40°, 40°)$. To prevent that the validation data becomes biased by only measuring at one Reynolds number, the validation runs are conducted at different flow speeds.

The measurement error per sample is defined by the difference between the measured value ϕ_i and the reference value ϕ:

$$e_i = \phi_i - \phi \tag{3.31}$$

It is worth noting that the reference values are not the true values as they contain uncertainties themselves. Potential error sources are:

- Measurement errors of the flow angles by the MoCap system (cf. section 3.3).

- Small temporal and spatial disturbances of the wind tunnel flow.

The error, as defined above, comprises:

- random errors (noise), and
- systematic errors (bias)

A common measure for the accuracy of a measurement system is the so-called root-mean-square error ($RMSE$):

$$RMSE = \sqrt{\frac{1}{M}\sum_i e_i^2} \tag{3.32}$$

Here, M is a label for the number of samples. A small $RMSE$ implies a high accuracy and vice versa, a low $RMSE$ implies a low accuracy. Unlike the accuracy, which accounts for both error components, the trueness is associated with the systematic error only. A widely used measure for the trueness is the mean of the error (ME), i.e.

$$ME = \frac{1}{M}\sum_i e_i \tag{3.33}$$

A small ME indicates a small systematic error and in turn a high trueness. Both the $RMSE$ and the ME of a five-hole probe are functions of the flow around the probe, in particular, they depend on α and β. Since the probe is hand-guided, it is impossible to collect data for each pitch-yaw angle combination individually. Despite this, one can estimate the distribution of the RMSE over α and β by accounting only for samples in the vicinity of a point. In practice, this requires a k-nearest neighbor search. The results are shown in figure 3.11 for the pitch α, the yaw β, the normalized velocity and the static pressure normalized by the dynamic pressure. Clearly, as expected in section 3.2.3 the $RMSE$ of the flow angles α and β is in general below 0.5° but can exceed 1° at large yaw and pitch. Moreover, the $RMSE$ of the velocity is typically smaller than 1%. More problematic is the static pressure, where $RMSE$ values as high as 10% are observed. This extreme values can partly be explained by the high uncertainties associated with the reference value. In

the ETH wind tunnel the static pressure is externally measured by means of a manual barometer.

Ignoring the dependence on α and β, one can compute the overall $RMSE$ and ME values, which may serve as rough accuracy and trueness estimates. The numerical values are listed in table 3.1 and

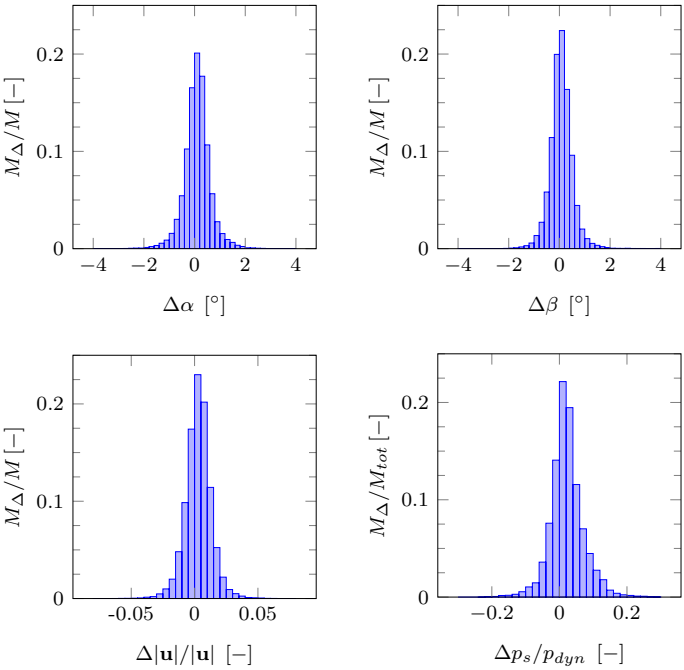


Figure 3.12: Error histograms

Table 3.1: Performance of the tested five-hole probe

	$\Delta\alpha$ [°]	$\Delta\beta$ [°]	$\frac{\Delta\|\mathbf{u}\|}{\|\mathbf{u}\|}$ [–]	$\frac{\Delta p_s}{p_{dyn}}$ [–]
Accuracy: RMSE	5.2e-1	4.8e-1	1.0e-2	5.1e-2
Trueness: ME	5.0e-3	-3.0e-2	3.9e-4	1.2e-2

the normalized distributions are shown as histograms in figure 3.12. While these results provide an indication of the probe's performance, they are not fully conclusive since:

- the tests were conducted at a constant, low turbulence level, and
- there is a risk that the computed measures ($RMSE$ and ME) are dominated by uncertainties in the reference flow properties.

3.3 Tracking accuracy

The use of a five-hole probe in ProCap requires that the position and orientation of the probe are known at every point in time when measurements are taken. Although the method is not limited to a particular tracking system, often optical, marker-based trackers provide the best choice due to their outstanding performance (low latency, high spatial and temporal resolution). Tracking accuracy plays an important role in improving the overall measurement quality. To this end, this section contains a brief assessment of the positional and orientational accuracy of a probe being tracked by a marker-based camera system. In principle, the theory given below is applicable for all tracking systems but elaborated here on the basis of the used Qualisys system with the Oqus3 cameras.

3.3.1 Tracking error of a single marker

When an optical tracking system is used, the positional accuracy of a single marker is highly dependent on a number of factors including number cameras, geometrical setting, lighting conditions etc.. Below, there is a list of the most important errors:

- **Calibration errors** are caused by inaccurate scaling of the three room axes during the calibration.
- **Occlusion errors** arise from objects that partly occlude the cameras' line of sight to a marker.

- **Identification/correspondence errors** occur when the identification of two neighboring markers is mixed up or when by mistake a single marker is interpreted as two markers.
- **Thermal drift** may be present due to the heating of the cameras.
- **Quantization errors** result from the discretization of the image.
- **Dynamic errors** comprise uncertainties which are caused by the motion of the markers, e.g. positional error due to finite exposure time (elongated blob in the image plane) or due to imprecise camera synchronization.
- **Noise/Jitter**

Phenomenologically, these errors fall into two categories:

1. Systematic errors
2. Random errors

Usually, random errors can be well approximated by random variables with a Gaussian distribution. This simplification does normally not apply to systematic errors. However, *Bauer* (2007) demonstrated that because systematic errors arise from many different sources, they can be reduced to a single quasi-random error in the image plane with an isotropic, unbiased Gaussian distribution. This approximation, however, is only valid if the calibration errors are completely eliminated beforehand. *Bauer* (2007) suggested removing the calibration errors manually by assuming independent scaling of the three space axes. If data for such a correction is available, the error at every point in the region of interest can be estimated using the uncertainty propagation analysis described in *Borer* (2014) (chapter B.5). For the cameras used in this work, measurements show that the standard deviation of the marker errors in the image plane is approximately 1/64 pixel. Applying Borer's method, the shape of the resulting 2σ-ellipsoids in the measurement domain is then a function of the camera setup and the marker position only (cf. fig. 3.13). At the center of the covered domain the 2σ-ellipsoids become spheres

if the cameras are optimally arranged. In the ETH wind tunnel, where the working distance is approximately $2m$, the 2σ-confidence interval is on the order of $10\mu m$ provided that all four Oqus3 cameras capture the marker being analyzed. But as indicated before, this value is rather a measure for the precision than for the accuracy of the marker positions, as in most cases one has neither the tools nor the data or the time to eliminate the systematic errors introduced by the calibration process. More reliable accuracy measures are therefore:

- **Reconstruction error:** The reconstruction error is the shortest distance between a reconstructed 3D point and the ray associated with the projection of the measured 2D position in the image plane into the 3D space. Qualisys provides two related measures:

    1. The *average residual* is the mean of the reconstruction errors of all the points measured by a single camera during a calibration or measurement run. The cameras used in this work have an average residual which is typically smaller than $0.5mm$ for a working distance of about $2m$.

    2. The *3D residual* is computed by averaging the reconstruction errors of all rays that belong to the same 3D point. It is calculated in real-time for all reconstructed marker points. For a working distance of $2m$ and markers with a diameter of about $7mm$, the 3D residual is usually smaller than $0.5mm$. For better control Qualisys allows the user to select the maximum accepted 3D residual. A too small threshold value usually leads to poor measurement results as so-called ghost trajectories are likely to be produced. On the other hand, a too large value also results in defective trajectories as the triangulation algorithm attempts to merge data from different markers. It is recommended to set the maximum value 2-5 times larger than the average. $2mm$ proved to be a good threshold value for measurements in the ETH wind tunnel, where the cameras are about $2m$ away from the actual measurement domain.

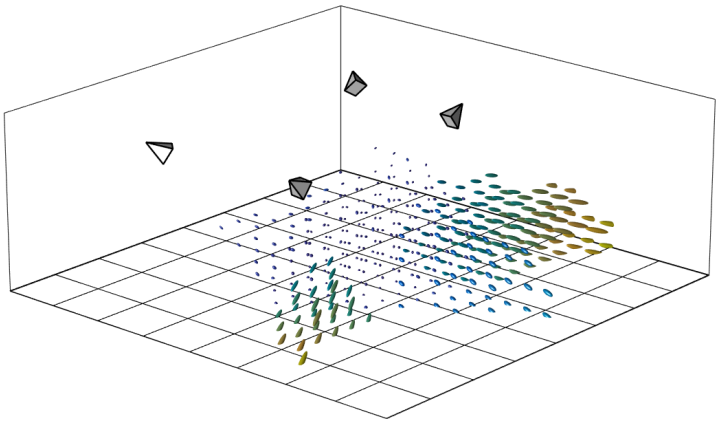


Figure 3.13: Illustration of the 2σ-confidence intervals for a given camera setup.

- **Reprojection error:** The reprojection error is the distance of the measured 2D position and the point obtained by the backward projection of the reconstructed 3D point on the image plane. Qualisys' tracking software does not provide the reprojection error directly, but on the basis of the recorded data[§] one can compute the reprojection error separately.

- **Root-mean-square error of a rigid two marker target:** The root-mean-square error of the measured distance between two markers belonging to the same rigid body provides useful information about the accuracy of the tracking system. In case the target is in motion, this measure does account for dynamic errors but ignores the fact that in reality, the error is also a function of space. If the camera system is calibrated using a wand, the root-mean-square error of the distance between the two markers is obtained without additional effort. Typically, this value is between $0.1 - 0.5mm$.

To sum up, the measures provided by the camera system used indicate that for most measurement scenarios the error of the trian-

[§]Qualisys provides both 2D and 3D marker data

gulated 3D position of a marker is smaller than $0.5mm$. Of course, this requires that the system is properly calibrated. Furthermore, if no additional correction is applied, the tracking accuracy is mainly dictated by the calibration errors (i.e. uncertainties in the extrinsic camera parameters) which are known to be systematic and cannot be modeled by an unbiased Gaussian distribution.

3.3.2 Tracking error of a rigid body

The aim of this subsection is to evaluate the influence of the tracking errors of the markers on the rotational and positional accuracy of the probe at hand. Determining the position and orientation of a rigid body on the basis of a few marker positions is described by the so-called orthogonal Procrustes problem:

Problem 1 (Orthogonal Procrustes problem)
Given are two ordered sets of N points in $\mathbb{R}^3$:

$$\mathcal{X} = \{\boldsymbol{x_i}\}_{i=1}^{N} \tag{3.34}$$

$$\mathcal{Y} = \{\boldsymbol{y_i}\}_{i=1}^{N} \tag{3.35}$$

Each of these sets represents the same rigid body but at a different location and with a different orientation (see figure 3.14). Find a rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ and a translation vector $\boldsymbol{t} \in \mathbb{R}^3$, such that the mapping

$$\boldsymbol{z_i} = \boldsymbol{R}\boldsymbol{x_i} + \boldsymbol{t}, \quad i = 1, \ldots, N \tag{3.36}$$

minimizes the mean squared error between $\mathcal{Y}$ and $\mathcal{Z} = \{\boldsymbol{z_i}\}_{i=1}^{N}$, i.e.

$$E = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{z_i} - \boldsymbol{y_i}\|^2 \rightarrow min. \tag{3.37}$$

The solution of this minimization problem is obtained in two steps:

(i) Find the optimal translation vector $\boldsymbol{t}_*$.

(ii) Find the optimal rotation matrix $\boldsymbol{R}_*$.

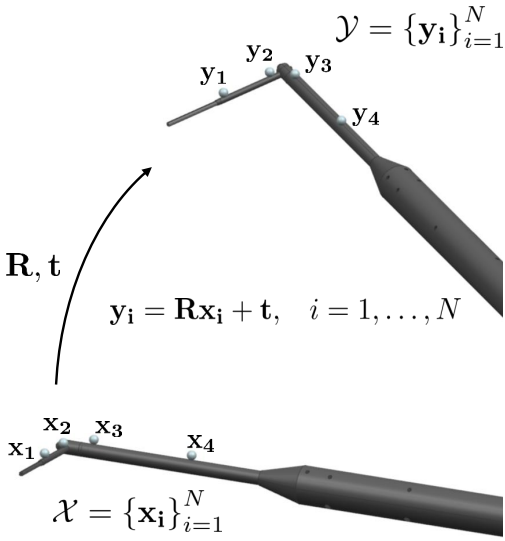


Figure 3.14: Orthogonal Procrustes problem

Derivation of the optimal translation vector

Substituting the mapping ansatz (3.36) into equation (3.37) yields

$$\begin{aligned} E &= \frac{1}{N}\sum_{i=1}^{N} \left(\boldsymbol{R}\boldsymbol{x_i} + \boldsymbol{t} - \boldsymbol{y_i}\right)^T \left(\boldsymbol{R}\boldsymbol{x_i} + \boldsymbol{t} - \boldsymbol{y_i}\right) \\ &= \frac{1}{N}\sum_{i=1}^{N} \left[\|\boldsymbol{t}\|^2 + 2\left(\boldsymbol{R}\boldsymbol{x_i} - \boldsymbol{y_i}\right)^T \boldsymbol{t} + \|\boldsymbol{R}\boldsymbol{x_i} - \boldsymbol{y_i}\|^2\right] \end{aligned} \tag{3.38}$$

Since this function is convex with respect to the translation vector $\boldsymbol{t}$, the optimum $\boldsymbol{t_*}$ is found as follows:

$$\left.\frac{\partial E}{\partial \boldsymbol{t}}\right|_{\boldsymbol{t}=\boldsymbol{t}_*} = \frac{1}{N}\sum_{i=1}^{N}\left[2\boldsymbol{t}_* + 2\left(\boldsymbol{R}_*\boldsymbol{x_i} - \boldsymbol{y_i}\right)\right] \stackrel{!}{=} \boldsymbol{0}$$

$$\Leftrightarrow \boldsymbol{t}_* = \underbrace{\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y_i}}_{=:\boldsymbol{y_c}} - \boldsymbol{R}_* \underbrace{\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x_i}}_{=:\boldsymbol{x_c}}$$

$$\Leftrightarrow \boldsymbol{t}_* = \boldsymbol{y_c} - \boldsymbol{R}_*\boldsymbol{x_c} \tag{3.39}$$

Here, $\boldsymbol{x_c}$ and $\boldsymbol{y_c}$ denote the positions of the centroid of the rigid body before and after the transformation. Equation (3.39) reveals that the optimal translation vector $\boldsymbol{t}_*$ depends on the optimal rotation matrix $\boldsymbol{R}_*$.

Derivation of the optimal rotation matrix

To find the optimal rotation matrix, the translation vector in equation (3.37) is replaced by its optimum (3.39):

$$E = \frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{R}\underbrace{\left(\boldsymbol{x_i} - \boldsymbol{x_c}\right)}_{=:\breve{\boldsymbol{x}}_i} - \underbrace{\left(\boldsymbol{y_i} - \boldsymbol{y_c}\right)}_{=:\breve{\boldsymbol{y}}_i}\right\|^2, \tag{3.40}$$

where $\breve{\boldsymbol{x}}_i$ and $\breve{\boldsymbol{y}}_i$ are the coordinates of the i-th marker relative to the respective body-centered coordinate system. The standard approach to solving this optimization problem is based on the singular value decomposition (SVD) of the cross-covariance matrix

$$\breve{\boldsymbol{C}} = \sum_{i=1}^{N}\breve{\boldsymbol{x}}_i\breve{\boldsymbol{y}}_i^{\,T} \tag{3.41}$$

and is commonly known as Kabsch algorithm (*Kabsch*, 1976). Here, we apply a slightly different algorithm which is based on the paper of

Coutsias et al. (2004). This alternative approach replaces the SVD of a non-symmetric 3×3-Matrix by an eigendecomposition (ED) of a symmetric 4×4-Matrix. While the results of these two algorithms are identical, the latter has a number of advantages:

- Our goal is to investigate how errors in the marker positions influence the orientational accuracy of the probe. Studying error propagation of a SVD compared to an ED is in general more difficult.

- Both rotations and reflections are represented by orthogonal matrices. For a reflection the determinant of this transformation matrix is -1, while for a rotation it is $+1$. In certain situations a reflection can yield a smaller mean squared error than a rotation. Because reflections have no meaning in the context of rigid body tracking, special attention has to be paid to exclude them from the space of possible solutions. In the author's opinion, the method of Coutsias provides a more natural way to omit reflections than the standard Kabsch algorithm.

In the following, the method of *Coutsias et al.* (2004) is briefly outlined. As quaternions play a central role in the derivation of this method, the interested reader is referred to appendix A, where one can find a summary of the basic concepts of quaternions. It is straightforward to show that in quaternion notation equation (3.40) can be expressed as follows

$$E = \frac{1}{N} \sum_{i=1}^{N} \left[\left(q_q \, x_{q,i} \overline{q}_q - y_{q,i} \right) \overline{\left(q_q \, x_{q,i} \overline{q}_q - y_{q,i} \right)} \right]_0 . \tag{3.42}$$

Here, the bar indicates the conjugation of a quaternion. Furthermore, the subscript q labels an element as a quaternion, while the subscript 0 stands for the scalar component of a quaternion. Moreover, $x_{q,i}$ and $y_{q,i}$ denote the pure quaternions of the shifted marker positions, i.e.

$$x_{q,i} = (0, \breve{\boldsymbol{x}}_{\boldsymbol{i}}) \quad \text{and} \quad y_{q,i} = (0, \breve{\boldsymbol{y}}_{\boldsymbol{i}}) \tag{3.43}$$

The quaternion q_q has unit length and represents a rotation around an axis $\check{\boldsymbol{c}}$ and with an angle ϕ, i.e.

$$q_q = \left(q_0, [q_1, q_2, q_3]^T\right) = \left(\cos\frac{\phi}{2}, \check{\boldsymbol{c}}\sin\frac{\phi}{2}\right) \tag{3.44}$$

Using the distributive property of quaternions, the formula of the mean squared error can be rearranged in the following manner

$$\begin{aligned} E &= \frac{1}{N}\sum_{i=1}^{N}\Big[\left(q_q x_{q,i}\overline{q}_q\right)\overline{\left(q_q x_{q,i}\overline{q}_q\right)} + y_{q,i}\overline{y}_{q,i} - \left(q_q x_{q,i}\overline{q}_q\right)\overline{y}_{q,i} \\ &\qquad -y_{q,i}\overline{\left(q_q x_{q,i}\overline{q}_q\right)}\Big]_0 \\ &= \frac{1}{N}\sum_{i=1}^{N}\Big[x_{q,i}\overline{x}_{q,i} + y_{q,i}\overline{y}_{q,i} - \left(q_q x_{q,i}\overline{q}_q\right)\overline{y}_{q,i} - y_{q,i}\overline{\left(q_q x_{q,i}\overline{q}_q\right)}\Big]_0 \\ &= \frac{1}{N}\sum_{i=1}^{N}\left\{\|\check{\boldsymbol{x}}_{\boldsymbol{i}}\|^2 + \|\check{\boldsymbol{y}}_{\boldsymbol{i}}\|^2 + 2\underbrace{\left[y_{q,i}q_q x_{q,i}\overline{q}_q\right]_0}_{=:P}\right\} \end{aligned} \tag{3.45}$$

For the remaining part of this derivation, the quaternions are expressed as column vectors with four elements, i.e.

$$\boldsymbol{a} = [a_q]_v = [a_0, a_1, a_2, a_3]^T \text{ with } a_q = (a_0, [a_1, a_2, a_3]^T), \tag{3.46}$$

where the subscript v implies that the quaternion is interpreted as a column vector. In appendix A, it is shown that in vector notation the multiplication of two quaternions, e.g. $u_q v_q$, can either be expressed by

$$\boldsymbol{A}_L\left(u_q\right)\boldsymbol{v} \quad \text{or} \quad \boldsymbol{A}_R\left(v_q\right)\boldsymbol{u}, \tag{3.47}$$

where $\boldsymbol{A}_L$ and $\boldsymbol{A}_R$ are defined as in formula (A.5). With the help of these conventions, the term P in equation (3.45) can be modified

in the following way

$$\begin{aligned} P &= \left[y_{q,i} q_q x_{q,i} \overline{q}_q \right]_0 \\ &= \boldsymbol{q}^T \left[y_{q,i} q_q x_{q,i} \right]_v \\ &= \boldsymbol{q}^T \boldsymbol{A}_L \left(y_{q,i} \right) \boldsymbol{A}_R \left(x_{q,i} \right) \boldsymbol{q}. \end{aligned} \tag{3.48}$$

Hence, the mean squared error can be written as follows

$$E = \frac{1}{N} \sum_{i=1}^{N} \left\{ \left\| \breve{\boldsymbol{x}}_{\boldsymbol{i}} \right\|^2 + \left\| \breve{\boldsymbol{y}}_{\boldsymbol{i}} \right\|^2 \right\} - \frac{2}{N} \boldsymbol{q}^T \boldsymbol{M} \boldsymbol{q} \tag{3.49}$$

where the matrix $\boldsymbol{M}$ is defined as

$$\boldsymbol{M} = - \sum_{i=1}^{N} \boldsymbol{A}_L \left(y_{q,i} \right) \boldsymbol{A}_R \left(x_{q,i} \right). \tag{3.50}$$

$\boldsymbol{M}$ is a symmetric matrix and can be expressed in terms of the cross-covariance matrix $\breve{\boldsymbol{C}}$ (defined in (3.41)):

$$\boldsymbol{M} = \begin{bmatrix} \breve{C}_{11} + \breve{C}_{22} + \breve{C}_{33} & \breve{C}_{23} - \breve{C}_{32} & \breve{C}_{31} - \breve{C}_{13} & \breve{C}_{12} - \breve{C}_{21} \\ \breve{C}_{23} - \breve{C}_{32} & \breve{C}_{11} - \breve{C}_{22} - \breve{C}_{33} & \breve{C}_{12} + \breve{C}_{21} & \breve{C}_{13} + \breve{C}_{31} \\ \breve{C}_{31} - \breve{C}_{13} & \breve{C}_{12} + \breve{C}_{21} & \breve{C}_{22} - \breve{C}_{11} - \breve{C}_{33} & \breve{C}_{23} + \breve{C}_{32} \\ \breve{C}_{12} - \breve{C}_{21} & \breve{C}_{13} + \breve{C}_{31} & \breve{C}_{23} + \breve{C}_{32} & \breve{C}_{33} - \breve{C}_{11} - \breve{C}_{22} \end{bmatrix} \tag{3.51}$$

Revisiting the minimization problem, the optimal rotation quaternion $\boldsymbol{q_*}$ is found by minimizing the function E (given by eq. (3.49)). Since only the last term depends on $\boldsymbol{q}$ the minimization problem is equivalent to the following maximization problem:

$$\boldsymbol{q}^T \boldsymbol{M} \boldsymbol{q} \rightarrow \max_{\|\boldsymbol{q}\|=1} \tag{3.52}$$

Since $\boldsymbol{M}$ is symmetric, the Courant-Fischer Min-Max theorem states that the maximum is the largest eigenvalue of $\boldsymbol{M}$. In another way of saying, the optimal rotation quaternion $\boldsymbol{q}_*$ is equal to the normalized eigenvector that belongs to the largest eigenvalue of $\boldsymbol{M}$.

Error-free case

Before we focus on the error propagation properties of the algorithm described before, it is important to examine the case where the measured points $\mathcal{Y}$ are assumed to be exact. This implies that there exists a rotation matrix $\boldsymbol{R}$ and a translation vector $\boldsymbol{t}$ such that the mean squared error E as defined in eq. (3.36) is zero. Without loss of generality, the orthogonal Procrustes problem can be expressed in terms of the principle axes of the point set $\mathcal{X}$. More precisely, because the auto-covariance matrix

$$\breve{\boldsymbol{A}} = \sum_{i=1}^{N} \breve{\boldsymbol{x}}_{\boldsymbol{i}} \breve{\boldsymbol{x}}_{\boldsymbol{i}}{}^{T} \tag{3.53}$$

is real, symmetric and positive definite, the eigenvalues $\mu_1 \leq \mu_2 \leq \mu_3$ are both non-negative and real. Therefore, the eigenspace of $\breve{\boldsymbol{A}}$ can be used to construct an orthonormal basis $\{\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3}\}$ (a.k.a. principle axes) for which the auto-covariance matrix $\breve{\boldsymbol{A}}$ becomes diagonal. If the eigenvalues are mutually different, $\{\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3}\}$ are simply the normalized eigenvectors. Hence, the shifted marker positions expressed in the principle axes coordinate system are given by

$$\breve{\boldsymbol{x}}'_{\boldsymbol{i}} = \boldsymbol{S}^T \breve{\boldsymbol{x}}_{\boldsymbol{i}} \tag{3.54}$$

$$\breve{\boldsymbol{y}}'_{\boldsymbol{i}} = \boldsymbol{S}^T \breve{\boldsymbol{y}}_{\boldsymbol{i}} \tag{3.55}$$

where

$$\boldsymbol{S} = [\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3}] \tag{3.56}$$

Here and in the following, we use the prime to indicate that the vector is expressed in the principle axes coordinate system. Similarly, $\boldsymbol{R'}$ and q'_q symbolize the rotation matrix and the rotation quaternion with respect to the principle axes coordinate system, i.e.

$$\boldsymbol{R'} = \boldsymbol{S}^T \boldsymbol{R} \boldsymbol{S} \quad \text{and} \quad q'_q = \left(q'_0, [q'_1, q'_2, q'_3]^T\right) = \left(q_0, \boldsymbol{S}^T [q_1, q_2, q_3]^T\right)$$

Since we assume that the measurements $\mathcal{Y}$ are exact, the cross-covariance matrix $\check{\boldsymbol{C}}'$ has the following form

$$\check{\boldsymbol{C}}' = \sum_{i=1}^{N} \check{\boldsymbol{x}}_i' \check{\boldsymbol{y}}_i'^T = \underbrace{\sum_{i=1}^{N} \check{\boldsymbol{x}}_i' \check{\boldsymbol{x}}_i'^T}_{=\check{\boldsymbol{A}}'} \boldsymbol{R}'^T = \operatorname{diag}(\mu_1, \mu_2, \mu_3)\, \boldsymbol{R}'^T. \quad (3.57)$$

Now, let $q_q' = (q_0', q_1', q_2', q_3')$ be the rotation quaternion that transforms $\check{\boldsymbol{x}}_i'$ to $\check{\boldsymbol{y}}_i'$, then it is easy to verify that the eigenvalue decomposition of $\boldsymbol{M}'$ yields:

Eigenvalues:	Eigenvectors:
$\lambda_1 = \mu_1 + \mu_2 + \mu_3$	$\boldsymbol{q_1'} = [q_0', q_1', q_2', q_3']^T$
$\lambda_2 = \mu_1 - \mu_2 - \mu_3$	$\boldsymbol{q_2'} = [q_1', -q_0', -q_3', q_2']^T$
$\lambda_3 = \mu_2 - \mu_1 - \mu_3$	$\boldsymbol{q_3'} = [q_2', q_3', -q_0', -q_1']^T$
$\lambda_4 = \mu_3 - \mu_1 - \mu_2$	$\boldsymbol{q_4'} = [q_3', -q_2', q_1', -q_0']^T$

(3.58)

In accordance with the theory, the eigenvector with the largest eigenvalue, that is $\boldsymbol{q_1'}$, does indeed correspond to the rotation quaternion. Now, we have all the tools to study how the measurement errors affect the solution of the orthogonal Procrustes problem.

Linear perturbation analysis

Suppose the measurements have some degree of uncertainty that stems from the tracking of the markers (see figure 3.15):

$$\tilde{\boldsymbol{y}}_i' = \boldsymbol{y_i'} + \boldsymbol{\Delta y_i'}, \quad i = 1, \ldots, N \quad (3.59)$$

For the sake of generality, we do not further specify the positional errors of the markers $\boldsymbol{\Delta y_i'}$, except that they are assumed to be small in magnitude. Naturally, these positional errors also give rise to errors in the estimate of the rotation quaternion

$$\tilde{\boldsymbol{q}}_1' = \boldsymbol{q_1'} + \boldsymbol{\Delta q_1'} \quad (3.60)$$

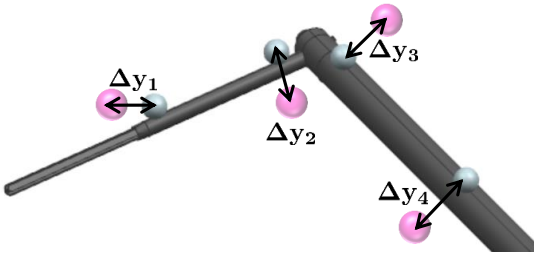


Figure 3.15: Marker position errors

Following the method outlined in appendix B, the first order approximation of the eigenvector error $\boldsymbol{\Delta q'_1}$ is

$$\boldsymbol{\Delta q'_1} = \sum_{i=1}^{3} \epsilon_i \boldsymbol{q'_{i+1}} \tag{3.61}$$

with

$$\epsilon_1 = \frac{\boldsymbol{q'_2}^T \boldsymbol{\Delta M}\, \boldsymbol{q'_1}}{2\mu_2 + 2\mu_3}, \quad \epsilon_2 = \frac{\boldsymbol{q'_3}^T \boldsymbol{\Delta M}\, \boldsymbol{q'_1}}{2\mu_1 + 2\mu_3}, \quad \epsilon_3 = \frac{\boldsymbol{q'_4}^T \boldsymbol{\Delta M}\, \boldsymbol{q'_1}}{2\mu_1 + 2\mu_2}. \tag{3.62}$$

As before, $\left\{\boldsymbol{q'_i}\right\}_{i=1}^{4}$ represent the eigenvectors of the undisturbed rotation scenario, while $\{\mu_i\}_{i=1}^{3}$ are the eigenvalues of the undisturbed cross-correlation matrix $\boldsymbol{\breve{C}'}$. Moreover, $\boldsymbol{\Delta M}$ is simply the deviation of the matrix $\boldsymbol{M}$ caused by the positional errors of the markers. In quaternion notation the disturbed eigenvector $\boldsymbol{\tilde{q}'_1}$ can also be expressed by a concatenation of two rotations, namely the true rotation given by q_q and a rotation s_q caused by the measurement errors. In mathematical terms, this means:

$$\boldsymbol{\tilde{q}'_1} = \boldsymbol{q'_1} + \boldsymbol{\Delta q'_1} \quad \hat{=} \quad \tilde{q}_q = s_q q_q, \tag{3.63}$$

where s_q is defined as follows

$$s_q = (\cos(\Delta\phi/2), \sin(\Delta\phi/2)\boldsymbol{p}) \tag{3.64}$$

As $\Delta\phi$ is assumed to be small, s_q can be linearized

$$s_q \approx \left(1, \frac{\Delta\phi}{2}\boldsymbol{p}\right) \tag{3.65}$$

Substituting this approximation into equation (3.63) yields

$$[s_q q_q]_v = \boldsymbol{q_1'} + \underbrace{\frac{\Delta\phi}{2}\begin{bmatrix} -[q_1', q_2', q_3']\,\boldsymbol{p} \\ q_0'\boldsymbol{p} + \boldsymbol{p} \times [q_1', q_2', q_3']^T \end{bmatrix}}_{=:\boldsymbol{\Delta q_1'}}. \tag{3.66}$$

The subscript v indicates that the quaternion is expressed as a vector. One can show that if

$$\frac{\Delta\phi}{2}\boldsymbol{p} = \boldsymbol{R'}\,[\epsilon_1, \epsilon_2, \epsilon_3] \tag{3.67}$$

the second term on the right-hand side of equation (3.66) conforms to the previously derived expression (3.61). Now, by taking the norm of equation (3.67) and using the two facts that

- the multiplication of $\boldsymbol{R'}$ with a vector preserves the length of the vector, and that
- the length of $\boldsymbol{p}$ is 1,

we finally find the first order approximation of the angular error:

$$\boxed{|\Delta\phi| \approx 2\sqrt{\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2}} \tag{3.68}$$

Remarks:

- To avoid large angle errors, the markers, which define the body, shall be placed such that at least two eigenvalues of the auto-covariance matrix $\breve{\boldsymbol{A}}$ are much larger than zero. This ensures that the denominators of ϵ_1, ϵ_2 and ϵ_3 are far from being zero.
- This error estimation allows us to study two special scenarios: The worst case and the random error scenario. Both will be discussed below.

Worst case scenario

First, we study the case, where the positional errors $\boldsymbol{\Delta y_i}$ of the markers remain unspecified except that their magnitudes are smaller than a certain threshold $|\Delta \boldsymbol{y}|_{max}$. Based on that, we aim to find the marker configuration which produces the largest angle error. It is clear that for every rigid body rotation there exists a similar worst case that returns the exactly same orientation error. Therefore, without loss of generality we can assume that

$$q_q = (1, 0, 0, 0) \tag{3.69}$$

which simplifies eq. (3.62) as follows

$$\begin{aligned} \epsilon_1 &= \frac{-\Delta M_{12}\left(\boldsymbol{\Delta y_1}, \ldots, \boldsymbol{\Delta y_N}\right)}{2\mu_2 + 2\mu_3}, \\ \epsilon_2 &= \frac{-\Delta M_{13}\left(\boldsymbol{\Delta y_1}, \ldots, \boldsymbol{\Delta y_N}\right)}{2\mu_1 + 2\mu_3}, \\ \epsilon_3 &= \frac{-\Delta M_{14}\left(\boldsymbol{\Delta y_1}, \ldots, \boldsymbol{\Delta y_N}\right)}{2\mu_1 + 2\mu_2}. \end{aligned} \tag{3.70}$$

Thus, in order to find the largest angle error, we have to solve the

Table 3.2: Marker positions of the tested five-hole probe

	x $[mm]$	y $[mm]$	z $[mm]$
Probe tip	0.00	0.00	0.00
Marker 1	40.60	0.00	5.50
Marker 2	75.36	0.00	5.50
Marker 3	89.07	−45.18	8.50
Marker 4	87.85	−113.59	8.50

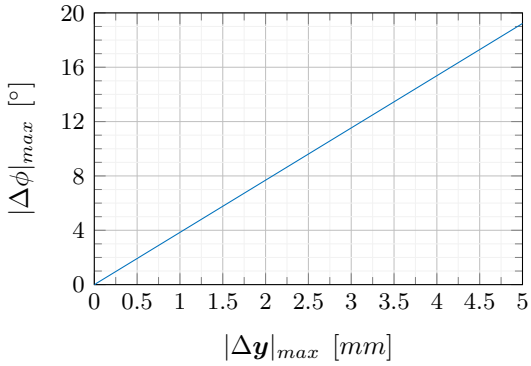


Figure 3.16: Maximum angle error as a function of the maximum positional error $|\boldsymbol{\Delta y}|_{max}$

following constrained minimization problem:

$$-\left[\frac{\Delta M_{12}^2}{(\mu_2+\mu_3)^2}+\frac{\Delta M_{13}^2}{(\mu_1+\mu_3)^2}+\frac{\Delta M_{14}^2}{(\mu_1+\mu_2)^2}\right]\rightarrow \min_{\boldsymbol{\Delta y_1},\ldots,\boldsymbol{\Delta y_N}} \tag{3.71}$$

$$\text{subject to } |\boldsymbol{\Delta y_i}| \leq |\boldsymbol{\Delta y}|_{max}, \quad i=1,\ldots,N$$

Due to the non-linear constraints, no analytic solution could be found. However, the problem can be solved numerically using Matlab's *fmincon* function. The marker positions for the five-hole probe used in this work are listed in table 3.2. The solution to this problem is depicted in figure 3.16. It is found that the maximum angle error $|\Delta\phi|_{max}$ increases linearly with the maximum positional error at a rate of approximately $3.8°/mm$.

With the help of this result, we can easily define an upper bound for the positional error of the probe tip. In ProCap, the position of the tip $\boldsymbol{y_{pt}}$ is computed on the basis of the nearest marker $\boldsymbol{y_1}$:

$$\boldsymbol{y_{pt}} = \boldsymbol{y_1} + \boldsymbol{R}\underbrace{(\boldsymbol{x_{pt}} - \boldsymbol{x_1})}_{=:\boldsymbol{l}} \tag{3.72}$$

From this equation we can deduce the maximum positional error of the tip

$$\begin{aligned} |\boldsymbol{\Delta y_{pt}}| &\leq |\boldsymbol{\Delta y}|_{max} + 2|\boldsymbol{l}|\sin\left(\frac{|\Delta\phi|_{max}}{2}\right) \\ &\approx |\boldsymbol{\Delta y}|_{max} + |\boldsymbol{l}|\,|\Delta\phi|_{max} \end{aligned} \tag{3.73}$$

With $|\boldsymbol{l}| = 40.2mm$, the maximum error of the probe tip position can be estimated by the following formula:

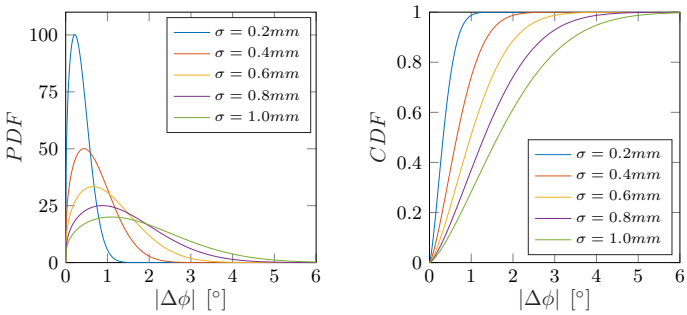
$$|\boldsymbol{\Delta y_{pt}}| \approx 3.7|\boldsymbol{\Delta y}|_{max} \tag{3.74}$$

From section 3.3.1 we know that for the measurements presented in this work the maximum error of a marker is estimated to be smaller than $0.5mm$. Larger errors which may occur due to an erroneous correspondence search or marker occlusion are eliminated by the software considering the 3D-residual tolerance and the bone length tolerance. The bone length tolerance is a user-selected threshold value which limits the maximum variation of the distance between all possible marker pairs of a rigid body. Concluding this section, we can say that for the presented measurements the tracking accuracy of the five-hole probe is better than $1.9mm$ in position and $1.9°$ in orientation.

Random error scenario

In this case we assume that the positional errors of the markers can be modeled by independent random vectors having a multivariate Gaussian distribution with zero mean. Naturally, this assumption is only valid in the absence of calibration errors. Furthermore, we assume that the cameras are optimally placed, which allows us to specify the positional error distribution:

$$\boldsymbol{\Delta y_i} \sim \mathcal{N}\left(\boldsymbol{0}, \sigma^2\boldsymbol{I}\right), \quad i = 1, \ldots, N \tag{3.75}$$



(a) PDF of the angle error **(b)** CDF of the angle error

Figure 3.17: PDF and CDF of the angle error depending on σ

This error model turns the formulas for ϵ_1, ϵ_2 and ϵ_3 (see eq. (3.62)) into the following distributions

$$\epsilon_1 \sim \mathcal{N}\left(0, \frac{\sigma^2}{4(\mu_2+\mu_3)}\right), \tag{3.76}$$

$$\epsilon_2 \sim \mathcal{N}\left(0, \frac{\sigma^2}{4(\mu_1+\mu_3)}\right), \tag{3.77}$$

$$\epsilon_3 \sim \mathcal{N}\left(0, \frac{\sigma^2}{4(\mu_1+\mu_2)}\right). \tag{3.78}$$

Revisiting equation (3.68), the distribution of the angle error $|\Delta\phi|$ can be approximated by means of the Welch-Satterthwaite equation, i.e.

$$|\Delta\phi| \sim \text{Nakagami}\left(\widetilde{m}, \widetilde{\Omega}\right) \tag{3.79}$$

with

$$\widetilde{m} = \frac{0.5\left(\frac{1}{\mu_1+\mu_2} + \frac{1}{\mu_1+\mu_3} + \frac{1}{\mu_2+\mu_3}\right)^2}{\left(\frac{1}{\mu_1+\mu_2}\right)^2 + \left(\frac{1}{\mu_1+\mu_3}\right)^2 + \left(\frac{1}{\mu_2+\mu_3}\right)^2} \tag{3.80}$$

and

$$\widetilde{\Omega} = \sigma^2\left(\frac{1}{\mu_1+\mu_2} + \frac{1}{\mu_1+\mu_3} + \frac{1}{\mu_2+\mu_3}\right) \tag{3.81}$$

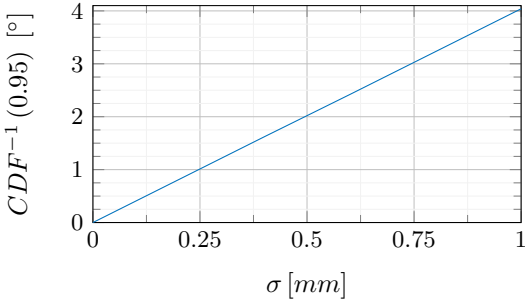


Figure 3.18: 95th precentile of the angle error as a function of σ

Based on the marker positions given in table 3.2, figure 3.17 plots the estimated probability density function (PDF) and the cumulative density function (CDF) for different noise levels $\sigma = 0.2 - 1.0$. As expected, a larger noise level raises the risk of large orientational errors. This is also confirmed by figure 3.18 which indicates a linear increase of the 95th precentile with the standard deviation of the marker positions σ. Last but not least, it is worth noting that the presented approximation of the angle error distribution may also be valid for cases where the spatial variations of the calibration errors are small considering the spread of the markers on the probe. Since the main part of the calibration error arises from the uncertainties of the extrinsic camera parameters, this is likely to be the case.

3.4 Corrections for the probe motion

The way ProCap works, the pressure distribution on the surface of the probe does not only depend on the flow to be studied but also on the motion of the probe itself. This is particularly true at low flow speeds. Phenomenologically, three distinct effects can be identified:

- **Linear motion:** An imaginary observer sitting on a moving probe registers a wind speed that differs from the flow velocity measured by a stationary observer. The flow velocity seen by the probe is essentially the superposition of the true flow

velocity and the probe speed.

- **Rotary motion:** By the laws of physics, a rotary motion of a body with finite size induces a fluid flow on its surface. This implies that the pressure readings of a five-hole probe are biased if the probe undergoes a change of orientation.

- **Acceleration effects:** If the probe speed is changed rapidly (e.g. abrupt movements), the time-stationarity assumption, on which the calibration of the probe is based, is violated. In other words, terms that account for the temporal changes of the apparent flow become important and cannot be neglected.

Because in most cases the probe movements are regular and relatively slow, the last two effects can be considered to have little impact on the pressure measurements. As the probe position is tracked in time, the translational motion of the probe can be filtered out easily from the measurements. Figure 3.19 gives an overview of the correction scheme implemented in ProCap. It involves the following steps:

❶ **Estimation of the probe velocity:** Based on the tracked positional data of the markers, one can deduce the position and velocity of the probe tip at every frame instant. Calculating the velocity by a first order backward difference scheme proved to be insufficient as the measurement errors in the positional data are amplified. Currently, ProCap retrieves the velocity by applying a 1D-filter to each component of the discrete position-signal. The shape of this filter is defined by the derivative of a Gaussian filter[¶]. This approach causes a delay of half the filter width.[‖] For augmented reality and virtual reality applications, it is recommended to keep the time lag smaller than $15ms$ to provide a realistic experience. To this end, the software is designed such that the introduced time lag does only affect the velocity estimate but not the position of the virtual probe. An alternative, more elegant way of estimating the velocity

[¶] cf. differentiation property of convolutions

[‖] When measuring with $100Hz$, a filter width of 20 frames has proven to be a good choice.

of the probe might be the use of a Kalman filter. For time reasons, this approach is not yet included.

❷ **Determining the apparent flow quantities:** Based on the pressure measurements the instantaneous flow quantities (i.e. velocity, static and total pressure) relative to the probe coordinate system are computed using the previously determined calibration maps. The breve accent symbolizes that components of the velocity vector are based on the probe coordinate system. The subscript a stands for "apparent" and indicates that the measured quantity corresponds to what is seen by the probe.

❸ **Transformation to global coordinate system:** The apparent flow velocity is transformed to the global coordinate system using the corresponding tracking data of the probe.

❹ **Correction:** Superposition of the apparent flow speed and the probe velocity reveals the true flow velocity at the point of measurement. In addition, the measured stagnation pressure is also biased by the movement of the probe. However, knowing the true velocity, the correction is straightforward.

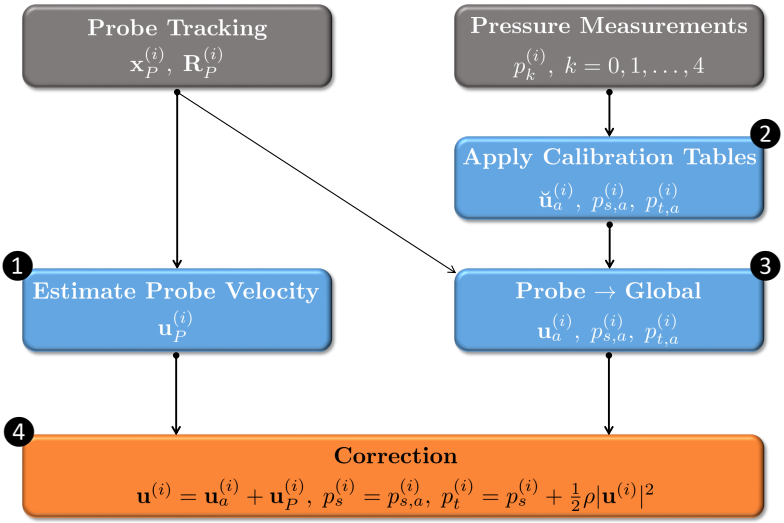


Figure 3.19: Velocity correction scheme.

Chapter 4

Flow field reconstruction

The reconstruction of the flow field onto a regular grid based on measured data points is undoubtedly a key task of the ProCap measurement system. This becomes particularly evident when considering that most real-time flow visualization features implemented in ProCap (e.g. arrow glyphs, streamlines, contour planes) rely on the reconstructed flow field. This chapter provides a detailed exposition of the reconstruction problem encountered in ProCap. In the first section, the general scattered data approximation problem is introduced. Section 4.2 briefly sketches the particular challenges to the reconstruction process imposed by the ProCap measurement system. With respect to these challenges, the most common meshfree approximation techniques are compared. The preferred method (Moving least squares) is introduced in section 4.3. Subsequent to that, the performance of two moving least squares subvariants are demonstrated by means of a simple example (sec. 4.4). In section 4.5 we address the complex question of how to select the bandwidth of the weight function by reviewing two different approaches. This is followed by a detailed description of the method's implementation into ProCap (sec. 4.6). The attention is directed at the data reduction step, the bandwidth selection and the implementation on the graphics processing unit (GPU). The final section of this chapter is devoted to a physically based extension of the approximation technique, the so-called pseudo-divergence-free moving least squares method.

4.1 The scattered data approximation problem

The objective of scattered data approximation is to find an estimate $\widehat{u}$ of an unknown function $u : \mathbb{R}^d \to \mathbb{R}$ on the basis of a few samples $\mathcal{U} = \{u_i\}_{i=1}^{M}$. These samples represent the function values at M

distinct, usually unorganized data sites $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{M}$. If no errors are present, one often aims at finding an estimate that reproduces the given data exactly, i.e.

$$\widehat{u}\,(\boldsymbol{x}_i) = u_i, \quad i = 1, \ldots, M \tag{4.1}$$

Such problems are referred to as scattered data interpolation. On the other hand, if the sampled data or the data sites contain errors, an exact interpolation often leads to an inaccurate reproduction of the unknown function (e.g. overfitting). To overcome this issue, the interpolation requirement (4.1) is relaxed in favor of other criteria, e.g. continuity, smoothness, etc.. This second type of problem is commonly referred to as scattered data approximation.

Scattered data approximation problems arise in different fields of science and engineering, such as

- *Computer Graphics*, e.g. surface reconstruction from point clouds, texture mapping
- *Tomography and medical imaging*, e.g. tomographic reconstruction
- *Geophysics*, e.g. terrain modeling, mapping and predictions
- *Statistics*, e.g. data regression
- *Computational science and engineering*, e.g. meshless methods for solving partial differential equations

As a result of that fact, the vast majority of papers devoted to scattered data approximation is published in subject-specific journals leading to the somewhat irritating circumstance that similar concepts/methods are known by many different names. On the other hand, multidisciplinary reviews of scattered data interpolation and approximation methods are relatively rare considering the frequency of occurrence of the problem. Noteworthy studies on this topic are *Franke* (1982), *Alfeld* (1989), *Franke and Nielson* (1991), *Mitas and Mitasova* (1999) and more recently *Fasshauer* (2007).

4.2 Scattered data approximation in ProCap

At first glance, the flow field reconstruction in ProCap is similar to any other scattered data approximation problem, which means that accuracy is the primary and often sole focus. A closer examination, however, reveals several additional challenges that need to be addressed accordingly:

- **Performance/real-time capability:** The user adapts the scanning path according to the displayed visualization of the flow field. Therefore, a continuous and instantaneous update of the reconstruction is essential to ensure the efficiency of the method.

- **Dynamic and large dataset:** A potential difficulty arises from the fact that during the measurement the number of data points is continuously growing. Regarding the real-time capability of the method, this fact poses special requirements on the data structure and on the approximation technique. In contrast, the standard approximation scenario presumes that all sampling data is known a priori. Therefore, it is no big surprise that little attention has been paid to approximation problems with a dynamically changing dataset so far.

- **Multiple output/scalability:** Conventional approximation techniques focus on reproducing scalar-valued functions only. In our method, the number of measured quantities and by that the number of functions to be approximated may be considerably larger than one. It depends on the probe used for the measurements. From a computational point of view, treating each of these quantities independently is very inefficient. Hence, an algorithm that approximates more than one function with as little additional work as possible is preferred.

- **Error handling:** The errors in the ProCap measurement data have different sources. Three classes of errors may be distinguished: Position error (tracking), measurement error (probe-based) and error due to the instationary flow field (physics). This distinction should not be understood as a clear-cut. For

instance, in the case of a direction-sensitive probe, the tracking is likely to influence the measurement error of the probe, too. The error usually varies not only between different measurements but also in time and space. Due to this complex character, the approximation scheme should be able to cope with all sorts of errors.

- **Data distribution and robustness:** As a result of the scanning approach, the data sites are located along a continuous path, i.e. the points are highly irregularly distributed. It is known that the geometric configuration of the data sites strongly affects the stability and accuracy of the approximation method. For instance, consider the 2D scenario, where data sites lie on a single line. Fitting a plane to these points is not unique as the slope perpendicular to the line is undefined. These points form a so-called degenerated point configuration for this kind of approximation. The chance to encounter this type of problem usually increases with the approximation order of the method and with the dimensionality of the domain.

- **Physical constraints:** One possibility to increase the reliability and consistency of the reconstructed field is to incorporate physical properties of the flow into the approximation scheme. For instance, incompressible velocity fields are known to be divergence-free. Therefore, the use of a method allowing only solenoidal vector fields may improve the overall reconstruction quality considerably.

Scattered data interpolation and approximation methods can be divided into two categories: Mesh-free and mesh-based methods. Examples of the latter are finite elements, wavelets, box splines, multivariate splines etc. Mesh-based scattered data methods work well if the data sites are moderately non-uniform, but often fail to cope with highly irregular data sites. As a consequence, these techniques have been discarded from the list of candidates. Truly mesh-free techniques, on the other hand, are generally more flexible regarding the non-uniformity of the data sites. The most common mesh-free

Table 4.1: Evaluation of meshless approximation methods (☼ good, ⛅ average, 🌧 poor)

	Accuracy	RT-capability	Dynamic data	Scalability	Robustness	Physics
Moving least squares	⛅	☼	☼	☼	⛅	⛅
Radial basis functions	☼	🌧	⛅	🌧	⛅	☼
Partition of unity	⛅	☼	🌧	☼	⛅	☼

techniques are radial basis functions*, moving least squares† and partition of unity. Among these three techniques, moving least squares is found to be the most appropriate technique with respect to the requirements mentioned above. The overview of the evaluation results is given in table 4.1.

Although radial basis functions are known to be very accurate, there are mainly two issues that are in conflict with ProCap:

1.) The method is global, i.e. each time a new sample arrives a large linear system has to be solved. An iterative solver, using the old solution as initial guess, may reduce the computational costs significantly. But even in that case, the number of arithmetic operations required exceeds the 'real-time' limits of a regular PC. In addition to that, also almost all visualization elements (e.g. isosurfaces) require being updated in the entire domain since the approximation changes globally.

2.) To get the best approximation within the radial basis function framework, one has to determine the optimal shape parameter of these radial functions. Unfortunately, applying the theo-

*a.k.a. Kriging, Gaussian process regression or Wiener-Kolmogorov prediction

†a.k.a. local polynomial regression or normalized convolution

retical optimum often results in an ill-conditioned and dense system of linear equations. So in practice, it comes down to a compromise between accuracy and numerical stability (a.k.a. accuracy vs. stability trade-off).

Regarding ProCap, partition of unity seems to be the best alternative to moving least squares. Besides the fact that there is only a limited amount of literature on the subject, the main question to be answered is how to choose the subdomains. There are essentially two options of how partition of unity can handle an input data set that grows in time:

1.) The division of the subdomains is temporally adjusted, e.g. by insertion of a new subdomain. However, in this case, the local character of the approximation scheme is diminished.

2.) At the cost of numerical stability, the approximation order in the affected subdomains is step-wise increased.

In the remainder of this chapter, moving least squares will be explained in more detail with special focus on the ProCap requirements and on the adaptions made. For the other two mesh-free approximation techniques the interested reader is referred to Appendix C or to more comprehensive sources (e.g. *Fasshauer* (2007), *Wendland* (2004)).

4.3 Moving least squares

Moving least squares (MLS) is a powerful multivariate approximation technique, especially when the data sites are non-uniformly distributed and the data values are noisy. In approximation theory, the method was first introduced by *Lancaster and Salkauskas* (1981) as a generalization of the famous Shepard's approximation method (*Shepard*, 1968). In statistics, MLS is better known as local polynomial regression (LPR), a nowadays well-established non-parametric regression technique. While first applications of LPR can be traced back as far as the end of the 19th century, the mathematical foundation was not laid before the late 1970s (*Stone*, 1977; *Cleveland*,

1979). In other fields such as image processing, MLS is occasionally called normalized convolution (NC). NC was first proposed by *Knutsson and Westin* (1993) as a method for interpolation and filtering of incomplete and uncertain image data.

MLS differs from most other approximation and interpolation techniques insofar as the global problem is split into a series of smaller local subproblems. In practical terms, instead of one large linear system one has to solve many small linear systems. Originally, this property was considered as a disadvantage because the resulting total number of arithmetic operations required to solve the global problem is rather high. But, with the emergence of massively parallel computing units this argument may no longer be true. As demonstrated below, the MLS approximation problem is inherently parallel and therefore it can be solved very efficiently. Regarding ProCap, a local change of the input data does not mean that the complete approximation needs to be recomputed. Due to the local character of the approximation scheme, only a few subproblems have to be updated.

Since *Bos and Salkauskas* (1989) showed that MLS is Backus-Gilbert optimal, there exist two equivalent formulations of the MLS approximation method: The standard formulation and the Backus-Gilbert formulation. Because both approaches are crucial for the understanding of MLS, they are briefly outlined in the two subsequent sections.

4.3.1 Standard formulation

As mentioned above, the key idea of MLS is to decompose the global curve fitting problem into a series of smaller local approximation problems. This is achieved in the following way: Around every evaluation point $\boldsymbol{x}_0$ the unknown function $u(\boldsymbol{x})$ is approximated by a linear combination of $N+1$ basis functions

$$\widehat{u}(\boldsymbol{x}, \boldsymbol{x}_0) = \sum_{j=0}^{N} c_j(\boldsymbol{x}_0)\, b_j(\boldsymbol{x} - \boldsymbol{x}_0)\,. \tag{4.2}$$

Here, $\{b_j\}_{j=0}^{N}$ denote the set of basis functions and $\{c_j\}_{j=0}^{N}$ are the coefficients to be determined.
Remarks:

- In principle, the linear approximation space $\mathcal{B} = \text{span}\{b_0, ..., b_N\}$ can be defined by any set of linearly independent basis functions. For smooth functions, low order polynomials proved to be a reasonable choice. Hereafter, we restrict our analysis to polynomials of degree m. For simplicity, the basis functions are assumed to be the corresponding monomials.

- In the formula above, the basis functions are centered around the evaluation point $\boldsymbol{x}_0$. This coordinate shift is not mandatory but strongly recommended as it improves the numerical stability of the resulting least squares problem (*Mirzaei et al.*, 2011).

- The coefficients $\{c_j\}_{j=0}^{N}$ are not constant but vary with the evaluation point $\boldsymbol{x}_0$. Thus, every time the evaluation point $\boldsymbol{x}_0$ is changed, another but similar approximation problem has to be solved. As the method's name suggests, the global approximation of the unknown function is obtained by moving the evaluation point to every location where the approximation is sought. Commonly, the evaluation points are arranged in a regular grid.

In order to measure the quality of the local, linear approximation ansatz, a weighted semi-inner product and its induced norm are introduced

$$\langle u, v \rangle_{w(\cdot,\boldsymbol{x}_0)} = \sum_{i=1}^{M} u(\boldsymbol{x}_i)\, v(\boldsymbol{x}_i)\, w(\boldsymbol{x}_i, \boldsymbol{x}_0), \tag{4.3}$$

$$\|u\|^2_{w(\cdot,\boldsymbol{x}_0)} = \langle u, u \rangle_{w(\cdot,\boldsymbol{x}_0)} = \sum_{i=1}^{M} u^2(\boldsymbol{x}_i)\, w(\boldsymbol{x}_i, \boldsymbol{x}_0), \tag{4.4}$$

in which $w(\boldsymbol{x}, \boldsymbol{x}_0)$ acts as a weight function. Typically, $w(\boldsymbol{x}, \boldsymbol{x}_0)$ is non-negative for all $\boldsymbol{x} \in \mathbb{R}^d$, has a maximum at $\boldsymbol{x}_0$ and decays monotonically as the distance $\|\boldsymbol{x} - \boldsymbol{x}_0\|$ goes to infinity. For simplicity,

Table 4.2: List of radial weight functions; χ is the indicator function

	$\phi(r)$
Inverse distance	$r^{-\beta}, \quad \beta \in \mathbb{N}$
Gaussian	e^{-r^2}
Inverse quadratic	$\left(1+r^2\right)^{-1}$
Inverse multiquadratic	$\left(1+r^2\right)^{-1/2}$
Epanechnikov	$\left(1-r^2\right)\chi_{[0,1]}(r)$
C^2-Wendland function	$(1-r)^4(4r+1)\chi_{[0,1]}(r)$
Cosine	$\cos^\beta\left(\frac{\pi}{2}r\right)\chi_{[0,1]}(r), \quad \beta \in \mathbb{N}$

we further assume w to be radial, i.e.

$$w(\boldsymbol{x}, \boldsymbol{x}_0) = \phi\left(\frac{\|\boldsymbol{x}-\boldsymbol{x}_0\|}{h(\boldsymbol{x}_0)}\right), \tag{4.5}$$

where $h(\boldsymbol{x}_0)$ is the so-called shape parameter of w. It can be modeled as a function of the evaluation point $\boldsymbol{x}_0$. To improve the quality of the reconstructed function, some authors (*Pham et al.*, 2006; *Ruppert and Wand*, 1994; *Liu*, 2001) suggested applying non-radial weight functions. The local shape and orientation of these weight functions depend on the point distribution and on the nature of the underlying function. Since these models are computationally expensive, they do not conform to the real-time requirement of ProCap. As shown below, in the case of radial weight functions, the approximation order is not affected by the choice of ϕ as long as $h(\boldsymbol{x}_0)$ scales with the local fill distance. However, from statistics it is known that the asymptotically optimal weight function is the so-called Epanechnikov kernel as it minimizes the asymptotic mean squared error (*Fan and Gijbels*, 1996). Some often employed radial weight functions are listed in table 4.2.

Returning to the MLS problem, the unknown coefficients $\{c_j\}_{j=0}^{N}$ of the local approximant (4.2) are found by minimizing the residual function

$$r(\boldsymbol{x}, \boldsymbol{x}_0) = u(\boldsymbol{x}) - \widehat{u}(\boldsymbol{x}, \boldsymbol{x}_0) \tag{4.6}$$

with respect to the previously introduced norm, i.e.

$$\|r(\boldsymbol{x}, \boldsymbol{x}_0)\|^2_{w(\cdot, \boldsymbol{x}_0)} \to \min. \tag{4.7}$$

The normal equation of this weighted least squares problem has the form

$$\boldsymbol{G}(\boldsymbol{x}_0)\, \boldsymbol{c}(\boldsymbol{x}_0) = \boldsymbol{v}(\boldsymbol{x}_0) \tag{4.8}$$

with

$$\begin{aligned} &\boldsymbol{G}_{i,j} = \langle b_i(\boldsymbol{x} - \boldsymbol{x}_0), b_j(\boldsymbol{x} - \boldsymbol{x}_0) \rangle_{w(\cdot, \boldsymbol{x}_0)}, \quad i, j = 0, \ldots, N \\ &\boldsymbol{c} = [c_0(\boldsymbol{x}_0), \ldots, c_N(\boldsymbol{x}_0)]^T \\ &\boldsymbol{v}_i = \langle u(\boldsymbol{x}), b_i(\boldsymbol{x} - \boldsymbol{x}_0) \rangle_{w(\cdot, \boldsymbol{x}_0)}, \quad i = 0, \ldots, N. \end{aligned} \tag{4.9}$$

Alternatively, the right-hand-side vector $\boldsymbol{v}$ and the Gram matrix $\boldsymbol{G}$ can be written in terms of the Vandermonde matrix

$$\boldsymbol{B}_{i,j+1} = b_j(\boldsymbol{x}_i - \boldsymbol{x}_0), \quad i = 1, \ldots, M, \quad j = 0, \ldots, N \tag{4.10}$$

and the weight matrix

$$\boldsymbol{W} = \operatorname{diag}[w(\boldsymbol{x}_1, \boldsymbol{x}_0), \ldots, w(\boldsymbol{x}_M, \boldsymbol{x}_0)] \tag{4.11}$$

in the following way

$$\boldsymbol{G} = \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{B} \tag{4.12}$$

$$\boldsymbol{v} = \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{u} \tag{4.13}$$

where

$$\boldsymbol{u} = [u_1, \ldots, u_M]^T. \tag{4.14}$$

From a computational point of view, one only has to account for the set of data sites located in the support of the weight function $w(\boldsymbol{x}, \boldsymbol{x}_0)$, i.e. $\mathcal{X}_* = \mathcal{X} \cap \operatorname{supp}[w(\boldsymbol{x}, \boldsymbol{x}_0)]$. By definition, the Gram

matrix $\boldsymbol{G}$ strongly depends on the sampling point distribution around $\boldsymbol{x}_0$. An inappropriate combination of the approximation space $\mathcal{B}$ and the weight function w (e.g. shape parameter h or support) leads to an ill-conditioned normal equation. This issue can be diminished by solving the error equation directly

$$\sqrt{\boldsymbol{W}}\boldsymbol{B}\boldsymbol{c} = \sqrt{\boldsymbol{W}}\boldsymbol{u} \tag{4.15}$$

using either a QR or SVD decomposition. In the worst case, however, the rank of the Matrix $\boldsymbol{B}$ is smaller than $N+1$, meaning that the weighted least squares problem (4.7) is not well-posed. Assuming that $\mathcal{B}$ spans the polynomial space of degree m, the set of data sites $\mathcal{X}_*$ is then said to be not m-unisolvent. Consequently, to guarantee uniqueness of the MLS solution, a weak constraint on the data site distribution has to be imposed: For every evaluation point $\boldsymbol{x}_0$ the set of data sites located in the support of the corresponding weight function $w(\boldsymbol{x}, \boldsymbol{x}_0)$ has to be m-unisolvent. From the Haar-Curtis theorem it becomes clear, that for $m > 0$, this is not directly linked to the number of distinct data sites.

The final step of the standard MLS algorithm is simply the evaluation of the linear approximation ansatz (4.2) at $\boldsymbol{x} = \boldsymbol{x}_0$, i.e.

$$\widehat{u}(\boldsymbol{x}_0, \boldsymbol{x}_0) = \sum_{j=0}^{N} c_j(\boldsymbol{x}_0)\, b_j(\boldsymbol{0}) \tag{4.16}$$

As we assume monomial basis functions, the formula simplifies to

$$\widehat{u}(\boldsymbol{x}_0) = c_0(\boldsymbol{x}_0). \tag{4.17}$$

An advantage of the MLS method lies in the possibility to estimate not only the function value u but also its spatial derivatives (multi-index notation):

$$\widehat{\partial^{\boldsymbol{\alpha}} u}(\boldsymbol{x}_0, \boldsymbol{x}_0) = \sum_{j=0}^{N} c_j(\boldsymbol{x}_0)\, \partial^{\boldsymbol{\alpha}} b(\boldsymbol{0}), \quad \boldsymbol{\alpha} \in \mathbb{N}_0^d \tag{4.18}$$

Clearly, this requires the polynomial degree m of the approximation

space to be larger than the order of the derivative $|\boldsymbol{\alpha}|$. It is important to point out that these estimates are not identical to the true derivatives of the approximation $\widehat{u}$, i.e. $\widehat{\partial^{\boldsymbol{\alpha}} u} \neq \partial^{\boldsymbol{\alpha}} \widehat{u}$. However, they possess the same approximation order. In the literature, these derivative estimates are ususally referred to as *diffuse* derivatives. They play a key role in the diffuse element method (DEM), a meshless technique to solve partial differential equations first proposed by *Nayroles et al.* (1992). In section 4.5, the approximation quality of these diffuse derivatives is analyzed.

Below, the algorithm of the standard MLS method is summarized:

Standard MLS

Input: data sites $\{\boldsymbol{x}_i\}_{i=1}^{M}$, data values $\{u_i\}_{i=1}^{M}$
Output: approximation values $\{\widehat{u}_k\}_{k=1}^{R}$
begin
  define basis functions $\boldsymbol{b}(\boldsymbol{x}) = [b_0(\boldsymbol{x}), \ldots, b_N(\boldsymbol{x})]^T$
  define weight function $\phi(r)$
  define evaluation sites $\{\boldsymbol{x}_{0,k}\}_{k=1}^{R}$
  for $k \leftarrow 1$ **to** R **do**
    determine optimal shape parameter h_k
    find $\mathcal{X}_* = \mathcal{X} \cap \operatorname{supp}\left[\phi\left(\frac{\boldsymbol{x}-\boldsymbol{x}_{0,k}}{h}\right)\right]$
    initialize $\boldsymbol{G} = \boldsymbol{0}$ and $\boldsymbol{u} = \boldsymbol{0}$
    foreach $\boldsymbol{x}_i \in \mathcal{X}_*$ **do**
      $w \leftarrow \phi\left(\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_{0,k}\|}{h_k}\right)$
      $\boldsymbol{G} \leftarrow \boldsymbol{G} + w\, \boldsymbol{b}(\boldsymbol{x}_i - \boldsymbol{x}_{0,k})\, \boldsymbol{b}^T(\boldsymbol{x}_i - \boldsymbol{x}_{0,k})$
      $\boldsymbol{u} \leftarrow \boldsymbol{u} + w\, u_i\, \boldsymbol{b}(\boldsymbol{x}_i - \boldsymbol{x}_{0,k})$
    end
    $\boldsymbol{c}(\boldsymbol{x}_{0,k}) \leftarrow \boldsymbol{G}^{-1}\boldsymbol{u}$
    $\widehat{u}_k \leftarrow \boldsymbol{b}^T(\boldsymbol{0})\, \boldsymbol{c}(\boldsymbol{x}_{0,k})$
  end
end

4.3.2 Backus-Gilbert formulation

In the late 1960s, two geophysicists, George Backus and Freeman Gilbert, developed an elegant method to solve ill-posed inverse problems, nowadays known as Backus-Gilbert method (*Backus and Gilbert* (1967), *Backus and Gilbert* (1968), *Backus and Gilbert* (1970)). Two decades later, *Bos and Salkauskas* (1989) showed that the Backus-Gilbert method and MLS are closely related by reformulating the MLS method. This section is devoted to this dual formulation beginning with a short introduction of the Backus-Gilbert method for general linear inverse problems. The second part focuses on scattered data approximation only. We will complete this section by demonstrating that the standard MLS and the Backus-Gilbert formulation are indeed identical.

The Backus-Gilbert method is only applicable if the known data is given in terms of M linear functionals $\{\mu_i(u)\}_{i=1}^{M}$. Note, in the case of scattered data approximation these are simply the point-evaluation functionals. Now, the objective of the Backus-Gilbert method is to find an estimate of another linear functional $\mu(u)$. Due to linear nature of this problem, a simple, linear approximation ansatz is proposed

$$\widehat{\mu}(u) = \sum_{i=1}^{M} \mu_i(u)\, \psi_i(\mu)\,. \tag{4.19}$$

Since the values $\{\mu_i(u)\}_{i=1}^{M}$ are known, all that remains to be determined are the coefficients $\{\psi_i(\mu)\}_{i=1}^{M}$. To this end, a function $q(\mu_i, \mu) \geq 0$ is introduced that acts as a separation measure. Broadly speaking, the separation measure should follow the rule: the more similar two functionals (arguments), the smaller is the function value q and vice versa. In the case of point-evaluation functionals, q is usually selected such that it only depends on the two points, where the function u is evaluated. In other words, the further apart the two points are the larger is the corresponding function value q. According to Backus and Gilbert, the optimal approximation of μ is obtained by minimizing the following quadratic form (spread

functional)

$$S\left(\psi_i, \ldots, \psi_M\right) = \frac{1}{2} \sum_{i=1}^{M} \left[\psi_i\left(\mu\right)\right]^2 q\left(\mu_i, \mu\right) \tag{4.20}$$

Obviously, by solely minimizing this functional we obtain the trivial solution $\psi_i = 0,\ i = 1, \ldots, M$. To prevent this from happening, an additional constraint is required. Backus and Gilbert stated this constraint as follows: If u belongs to a certain finite function space $\mathcal{B} = \text{span}\left\{b_0, ..., b_N\right\}$, the approximation method $\widehat{\mu}\left(u\right)$ has to be exact. Due to linearity this condition implies $N + 1$ reproduction constraints:

$$\sum_{i=1}^{M} \mu_i\left(b_j\right) \psi_i\left(\mu\right) = \mu\left(b_j\right), \quad j = 0, \ldots, N. \tag{4.21}$$

To sum up, the resulting constrained minimization problem reads as follows:

$$\begin{gathered} \tfrac{1}{2} \boldsymbol{\psi}^T \boldsymbol{Q} \boldsymbol{\psi} \rightarrow \min \\ \text{subject to } \boldsymbol{B}^T \boldsymbol{\psi} = \boldsymbol{b} \end{gathered} \tag{4.22}$$

where

$$\begin{aligned} &\boldsymbol{Q} = \text{diag}\left[q\left(\mu_1, \mu\right), \ldots, q\left(\mu_M, \mu\right)\right] \\ &\boldsymbol{\psi} = \left[\psi_1\left(\mu\right), \ldots, \psi_M\left(\mu\right)\right]^T \\ &\boldsymbol{B}_{i,j+1} = \mu_i\left(b_j\right), \quad i = 1, \ldots, M \text{ and } j = 0, \ldots, N \\ &\boldsymbol{b} = \left[\mu\left(b_0\right), \ldots, \mu\left(b_N\right)\right]^T. \end{aligned} \tag{4.23}$$

By introducing $N+1$ Lagrange multipliers $\boldsymbol{\lambda} = \left[\lambda_0\left(\mu\right), \ldots, \lambda_N\left(\mu\right)\right]^T$ the following linear system can be deduced (Euler-Lagrange equations)

$$\begin{bmatrix} \boldsymbol{Q} & -\boldsymbol{B} \\ \boldsymbol{B}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{b} \end{bmatrix}. \tag{4.24}$$

The upper part of this linear system can be rearranged

$$\boldsymbol{\psi} = \boldsymbol{Q}^{-1} \boldsymbol{B} \boldsymbol{\lambda}. \tag{4.25}$$

Substituting this expression into the lower part of equation (4.24), we obtain a decoupled system for the Lagrange multipliers

$$\boldsymbol{B}^T\boldsymbol{Q}^{-1}\boldsymbol{B}\boldsymbol{\lambda} = \boldsymbol{b}. \tag{4.26}$$

In a nutshell, the Backus-Gilbert method comprises three steps:

1. Compute $\boldsymbol{\lambda}$ by solving system (4.26)
2. Compute the coefficients $\boldsymbol{\psi}$ using equation (4.25)
3. Compute the approximation $\widehat{\mu}\,(\mu)$ by inserting $\boldsymbol{\psi}$ into the approximation ansatz (4.19)

By adapting the Backus-Gilbert technique to the scattered data approximation problem, the given functionals $\{\mu_i\}_{i=1}^M$ are simply the point-evaluation functionals, i.e.

$$\mu_i\,(u) = u\,(\boldsymbol{x}_i) = u_i, \quad i = 1, \ldots, M. \tag{4.27}$$

In order to attain a two-variable formulation, the functional μ is considered as the point-evaluation functional at a point $\boldsymbol{x}$, which lies in the very near proximity of the point $\boldsymbol{x}_0$. Thus, the approximation ansatz (4.19) can be rephrased as a local quasi-approximant:

$$\widehat{u}\,(\boldsymbol{x}, \boldsymbol{x}_0) = \sum_{i=1}^{M} u_i \psi_i\,(\boldsymbol{x}, \boldsymbol{x}_0)\,, \tag{4.28}$$

where $\{\psi_i\,(\boldsymbol{x}, \boldsymbol{x}_0)\}_{i=1}^M$ are known to be the generating functions of the quasi-approximant. The actual approximation at $\boldsymbol{x}_0$ is obtained by setting $\boldsymbol{x}$ equal $\boldsymbol{x}_0$:

$$\widehat{u}\,(\boldsymbol{x}_0, \boldsymbol{x}_0) = \sum_{i=1}^{M} u_i \psi_i\,(\boldsymbol{x}_0, \boldsymbol{x}_0)\,, \tag{4.29}$$

Furthermore, the separation measure $q\,(\mu_i, \mu)$ is simplified such that it only depends on $\boldsymbol{x}_i$ and $\boldsymbol{x}_0$. This approximation is justified by the fact that $\boldsymbol{x}$ is known to be very close to $\boldsymbol{x}_0$. In mathematical terms this means that $q\,(\mu_i, \mu)$ can be written as $q\,(\boldsymbol{x}_i, \boldsymbol{x}_0)$.

Without loss of generality and to improve numerical robustness, the reconstruction constraints are expressed with respect to the shifted basis functions

$$\sum_{i=1}^{M} b_j\left(\boldsymbol{x}_i - \boldsymbol{x}_0\right)\psi_i\left(\boldsymbol{x}, \boldsymbol{x}_0\right) = b_j\left(\boldsymbol{x} - \boldsymbol{x}_0\right), \quad j = 0, \ldots, N. \tag{4.30}$$

Finally, the unknown generating functions $\{\psi_i\left(\boldsymbol{x}, \boldsymbol{x}_0\right)\}_{i=1}^{M}$ are found by solving equation (4.26) and (4.25), where, however, the terms and coefficients possess a much simpler form than in the general case:

$$\begin{aligned}
&\boldsymbol{Q} = \operatorname{diag}\left[q\left(\boldsymbol{x}_1, \boldsymbol{x}_0\right), \ldots, q\left(\boldsymbol{x}_M, \boldsymbol{x}_0\right)\right] \\
&\boldsymbol{\psi} = \left[\psi_1\left(\boldsymbol{x}, \boldsymbol{x}_0\right), \ldots, \psi_M\left(\boldsymbol{x}, \boldsymbol{x}_0\right)\right]^T \\
&\boldsymbol{B}_{i,j+1} = b_j\left(\boldsymbol{x}_i - \boldsymbol{x}_0\right), \quad i = 1, \ldots, M \text{ and } j = 0, \ldots, N \\
&\boldsymbol{b} = \left[b_0\left(\boldsymbol{x} - \boldsymbol{x}_0\right), \ldots, b_N\left(\boldsymbol{x} - \boldsymbol{x}_0\right)\right]^T \\
&\boldsymbol{\lambda} = \left[\lambda_0\left(\boldsymbol{x}, \boldsymbol{x_0}\right), \ldots, \lambda_N\left(\boldsymbol{x}, \boldsymbol{x_0}\right)\right]^T
\end{aligned} \tag{4.31}$$

To show the equivalence of the Backus-Gilbert and the standard MLS approach, two conditions have to be fulfilled:

(i) The weight function in the standard approach is the inverse of the separation measure in the Backus-Gilbert approach, i.e. $w\left(\boldsymbol{x}, \boldsymbol{x}_0\right) = 1/q\left(\boldsymbol{x}, \boldsymbol{x}_0\right)$

(ii) The approximation space $\mathcal{B}$ in the standard approach is identical to the reproduction function space of the Backus-Gilbert approach.

Due to the diagonality of $\boldsymbol{Q}$ and $\boldsymbol{W}$, the first condition yields

$$\boldsymbol{Q}^{-1} = \boldsymbol{W}. \tag{4.32}$$

Thus, equation (4.25) and (4.26) can be reformulated as follows

$$\boldsymbol{\psi} = \boldsymbol{W}\boldsymbol{B}\boldsymbol{\lambda} \tag{4.33}$$

$$\boldsymbol{B}^T\boldsymbol{W}\boldsymbol{B}\boldsymbol{\lambda} = \boldsymbol{b} \tag{4.34}$$

Now, by combining these two equations we get an explict formula

for the generating functions only

$$\boldsymbol{\psi} = \boldsymbol{W}\boldsymbol{B}\left[\boldsymbol{B}^T\boldsymbol{W}\boldsymbol{B}\right]^{-1}\boldsymbol{b}. \tag{4.35}$$

Inserting this expression into the quasi-approximation ansatz leads to:

$$\widehat{u} = \boldsymbol{\psi}^T\boldsymbol{u} = \boldsymbol{b}^T\left[\boldsymbol{B}^T\boldsymbol{W}\boldsymbol{B}\right]^{-T}\boldsymbol{B}^T\boldsymbol{W}\boldsymbol{u}, \tag{4.36}$$

where $\boldsymbol{u}$ is defined as in (4.14). Note that the matrix product in the squared brackets is the Gram matrix $\boldsymbol{G}$ as defined in (4.12). Moreover, as the Gram matrix is symmetric, the transpose operation has no effect. According to definition (4.13) the second factor in the above formula can be replaced by $\boldsymbol{v}$. These trivial modifications in conjunction with equation (4.8) lead to the following expression:

$$\widehat{u} = \boldsymbol{b}^T\boldsymbol{G}^{-1}\boldsymbol{v} = \boldsymbol{b}^T\boldsymbol{c}. \tag{4.37}$$

Evidently, this is the vector notation of the approximation ansatz in the standard MLS formulation and by that, we have shown that the standard and the Backus-Gilbert MLS formulations are indeed identical.

4.4 Examples

In this section, the two most simple subvariants of the MLS method are deduced, discussed and compared. These are the zeroth-order and the first-order MLS method. To illustrate and compare the performance of the two methods a simple one-dimensional approximation problem is solved. The function to be approximated is given by

$$u(x) = \cos(x)\sin(4x) + 0.1\cos(15x). \tag{4.38}$$

One hundred data sites are selected as realizations of a random variable with a uniform distribution in the interval $[0, \pi]$. In the first case the samples are the exact function values, while in the second case, the samples are perturbed by Gaussian noise with a standard deviation of 0.1. A Gaussian kernel is used as weight function with

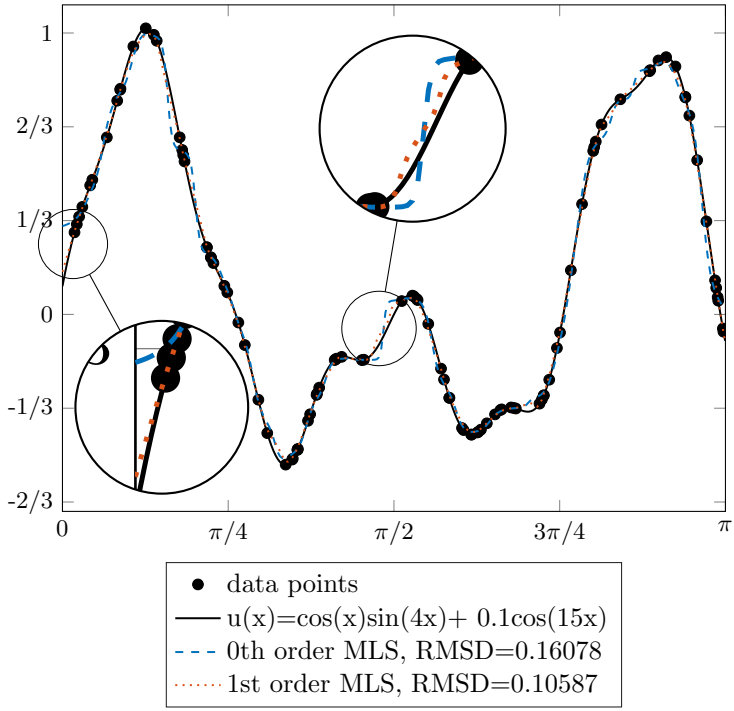


Figure 4.1: Comparison of 0th-order and 1st-order MLS using a Gaussian kernel with a globally fixed shape parameter ($h = 0.05$). The function values given at the data points are exact.

a globally fixed shape parameter h. In the error-free case, h is set equal 0.05, while in the disturbed case $h = 0.1$. The results are depicted in figures 4.1 and 4.2.

4.4.1 Zeroth-order MLS

In the simplest case, the unknown function is locally approximated by a constant. Thus, the finite approximation space is trivial and consists only of one single constant. Without loss of generality,

this constant is usually set equal to one. Because the Gram matrix consists of only one element

$$\boldsymbol{G} = \sum_{i=1}^{M} w\left(\boldsymbol{x}_i, \boldsymbol{x}_0\right), \tag{4.39}$$

the inversion and thus the computation of the coefficient $c_0\left(\boldsymbol{x}_0\right)$ becomes trivial. The formula of zeroth-order MLS approximation is

$$\widehat{u}\left(\boldsymbol{x}_0\right) = \sum_{i=1}^{M} u_i \frac{w\left(\boldsymbol{x}_i, \boldsymbol{x}_0\right)}{\sum\limits_{j=1}^{M} w\left(\boldsymbol{x}_j, \boldsymbol{x}_0\right)}. \tag{4.40}$$

From this formula, one can easily deduce explicit expressions for the generating functions

$$\psi_i\left(\boldsymbol{x}, \boldsymbol{x}_0\right) = \frac{w\left(\boldsymbol{x}_i, \boldsymbol{x}_0\right)}{\sum\limits_{j=1}^{M} w\left(\boldsymbol{x}_j, \boldsymbol{x}_0\right)}, \quad i = 1, \ldots, M \tag{4.41}$$

The zeroth-order MLS method is also referred to as Shepard's approximation method (*Shepard*, 1968). Occasionally, if w corresponds to the inverse distance function (cf. 4.2) the name inverse distance weighting is used. In other fields such as statistics, the method is usually called the Nadaraya-Watson estimator.

The main advantage of the zeroth-order MLS is its simplicity. As no linear system has to be solved, the computational costs are low compared to higher order methods. In addition to that, the method is known to be very robust. If the support of the weight function contains one or more data points the approximant is well-defined. Furthermore, since the generating functions are strictly positive, it is clear that the approximation is bounded by the smallest and largest data value in the support of the corresponding weight function (Min-Max property). This implies that the zeroth-order MLS has very poor extrapolation properties or leads to unsatisfactory results at locations where the data sites are highly non-uniformly distributed and the gradient of u is large (staircase effect). These two observations are highlighted in figure 4.1. The staircase effect is clo-

sely related to the selection of the shape parameter. It turns out that if the shape of w is too narrow relative to the local data site distribution, the approximation becomes very uneven (amplification of the staircase effect). Vice versa, a too wide shape of w leads to the unwanted effect of a very flat approximant (oversmoothing). Obviously, there is an optimum between the two extremes. In the case of noisy data, one takes advantage of the smoothing effect: by selecting a larger shape parameter, overfitting can be successfully avoided. The optimal shape parameter selection is covered in more detail in the next section.

For a better understanding of the flow topology, not only the distributions of the primary[‡] but also of derived quantities[§] are of interest. To determine these derived quantities one is often confronted with the task of computing spatial derivatives. Unfortunately, the zeroth-order MLS provides no diffuse derivatives and therefore, additional effort has to be put into the calculation of such derivative estimates.

4.4.2 First-order MLS

Now, we focus on the MLS method that exactly reproduces linear polynomials. Taking the monomials as basis functions, the Gram matrix $\boldsymbol{G}$ simplifies to

$$\boldsymbol{G} = \sum_{i=1}^{M} w\left(\boldsymbol{x}_i, \boldsymbol{x}_0\right) \widehat{\boldsymbol{x}}_i \otimes \widehat{\boldsymbol{x}}_i, \quad \text{with } \widehat{\boldsymbol{x}}_i = \left[1, \boldsymbol{x}_i^T - \boldsymbol{x}_0^T\right]^T. \tag{4.42}$$

The same simplification applies to the right-hand side of equation (4.8) yielding

$$\boldsymbol{v} = \sum_{i=1}^{M} w\left(\boldsymbol{x}_i, \boldsymbol{x}_0\right) u_i \widehat{\boldsymbol{x}}_i. \tag{4.43}$$

If the dimension d is small, a direct inversion of the Gram matrix is computationally still feasible as G is only a $(d+1) \times (d+1)$ matrix.

[‡] pressure and velocity

[§] e.g. vorticity, shear rate, Q-criterion, λ_2-criterion

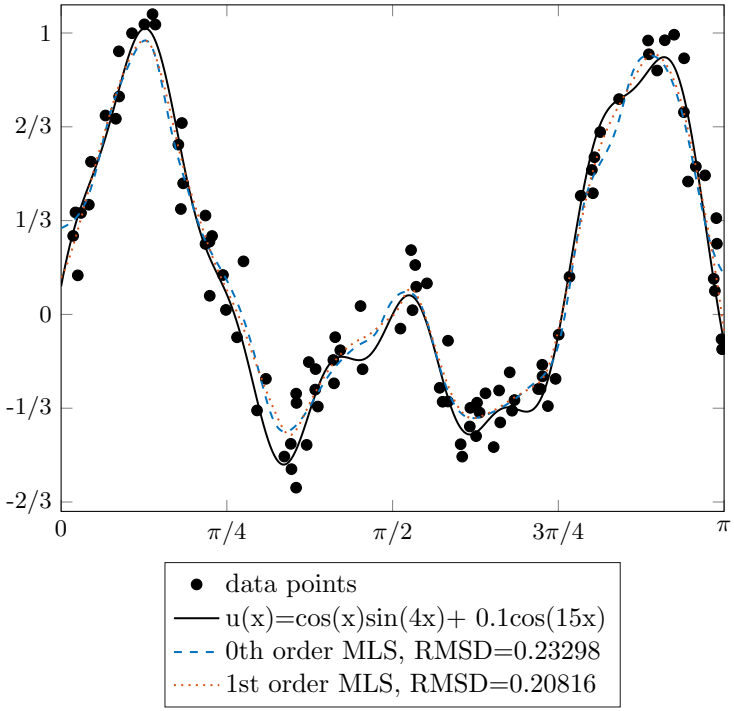


Figure 4.2: Comparison of 0th-order and 1st-order MLS using a Gaussian kernel with a globally fixed shape parameter ($h = 0.1$). The raw data is superimposed with Gaussian noise, $\sigma = 0.1$

Although numerically not ideal, one advantage of knowing the inverse of the Gram matrix explicitly is that the MLS approximation of a n-dimensional vector-valued function is obtained with relatively little computational effort. This is crucial for the scalability of ProCap, since the number of functions to be approximated onto the domain of interest depends on the chosen probe. For instance, a conventional five-hole probe measures four quantities: three velocity components and the static pressure.

A direct consequence of the monomial basis is that the first coefficient of the ansatz (4.2) c_0 is the approximation value at $\boldsymbol{x}_0$ itself,

while the other coefficients $c_1, \ldots, c_{d+1}$ represent the components of the diffuse gradient. Therefore, the first order MLS solution might be written as follows

$$\begin{bmatrix} \widehat{u}\,(\boldsymbol{x}_0) \\ \widehat{\partial_{x_1} u}\,(\boldsymbol{x}_0) \\ \vdots \\ \widehat{\partial_{x_d} u}\,(\boldsymbol{x}_0) \end{bmatrix} = \boldsymbol{G}^{-1}\boldsymbol{v}\,(\boldsymbol{x}_0)\,. \tag{4.44}$$

Figure 4.1 reveals that in general the first-order MLS approximation outperforms the zeroth-order counterpart in terms of accuracy. Especially, the extrapolation behavior at the boundaries is worth mentioning. Additionally, the staircase effect is barely visible. This means that the combination of highly non-uniform point set and large gradients is less problematic. Of course, the problem of highly non-uniform point distributions is still present, but only in regions with high curvature (Hessian). The observed improvements come at a cost:

- *Computational costs:* As mentioned above, the size of the Gram matrix is $(d+1) \times (d+1)$. Applying a Gauss-Jordan elimination scheme, the complexity of the matrix inversion is $\mathcal{O}((d+1)^3)$.
- *Robustness:* The first-order MLS approximation problem is only well-posed if at any point in the domain the Gram matrix is invertible, i.e. the point set in the support of every weight function w has to be at least 1-unisolvent. From Haar's theorem it is known that for $d > 1$ the condition of having more than d data sites in the support of w is not sufficient. Furthermore, as shown in the next section, unisolvence does not guarantee a good approximation. An unfavorable distribution of the data sites may lead to a very large approximation error despite unisolvency.

Besides that, the problem of choosing the weight function's shape parameter remains. It is even aggravated as the chance of facing a not well-posed or ill-conditioned approximation problem is increa-

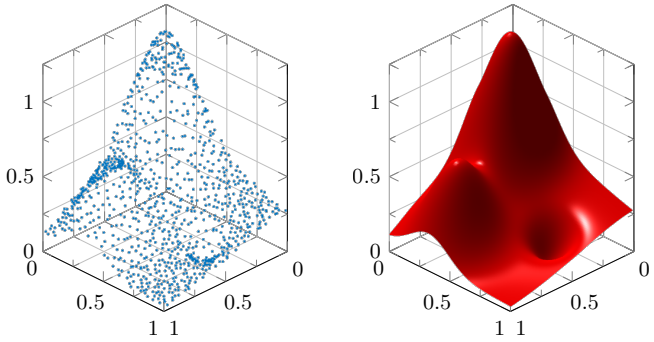
sed. In return, though, the first-order MLS method provides direct estimates of the gradients in form of the diffuse derivatives. This allows for the computation and visualization of derived quantities such as vorticity at almost no extra cost.

4.5 Error analysis and optimal shape parameter

The selection of the weight function's shape parameter $h(\boldsymbol{x}_0)$ is a difficult but important task. To highlight the importance, we start this section with a simple example in 2D. Let Franke's test function

$$u(x,y) = \tfrac{1}{2}\mathrm{e}^{-\frac{1}{4}(9x-7)^2-\frac{1}{4}(9y-3)^2} - \tfrac{1}{5}\mathrm{e}^{-(9x-4)^2-(9y-7)^2} + \tfrac{3}{4}\mathrm{e}^{-\frac{1}{4}(9x-2)^2-\frac{1}{4}(9y-2)^2} + \tfrac{3}{4}\mathrm{e}^{-\frac{1}{49}(9x+1)^2+\frac{1}{10}(9y+1)} \tag{4.45}$$

be the function we wish to approximate. As depicted in figure 4.3 its shape is characterized by two peaks, a depression and a flat part. The data sites are given by a set of 1024 Halton points, which are quasi-uniformly distributed within the unit square (cf. fig. 4.3). To keep the problem simple, the sampling values correspond to the exact function values. The left column of figure 4.4 shows the result obtained by the zeroth-order MLS method using a Gaussian kernel as weight function. Three different shape parameters are tested, $h = 0.02,\ 0.05$ and 0.1 (from top to bottom). Clearly, for $h = 0.02$, the zeroth-order MLS fails to return a smooth surface. More precisely, the reconstructed surface is afflicted with severe acne (staircase effect). Widening the shape of the weight function leads to a smoother surface but at the expense of overall accuracy. The inaccuracy is revealed by the deviation of the height of the highest peak and by the deflections near the boundaries with large gradients. As illustrated in the right column of figure 4.4, the first-order MLS methods generates significantly better results. Although the staircase effect is still visible at $h = 0.02$, the irregularities are clearly less pronounced than in the zeroth-order case. However, it is worth mentioning that the least squares problems near the boundaries are close to being ill-posed. This means that in the case of inaccurate input data, a strong amplification of the errors may be observed. Visually, the best result is obtained with a shape parameter of 0.05. The staircase effect is almost not visible and the dominance of the peaks and of the depression seems to be comparable to that of the exact surface. In addition to that the condition of the least squares problems near the boundaries is measurably improved due to the wider shape of the weight function. A further widening, however, leads to an



(a) Input data (1024 Halton points)

(b) Exact solution

Figure 4.3: Numerical example with Franke's test function

over-smoothed solution as shown in the right bottom of figure 4.4. In that case, the scale of the peaks and the depression is evidently smaller than in the original.

Compared to a ProCap measurement the reconstruction scenario in the previous example is much simpler, as the data sites are quasi-uniformly distributed, the input data is exact, and the problem is only two-dimensional. But even in this simple example, the significance of the shape parameter selection becomes clear. In the past, relatively little attention has been paid to that problem. In computer graphics, the shape parameter is usually tuned such that the result meets the visual expectation of the user. To the best of the author's knowledge there exist two more scientific approaches for assessing the mentioned selection problem: In approximation theory, first a tight-as-possible error bound is derived, which, in a second step, is minimized by varying the shape parameter accordingly. In statistics, the optimal shape parameter is obtained by minimizing an estimate of the asymptotic mean squared error or of a related accuracy measure. Below, both techniques are briefly outlined by touching only the relevant aspects.

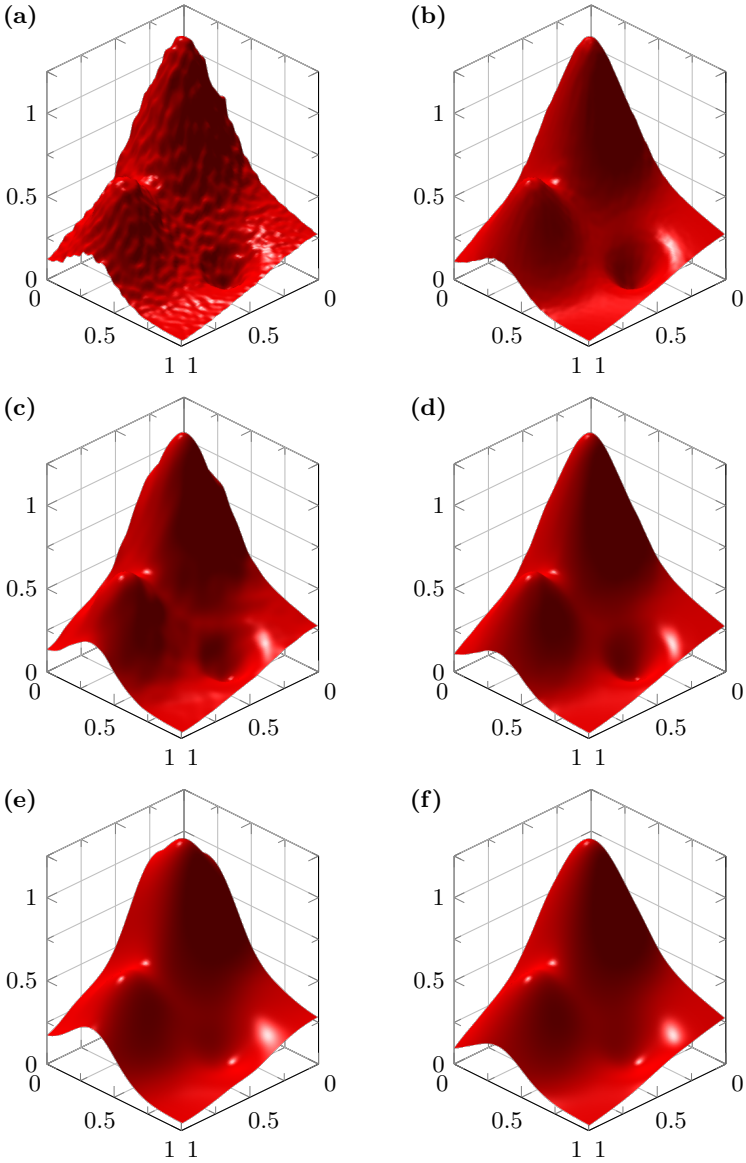


Figure 4.4: Effect of shape parameter h: (a,c,e) 0th order MLS; (b,d,f) 1st order MLS; (a,b) $h = 0.02$; (c,d) $h = 0.05$; (e,f) $h = 0.1$

4.5.1 Error analysis based shape parameter selection

In approximation theory, the accuracy of the MLS approximation is assessed by finding a tight-as-possible bound of the point-wise approximation error, which is defined by the difference between the MLS-approximation $\widehat{u}(\boldsymbol{x})$ and the function to be reconstructed $u(\boldsymbol{x})$:

$$e(\boldsymbol{x}) = \widehat{u}(\boldsymbol{x}) - u(\boldsymbol{x}). \tag{4.46}$$

Let u be $(m+1)$-times continuously differentiable, i.e. $u \in \mathcal{C}^{m+1}(\Omega)$, and let the basis functions in ansatz (4.2) span the polynomial space of degree m, i.e. $\mathcal{B} = \mathcal{P}_m^d$. From the Backus-Gilbert formulation it is known that the generating functions $\psi_i,\ i = 1, \ldots, M$ satisfy

$$\sum_{i=1}^{M} p(\boldsymbol{x}_i)\, \psi_i(\boldsymbol{x}, \boldsymbol{x}) = p(\boldsymbol{x}),\ \forall p \in \mathcal{P}_m^d. \tag{4.47}$$

In section 4.3.1 we have shown that if $|\boldsymbol{\gamma}| < m$ the MLS method provides an elegant and more importantly an efficient way to compute estimates of the derivatives $\partial^{\boldsymbol{\gamma}} u$. Therefore, the point-wise error formula (4.46) can be generalized to

$$e_{\boldsymbol{\gamma}}(\boldsymbol{x}) = \widehat{\partial^{\boldsymbol{\gamma}} u}(\boldsymbol{x}) - \partial^{\boldsymbol{\gamma}} u(\boldsymbol{x}). \tag{4.48}$$

For reasons of compactness the multi-index notation is used. By applying the triangle inequality, the point-wise error can be bounded as follows

$$|e_{\boldsymbol{\gamma}}(\boldsymbol{x})| \leq |\widehat{\partial^{\boldsymbol{\gamma}} u}(\boldsymbol{x}) - \partial^{\boldsymbol{\gamma}} p(\boldsymbol{x})| + |\partial^{\boldsymbol{\gamma}} p(\boldsymbol{x}) - \partial^{\boldsymbol{\gamma}} u(\boldsymbol{x})|, \tag{4.49}$$

where p denotes an arbitrary polynomial in $\mathcal{P}_m^d$. Now, the first term on the right-hand side is reformulated by using the local polynomial reconstruction property (4.47)

$$\begin{aligned} |\widehat{\partial^{\boldsymbol{\gamma}} u}(\boldsymbol{x}) - \partial^{\boldsymbol{\gamma}} p(\boldsymbol{x})| &= |\sum_{i=1}^{M} \partial^{\boldsymbol{\gamma}} \psi_i(\boldsymbol{x}, \boldsymbol{x}) \left[u(\boldsymbol{x}_i) - p(\boldsymbol{x}_{\boldsymbol{i}})\right]| \\ &\leq \|u - p\|_{L_\infty(B_{\delta_{\boldsymbol{x}}})} \sum_{i \in \mathcal{I}_{\boldsymbol{x}}} |\partial^{\boldsymbol{\gamma}} \psi_i(\boldsymbol{x}, \boldsymbol{x})|. \end{aligned} \tag{4.50}$$

Here, $B_{\delta_{\boldsymbol{x}}}$ is the ball centered at $\boldsymbol{x}$ and with radius $\delta_{\boldsymbol{x}}$ and corresponds to the support of the weight function $w(\cdot, \boldsymbol{x})$. $\mathcal{I}_{\boldsymbol{x}}$ is the indices collection of all data sites being contained in $B_{\delta_{\boldsymbol{x}}}$. As $\boldsymbol{x} \in B_{\delta_{\boldsymbol{x}}}$, the second term in (4.49) is bounded by $\|\partial^{\boldsymbol{\gamma}} u - \partial^{\boldsymbol{\gamma}} p\|_{L_\infty(B_{\delta_{\boldsymbol{x}}})}$ and therefore, inequality (4.49) changes to

$$\begin{aligned} |e_{\boldsymbol{\gamma}}(\boldsymbol{x})| \leq \|u-p\|_{L_\infty(B_{\delta_{\boldsymbol{x}}})} \sum_{i \in \mathcal{I}_{\boldsymbol{x}}} |\partial^{\boldsymbol{\gamma}} \psi_i(\boldsymbol{x}, \boldsymbol{x})| \\ + \|\partial^{\boldsymbol{\gamma}} u - \partial^{\boldsymbol{\gamma}} p\|_{L_\infty(B_{\delta_{\boldsymbol{x}}})} \end{aligned} \tag{4.51}$$

Note, the above inequality is true for all $p \in \mathcal{P}_m^d$. In particular, replacing p by the m-th order Taylor polynomial of u allows for further simplification. Taylor's theorem yields

$$\begin{aligned} \|u-p\|_{L_\infty(B_{\delta_{\boldsymbol{x}}})} &\leq C_1 \delta_{\boldsymbol{x}}^{m+1}, \\ \|\partial^{\boldsymbol{\gamma}} u - \partial^{\boldsymbol{\gamma}} p\|_{L_\infty(B_{\delta_{\boldsymbol{x}}})} &\leq C_2 \delta_{\boldsymbol{x}}^{m-|\boldsymbol{\gamma}|+1}, \end{aligned} \tag{4.52}$$

where the constants C_1 and C_2 are given by

$$\begin{aligned} C_1 &= \sum_{|\boldsymbol{\beta}|=m+1} \tfrac{1}{\boldsymbol{\beta}!} \left\|\partial^{\boldsymbol{\beta}} u\right\|_{L_\infty(B_{\delta_{\boldsymbol{x}}})} \\ C_2 &= \sum_{|\boldsymbol{\beta}|=m-|\boldsymbol{\gamma}|+1} \tfrac{1}{\boldsymbol{\beta}!} \left\|\partial^{\boldsymbol{\beta}+\boldsymbol{\gamma}} u\right\|_{L_\infty(B_{\delta_{\boldsymbol{x}}})} \end{aligned} \tag{4.53}$$

Finally, we end up with the following error bound

$$|e_{\boldsymbol{\gamma}}(\boldsymbol{x})| \leq C_1 \Lambda_{m,\boldsymbol{\gamma}}(\boldsymbol{x})\, \delta_{\boldsymbol{x}}^{m+1} + C_2 \delta_{\boldsymbol{x}}^{m-|\boldsymbol{\gamma}|+1} \tag{4.54}$$

Here, $\Lambda_{m,\boldsymbol{\gamma}}(\boldsymbol{x}) := \sum_{i \in \mathcal{I}_{\boldsymbol{x}}} |\partial^{\boldsymbol{\gamma}} \psi_i(\boldsymbol{x}, \boldsymbol{x})|$ is the so called Lebesgue function. In the next paragraph the importance of the Lebesgue function will be discussed.

In general, finding a bound for the Lebesgue function is not trivial except in the zeroth-order case (i.e. $m = 0$ and $\boldsymbol{\gamma} = \mathbf{0}$), where the generating functions are known explicitly. More specifically, $\Lambda_{0,\mathbf{0}} = 1$ unless $\mathcal{I}_{\boldsymbol{x}} = \emptyset$. Furthermore, as $C_1 = C_2$ the error bound of the zeroth-order MLS is simply

$$|e_{\mathbf{0}}(\boldsymbol{x})| \leq 2 C_1 \delta_{\boldsymbol{x}}. \tag{4.55}$$

For $m > 0$, the error analysis becomes rather complicated. If certain conditions on the data sites $\mathcal{X}$ are satisfied and if the support radius $\delta_{\boldsymbol{x}}$ is chosen accordingly, *Levin* (1998) showed that the MLS method provides an approximation order of $\mathcal{O}\left(\delta_{\boldsymbol{x}}^{m+1}\right)$. Based on that, *Wendland* (2001) succeeded to derive an explicit expression for the bound of the Lebesgue function. More recently, *Mirzaei et al.* (2011) extended Wendland's error analysis to obtain explicit error bounds not only for the function estimates but also for the diffuse derivatives. In the following, the results from *Levin* (1998), *Wendland* (2001) and *Mirzaei et al.* (2011) are briefly summarized.

To this end, three mathematical terms need to be introduced, the interior cone condition, the fill distance and the separation distance.

Definition 1 (Interior cone condition)
A set $\Omega \subset \mathbb{R}^d$ satisfies an interior cone condition with respect to a radius $r > 0$ and an angle $\theta \in (0, \pi/2)$ if for all $\boldsymbol{x} \in \Omega$ there exists a vector $\boldsymbol{\xi}(\boldsymbol{x})$ of length 1, such that the cone

$$\begin{aligned} &C(\boldsymbol{x}, \boldsymbol{\xi}, \theta, r) := \\ &\quad \left\{\boldsymbol{x} + \lambda \boldsymbol{y} : \boldsymbol{y} \in \mathbb{R}^d, \|\boldsymbol{y}\|_2 = 1, \lambda \in [0, r], \boldsymbol{y} \cdot \boldsymbol{\xi} \geq cos(\theta)\right\} \end{aligned} \tag{4.56}$$

is a subset of Ω.

Definition 2 (Fill distance)
Consider a domain $\Omega \subset \mathbb{R}^d$ and a finite collection of points $\mathcal{X}$. Then, the fill distance $h_{\mathcal{X},\Omega}$ is defined by the largest ball that has its center in Ω and contains no point of $\mathcal{X}$, i.e.

$$h_{\mathcal{X},\Omega} = \max_{\boldsymbol{x} \in \Omega} \min_{\boldsymbol{x}_i \in \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2 \tag{4.57}$$

Definition 3 (Separation distance)
For a given collection of points $\mathcal{X}$ the separation distance $q_{\mathcal{X}}$ is defined by the smallest distance between any two distinct points of $\mathcal{X}$, i.e.

$$q_{\mathcal{X}} = \min_{\boldsymbol{x}_i \neq \boldsymbol{x}_j} \|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2 \tag{4.58}$$

As before, let $\Omega \subset \mathbb{R}^d$ denote the domain in which the function u is to be approximated and let $\mathcal{X}$ be the set of data sites at which

u is known. The corresponding fill distance is denoted by $h_{\mathcal{X},\Omega}$. Assuming

(A1) that the domain Ω satisfies an interior cone condition with radius r and angle θ,

(A2) that $h_{\mathcal{X},\Omega} \leq \dfrac{3r\sin\theta}{16\left(1+\sin\theta\right)^2 m^2}$

(A3) and that the selected support radius $\delta_{\boldsymbol{x}}$ scales with the fill distance as follows

$$\delta_{\boldsymbol{x}} = \frac{32\left(1+\sin\theta\right)^2 m^2}{3\sin^2\theta} h_{\mathcal{X},\Omega},$$

it can be shown that the Lebesgue function $\Lambda_{m,\boldsymbol{\gamma}}$ is bounded by

$$\Lambda_{m,\boldsymbol{\gamma}} \leq 2\sqrt{\#\mathcal{I}_{\boldsymbol{x}} \frac{\max\limits_{\boldsymbol{r}\in B_{\delta_{\boldsymbol{x}}}} w\left(\boldsymbol{r},\boldsymbol{x}\right)}{\min\limits_{\boldsymbol{r}\in B_{\delta_{\boldsymbol{x}}/2}} w\left(\boldsymbol{r},\boldsymbol{x}\right)}} \left[\frac{3\sin^2\theta}{16m^2\left(1+\sin\theta\right)}\right]^{-|\boldsymbol{\gamma}|} \delta_{\boldsymbol{x}}^{-|\boldsymbol{\gamma}|} \tag{4.59}$$

Here, the expression $\#\mathcal{I}_{\boldsymbol{x}}$ stands for the number of data sites in the support of the weight function centered around the evaluation point $\boldsymbol{x}$. And $B_{\delta_{\boldsymbol{x}}}$ is the ball around $\boldsymbol{x}$ with radius $\delta_{\boldsymbol{x}}/2$. Combining this result with the previously derived formula (4.54) leads to the conclusion that the approximation order of the MLS estimate $\widehat{\partial^{\boldsymbol{\gamma}} u}$ is $\mathcal{O}(\delta_{\boldsymbol{x}}^{m-|\boldsymbol{\gamma}|+1})$ (or in terms of the fill distance $\mathcal{O}(h_{\mathcal{X},\Omega}^{m-|\boldsymbol{\gamma}|+1})$). Experimental verification of this statement can be found in *Wendland* (2001) and *Fasshauer* (2007). But it is important to point out that this is only correct if the assumptions $(A1)-(A3)$ hold.

Before we proceed, there are a few points worth mentioning:

- Due to the complexity of the problem, so far no error bounds have been derived for weight functions with infinite support (e.g. Gaussian kernel) and for weight functions with a locally adapting shape parameter (*Fasshauer*, 2007).

- The above error bound presumes that the support of w is not a function of the evaluation point $\boldsymbol{x}$ but a global constant.

For data sites which are quasi-uniformly distributed[¶] this approach usually leads to satisfactory results. Numerical tests performed by *Wendland* (2001) however suggest that the assumption (A3) is too restrictive and can be relaxed to obtain better results by choosing a much smaller support radius.

- Suppose that the data sites are highly non-uniformly distributed but still fulfill (A2). Then, in regions with a high point density, the bound of the Lebesgue function becomes rather large as it is linked to the number of data sites in the support of the weight function $\#\mathcal{I}_{\boldsymbol{x}}$. To solve this problem, *Lipman* (2009) recently introduced the stable MLS method, a technique that adjusts the weights w such that the Lebesgue function becomes independent of $\#\mathcal{I}_{\boldsymbol{x}}$. Unfortunately, this improvement comes at a cost: the method requires the computation of the Voronoi-tessellation of the data sites. Moreover, the method addresses only one problem caused by the data sites' non-uniformity, namely the influence of the Lebesgue function. The other problem is related to the global criterion of the support radius $\delta_{\boldsymbol{x}}$. Because the non-uniformity of the data sites leads to a large fill distance, it is likely that in regions with a high point density the suggested global support radius exceeds its local optimum.

The derived error bound is useful to select the shape parameter or rather the support radius of the weight function if the data sites are similarly distributed in all regions of the domain of interest Ω, if the fill distance is not too large with respect to Ω[‖] and if the data values are exact and not noisy. In ProCap all three requirements are violated which confirms our initial intuition that a globally constant shape parameter is not adequate. For approximation scenarios similar to the problem encountered in ProCap, the paper of *Lipman et al.* (2006) offers a relatively simple but computationally expensive solution. His idea is to minimize an estimate of the error bound at every evaluation point individually. He uses a slightly different error

[¶] the uniformity can be measured by the ratio between the separation distance and the fill distance

[‖] cf. (A2)

bound than the one derived above:

$$\begin{aligned}\left|e_{\boldsymbol{\gamma}}\left(\boldsymbol{x}\right)\right| \leq \quad & \Lambda_{m,\boldsymbol{\gamma}}(\boldsymbol{x})\varepsilon \\ & + \frac{C\left(\boldsymbol{x}\right)}{(m+1)!} \sum_{i \in \mathcal{I}_{\boldsymbol{x}}} \left|\partial^{\boldsymbol{\gamma}} \psi_i\left(\boldsymbol{x}, \boldsymbol{x}\right)\right| \sum_{\substack{|\boldsymbol{\beta}|= \\ m+1}} \left|\boldsymbol{x}_i - \boldsymbol{x}\right|^{\boldsymbol{\beta}}\end{aligned} \tag{4.60}$$

where ε is the maximum magnitude of the noise and

$$C\left(\boldsymbol{x}\right) = \max_{|\boldsymbol{\alpha}|=m+1} \left\|\partial^{\boldsymbol{\alpha}} u\left(\boldsymbol{x}\right)\right\|_{L_\infty(B_{\delta_{\boldsymbol{x}}})} \tag{4.61}$$

The derivation of this inequality is explained in appendix D. Furthermore, he formulates two approaches: one assumes that the unknown coefficient C is replaced by a global constant while the other uses a local estimate of C. Besides being computationally expensive, the handling of noisy data is questionable. The formula (4.60) suggests that the higher the noise level the more important is the minimization of the Lebesgue function (first term) and the less important is the size of the bandwidth (second term). Although this usually implies a larger bandwidth, it omits the fact that the average of independent random errors converges to zero with the number of samples. In other words, at high noise level the bandwidth obtained by Lipman's approach is likely to be below the optimum. This is best seen by considering the zeroth-order case. There, the Lebesgue function is known to be one and therefore the first term of the inequality (4.60) is a constant no matter how large the bandwidth is. Consequently, the noise level has no effect on the bandwidth computed by the approach above.

4.5.2 Shape parameter selection within the framework of LPR

From a mathematical perspective, the concepts of local polynomial regression (LPR) and MLS are identical. However, the selection of the optimal shape parameter is handled somewhat differently. In LPR, one attempts to minimize the mean squared error (MSE).

This is either done by cross-validation or by so-called plug-in methods (*Wand and Jones*, 1994; *Fan and Gijbels*, 1996). Plug-in methods often take the asymptotic MSE of the estimator as a basis for optimization. Although the approximation scheme in the current ProCap version relies on a more heuristic approach, the formula for the asymptotic MSE may provide useful information for future improvements. The results in this section are simplifications of the formulas presented by *Liu* (2001) and *Ruppert and Wand* (1994), since our attention is only devoted to locally constant (Nadaraya-Watson estimator) and locally linear regressions with *radial* weight functions.

In LPR, the sampling points are modeled as a set of independent and identically distributed $\mathbb{R}^d$-valued random vectors $\{\boldsymbol{X}_i\}_{i=1}^M$ with a probability density function $f_{\boldsymbol{X}} : \mathbb{R}^d \to \mathbb{R}$. For the derivation of the asymptotic MSE, it is assumed that

(a) the support of $f_{\boldsymbol{X}}$ covers the entire domain Ω, and

(b) $f_{\boldsymbol{X}}$ belongs to the continuously differentiable functions.

Within the LPR framework, $\{\boldsymbol{X}_i\}_{i=1}^M$ are generally called design variables. To each of these design variables a scalar response U_i is assigned. In regression analysis, one is interested in estimating the mean function

$$u(\boldsymbol{x}) = \mathrm{E}[U|\boldsymbol{X} = \boldsymbol{x}]. \tag{4.62}$$

For further analysis, a simple model of the scalar response is used

$$U_i = u(\boldsymbol{X}_i) + \sigma\epsilon_i. \tag{4.63}$$

The second term represents the noise, in which $\{\epsilon_i\}_{i=1}^M$ are mutually independent random variables with zero mean and unit variance and σ is the standard deviation. A more complex scenario, in which the noise is spatially correlated is described in detail in *Liu* (2001). As in the classical MLS approach, the locally constant estimator of $u(\boldsymbol{x})$ is obtained by solving the following minimization problem

$$\min_{c\in\mathbb{R}} \left\{ \sum_{i=1}^{M} (U_i - c)^2 w(\boldsymbol{X}_i, \boldsymbol{x}) \right\}. \tag{4.64}$$

For simplicity, the weight function w is assumed to be radial with a locally varying shape characterized by the function h:

$$w\left(\boldsymbol{X}_i, \boldsymbol{x}\right) = \frac{1}{h\left(\boldsymbol{x}\right)} \phi\left(\frac{\left\|\boldsymbol{X}_i - \boldsymbol{x}\right\|_2}{h\left(\boldsymbol{x}\right)}\right) \tag{4.65}$$

ϕ denotes a pre-defined kernel function, i.e. $\int \phi\left(r\right) \mathrm{d}r = 1$ and $\phi\left(-r\right) = \phi\left(r\right),\ \forall r \in \mathbb{R}$. We introduce two quantities describing the shape of the kernel function ϕ:

$$\begin{aligned} \sigma_\phi^2 &:= \int_{-\infty}^{\infty} r^2 \phi\left(r\right)\ \mathrm{d}r \\ p_\phi^2 &:= \int_{-\infty}^{\infty} \phi^2\left(r\right)\ \mathrm{d}r \end{aligned} \tag{4.66}$$

For non-radial weight functions, the interested reader is referred to *Liu* (2001) and *Ruppert and Wand* (1994). Returning to the minimization problem (4.64), the solution, as derived in section 4.3.1, is simply

$$\hat{u}\left(x\right) = c\left(x\right) = \sum_{i=1}^{M} U_i \frac{w\left(\boldsymbol{X}_i, \boldsymbol{x}\right)}{\sum_{j=1}^{M} w\left(\boldsymbol{X}_i, \boldsymbol{x}\right)} \tag{4.67}$$

In order to obtain the asymptotic mean squared error, we assume that as $M \to \infty$, $h \to 0$ with a rate slower than $\mathcal{O}\left(M^{-1/(d+2)}\right)$, i.e. $Mh^{d+2} \to \infty$ as $M \to \infty$. Under these assumptions *Liu* (2001) (Theorem 2.3) showed that the asymptotic MSE for the constant estimator is

$$AMSE_0 = \underbrace{h^4 \sigma_\phi^4 \left[\frac{\Delta u}{2} + \frac{\nabla u \cdot \nabla f_{\boldsymbol{X}}}{f_{\boldsymbol{X}}}\right]^2}_{=\text{Bias}^2} + \underbrace{\frac{\sigma^2 p_\phi^2}{M h^d f_{\boldsymbol{X}}}}_{=\text{Variance}} \tag{4.68}$$

Remarks:

- As intuitively expected, increasing the bandwidth leads to a smaller variance of the estimator but also to a larger bias.

- A larger curvature of the underlying function($\Delta u \uparrow$) leads to

a larger bias of the estimator.

- $\nabla f_{\boldsymbol{X}}/f_{\boldsymbol{X}}$ can be thought of as a directional measure of the local non-uniformity of the sampling points. Thus, a high gradient of u has a negative effect on the $AMSE$ only if the data points are highly non-uniformly distributed. In the previous section we called this observation staircase-effect.

- A larger noise level ($\sigma \uparrow$) leads to a larger variance of the estimator.

- The term $Mh^d f_{\boldsymbol{X}}$ reflects the *effective* number of sampling points in the support of the weight function around $\boldsymbol{x}$. Consequently, a higher effective number of sampling points leads to a smaller variance of the estimator.

The minimization of expression (4.68) returns the optimal bandwidth of the Nadaraya-Watson estimator with respect to the asymptotic MSE:

$$h_{0,\text{opt}} = \left[\frac{d\sigma^2 p_\phi^2}{M f_{\boldsymbol{X}} \sigma_\phi^4 \left(\Delta u + \frac{2\nabla u \cdot \nabla f_{\boldsymbol{X}}}{f_{\boldsymbol{X}}} \right)^2} \right]^{\frac{1}{d+4}} \tag{4.69}$$

This formula is often the starting point of the so-called plug-in methods. The idea of plug-in methods is to replace the unknowns (i.e. $f_{\boldsymbol{X}}, \sigma^2, \Delta u, \nabla u$) by sophisticated estimates. As in practice these calculations are computationally expensive, plug-in methods are currently no option for multi-dimensional real-time applications such as ProCap.

For the linear estimator, one has to solve a slightly different minimization problem:

$$\min_{\substack{c\in\mathbb{R} \\ \boldsymbol{e}\in\mathbb{R}^d}} \left\{ \sum_{i=1}^{M} \left(U_i - c - \boldsymbol{d} \cdot (\boldsymbol{X}_i - \boldsymbol{x}) \right)^2 w\left(\boldsymbol{X}_i, \boldsymbol{x}\right) \right\}. \tag{4.70}$$

The solution to this problem has been derived in section 4.3.1

$$\hat{u}\left(x\right) = c\left(x\right) = \boldsymbol{i}_0^T \boldsymbol{G}^{-1} \boldsymbol{v} \tag{4.71}$$

$\boldsymbol{G}$ and $\boldsymbol{v}$ are defined by (4.12) and (4.13), respectively. The vector $\boldsymbol{i}_0 \in \mathbb{R}^{d+1}$ is simply $[1, 0, \ldots, 0]^T$.

Using the same assumptions as in the constant estimator case, *Liu* (2001) also derived an expression for the asymptotic MSE of the locally linear estimator (Theorem 2.1):

$$AMSE_1 = \underbrace{h^4 \frac{\sigma_\phi^4 \left(\Delta u\right)^2}{4}}_{=\text{Bias}^2} + \underbrace{\frac{\sigma^2 p_\phi^2}{M h^d f_{\boldsymbol{X}}}}_{=\text{Variance}} \tag{4.72}$$

From this expression the optimal bandwidth can be deduced:

$$h_{1,\text{opt}} = \left[\frac{d\sigma^2 p_\phi^2}{M f_{\boldsymbol{X}} \sigma_\phi^4 \left(\Delta u\right)^2} \right]^{\frac{1}{d+4}}. \tag{4.73}$$

The only difference to the locally constant estimator is the fact that the second term in the bias is missing. Consequently, the approximation quality of the locally linear estimator

(i) does not depend on the gradient of the function to be approximated,

(ii) is less affected by the non-uniformity of the sampling points.

However, as mentioned in the previous section the second statement is only valid if the minimization problem (4.70) is well-posed. In other words, the locally linear estimator usually provides better results than the locally constant estimator albeit at the expense of robustness.

4.6 Shape parameter selection in ProCap

In the previous section the importance but also the complexity of the bandwidth selection in MLS has been demonstrated and discussed. Although this problem is well known, so far no consensus on the optimal selection scheme has been reached. It is known that the performance of a selection scheme often depends on the particular approximation problem. Regarding ProCap, the non-uniformity of the data sites and the real-time requirement make the direct application of existing solutions difficult, if not impossible. In the current version, ProCap relies on a rather simple but efficient selection approach. The details of this heuristic scheme will be discussed in the second part of this section. In the first part a local error bound of the MLS approximant is derived which will be used to justify the proposed selection scheme.

4.6.1 Derivation of a local error bound

The aim of this subsection is the derivation of the theorem that forms the basis of the bandwidth selection scheme applied in ProCap. The mathematical building blocks are found in *Wendland* (2001) and *Mirzaei et al.* (2011). However, we pursue a somewhat different goal: Instead of formulating an error bound in terms of the global fill distance an error bound that depends on the local fill distance is sufficient. This simplifies the analysis considerably as the local fill distance has not to be linked with the global fill distance.

First, let us define two balls $B_{\delta_{\boldsymbol{x}}/2}$ and $B_{\delta_{\boldsymbol{x}}}$, both centered at the evaluation point $\boldsymbol{x}$. The radius of the first ball $B_{\delta_{\boldsymbol{x}}/2}$ shall be half the radius of the second ball $B_{\delta_{\boldsymbol{x}}}$, with $\delta_{\boldsymbol{x}}$ denoting the radius of the second ball. As before, let $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{M}$ be a set of arbitrary but mutually distinct data points in $\mathbb{R}^d$. Moreover, the sets $\mathcal{X}_{\delta_{\boldsymbol{x}}/2}$ and $\mathcal{X}_{\delta_{\boldsymbol{x}}}$ are the intersections $\mathcal{X} \cap B_{\delta_{\boldsymbol{x}}/2}$ and $\mathcal{X} \cap B_{\delta_{\boldsymbol{x}}}$, respectively. Note, that $\mathcal{X}_{\delta_{\boldsymbol{x}}/2} \subseteq \mathcal{X}_{\delta_{\boldsymbol{x}}} \subseteq \mathcal{X}$. $\mathcal{I}_{\delta_{\boldsymbol{x}}/2}$ and $\mathcal{I}_{\delta_{\boldsymbol{x}}}$ stand for the indices collection of $\mathcal{X}_{\delta_{\boldsymbol{x}}/2}$ and $\mathcal{X}_{\delta_{\boldsymbol{x}}}$, respectively.

Since a ball satisfies an interior cone condition with the same radius and an angle $\theta = \pi/3$, corollary (4.10) of *Mirzaei et al.* (2011)

can be applied on $B_{\delta_{\boldsymbol{x}}/2}$:

Corollary 1
Let m be a positive integer and let $h_{\mathcal{X}_{\delta_{\boldsymbol{x}}/2},B_{\delta_{\boldsymbol{x}}/2}}$ be the fill distance of $\mathcal{X}_{\delta_{\boldsymbol{x}}/2}$ in $B_{\delta_{\boldsymbol{x}}/2}$. If

$$h_{\mathcal{X}_{\delta_{\boldsymbol{x}}/2},B_{\delta_{\boldsymbol{x}}/2}} \leq \frac{\sqrt{3}}{8m^2\left(2+\sqrt{3}\right)}\delta_{\boldsymbol{x}} \tag{4.74}$$

then there exist for every $\boldsymbol{x}' \in B_{\delta_{\boldsymbol{x}}/2}$ (in particular for $\boldsymbol{x}$) and every multi-index $\boldsymbol{\gamma} \in \mathbb{N}_0^d$ with $|\boldsymbol{\gamma}| \leq m$ real numbers $s_{i,\boldsymbol{\gamma}}(\boldsymbol{x}')$ such that

$$\sum_{i\in\mathcal{I}_{\delta_{\boldsymbol{x}}/2}} |s_{i,\boldsymbol{\gamma}}(\boldsymbol{x}')| \leq 2\left(2+\sqrt{3}\right)^{-|\boldsymbol{\gamma}|} h^{-|\boldsymbol{\gamma}|}_{\mathcal{X}_{\delta_{\boldsymbol{x}}/2},B_{\delta_{\boldsymbol{x}}/2}} \tag{4.75}$$

$$\leq 2\left(\frac{8m^2}{\sqrt{3}}\right)^{|\boldsymbol{\gamma}|} \delta_{\boldsymbol{x}}^{-|\boldsymbol{\gamma}|} \tag{4.76}$$

and

$$\sum_{i\in\mathcal{I}_{\delta_{\boldsymbol{x}}/2}} s_{i,\boldsymbol{\gamma}}(\boldsymbol{x}')\, p(\boldsymbol{x}_i) = \partial^{\boldsymbol{\gamma}} p(\boldsymbol{x}') \tag{4.77}$$

for all $p \in \mathcal{P}_m^d$.

Suppose $\tilde{s}_{i,\boldsymbol{\gamma}}$ is the trivial extension of the numbers $s_{i,\boldsymbol{\gamma}}$ onto $\mathcal{I}_{\delta_{\boldsymbol{x}}}$:

$$\tilde{s}_{i,\boldsymbol{\gamma}}(\boldsymbol{x}') = \begin{cases} s_{i,\boldsymbol{\gamma}}(\boldsymbol{x}'), & i \in \mathcal{I}_{\delta_{\boldsymbol{x}}/2} \\ 0, & \text{otherwise} \end{cases} \tag{4.78}$$

As in chapter 4, the generating functions of the mth-order MLS approximation at $\boldsymbol{x}$ are denoted by $\psi_i(\boldsymbol{x}', \boldsymbol{x})$. If we assume that the support of the weight function w is given by the ball $B_{\delta_{\boldsymbol{x}}}$, the Lebesgue function at $\boldsymbol{x}$ can be written as follows

$$\Lambda_{m,\gamma}(\boldsymbol{x}) = \sum_{i\in\mathcal{I}_{\delta_{\boldsymbol{x}}}} \left|\partial^{\boldsymbol{\gamma}}\psi_i(\boldsymbol{x},\boldsymbol{x})\right| \tag{4.79}$$

To obtain an explicit expression for the bound of the Lebesgue

function, we first apply the Cauchy-Schwarz inequality to this equation yielding

$$\begin{aligned}\Lambda_{m,\gamma}(\boldsymbol{x}) &= \sum_{i\in\mathcal{I}_{\delta_{\boldsymbol{x}}}} \left|\partial^{\gamma}\psi_i(\boldsymbol{x},\boldsymbol{x})\right| \sqrt{\frac{w(\boldsymbol{x}_i,\boldsymbol{x})}{w(\boldsymbol{x}_i,\boldsymbol{x})}} \\ &\leq \underbrace{\sqrt{\sum_{i\in\mathcal{I}_{\delta_{\boldsymbol{x}}}} \left|\partial^{\gamma}\psi_i(\boldsymbol{x},\boldsymbol{x})\right|^2 \frac{1}{w(\boldsymbol{x}_i,\boldsymbol{x})}}}_{=:A} \underbrace{\sqrt{\sum_{i\in\mathcal{I}_{\delta_{\boldsymbol{x}}}} w(\boldsymbol{x}_i,\boldsymbol{x})}}_{=:B} \end{aligned} \tag{4.80}$$

While finding a bound for the second term (B) is trivial

$$B \leq \sqrt{\#\mathcal{I}_{\delta_{\boldsymbol{x}}} \max_{\boldsymbol{x}'\in B_{\delta_{\boldsymbol{x}}}} w(\boldsymbol{x}',\boldsymbol{x})}, \tag{4.81}$$

the steps used for bounding the first term (A) are a bit more involved. Note that by definition of the MLS approach (Backus-Gilbert), the generating functions ψ_i minimize term A under the polynomial reproduction constraint (cf. eq. (4.77)). Thus, if the local fill distance $h_{\mathcal{X}_{\delta_{\boldsymbol{x}}/2},B_{\delta_{\boldsymbol{x}}/2}}$ satisfies condition (4.74), it is valid to write

$$\begin{aligned} A &\leq \sqrt{\sum_{i\in\mathcal{I}_{\delta_{\boldsymbol{x}}}} \left|\tilde{s}_{i,\boldsymbol{\gamma}}(\boldsymbol{x})\right|^2 \frac{1}{w(\boldsymbol{x}_i,\boldsymbol{x})}} = \sqrt{\sum_{i\in\mathcal{I}_{\delta_{\boldsymbol{x}}/2}} \left|s_{i,\boldsymbol{\gamma}}(\boldsymbol{x})\right|^2 \frac{1}{w(\boldsymbol{x}_i,\boldsymbol{x})}} \\ &\leq \sqrt{\frac{\sum\limits_{i\in\mathcal{I}_{\delta_{\boldsymbol{x}}/2}} \left|s_{i,\boldsymbol{\gamma}}(\boldsymbol{x})\right|^2}{\min\limits_{\boldsymbol{x}'\in B_{\delta_{\boldsymbol{x}}/2}} w(\boldsymbol{x}',\boldsymbol{x})}} \leq \frac{\sum\limits_{i\in\mathcal{I}_{\delta_{\boldsymbol{x}}/2}} \left|s_{i,\boldsymbol{\gamma}}(\boldsymbol{x})\right|}{\sqrt{\min\limits_{\boldsymbol{x}'\in B_{\delta_{\boldsymbol{x}}/2}} w(\boldsymbol{x}',\boldsymbol{x})}} \\ &\leq \frac{2}{\sqrt{\min\limits_{\boldsymbol{x}'\in B_{\delta_{\boldsymbol{x}}/2}} w(\boldsymbol{x}',\boldsymbol{x})}} \left(\frac{8m^2}{\sqrt{3}}\right)^{|\boldsymbol{\gamma}|} \delta_{\boldsymbol{x}}^{-|\boldsymbol{\gamma}|} \end{aligned} \tag{4.82}$$

Finally, by means of equation (4.54) we can now formulate our main result:

Theorem 1
Given a set of points $\mathcal{X}$, suppose that the fill distance of the subset $\mathcal{X}_{\delta_{\boldsymbol{x}}/2} = \mathcal{X} \cap B_{\delta_{\boldsymbol{x}}/2}$ in $B_{\delta_{\boldsymbol{x}}/2}$ satisfies

$$h_{\mathcal{X}_{\delta_{\boldsymbol{x}}/2}, B_{\delta_{\boldsymbol{x}}/2}} \leq \frac{\sqrt{3}}{8m^2\left(2+\sqrt{3}\right)}\delta_{\boldsymbol{x}} \tag{4.83}$$

Then the approximation error at $\boldsymbol{x}$ is bounded as follows

$$\begin{aligned} &|e_{\boldsymbol{\gamma}}(\boldsymbol{x})| \leq \\ &\left[2C_1\sqrt{\#\mathcal{I}_{\delta_{\boldsymbol{x}}}\frac{\max\limits_{\boldsymbol{x}'\in B_{\delta_{\boldsymbol{x}}}} w(\boldsymbol{x}',\boldsymbol{x})}{\min\limits_{\boldsymbol{x}'\in B_{\delta_{\boldsymbol{x}}/2}} w(\boldsymbol{x}',\boldsymbol{x})}}\left(\frac{8m^2}{\sqrt{3}}\right)^{|\boldsymbol{\gamma}|} + C_2\right]\delta_{\boldsymbol{x}}^{m-|\boldsymbol{\gamma}|+1} \end{aligned} \tag{4.84}$$

with

$$\begin{aligned} C_1 &= \sum_{|\boldsymbol{\beta}|=m+1} \tfrac{1}{\boldsymbol{\beta}!}\left\|\partial^{\boldsymbol{\beta}}u\right\|_{L_\infty(B_{\delta_{\boldsymbol{x}}})} \\ C_2 &= \sum_{|\boldsymbol{\beta}|=m-|\boldsymbol{\gamma}|+1} \tfrac{1}{\boldsymbol{\beta}!}\left\|\partial^{\boldsymbol{\beta}+\boldsymbol{\gamma}}u\right\|_{L_\infty(B_{\delta_{\boldsymbol{x}}})} \end{aligned} \tag{4.85}$$

Considering the most relevant case $\gamma = 0$, the above error formula reduces to

$$|e_{\mathbf{0}}(\boldsymbol{x})| \leq C_1\left[1+2\sqrt{\#\mathcal{I}_{\delta_{\boldsymbol{x}}}\frac{\max\limits_{\boldsymbol{x}'\in B_{\delta_{\boldsymbol{x}}}} w(\boldsymbol{x}',\boldsymbol{x})}{\min\limits_{\boldsymbol{x}'\in B_{\delta_{\boldsymbol{x}}/2}} w(\boldsymbol{x}',\boldsymbol{x})}}\right]\delta_{\boldsymbol{x}}^{m+1} \tag{4.86}$$

4.6.2 A heuristic approach

Challenges

Figure 4.5 gives an example of how the data sites are typically distributed during the course of a measurement. The non-uniformity is usually defined by the ratio between the fill distance and the separation distance. Although the non-uniformity decreases as the measurement progresses, the local fill distance $h_{\mathcal{X}_{\delta_{\boldsymbol{x}}/2}, B_{\delta_{\boldsymbol{x}}/2}}$ remains large. Hence, if the shape parameter is chosen such that inequa-

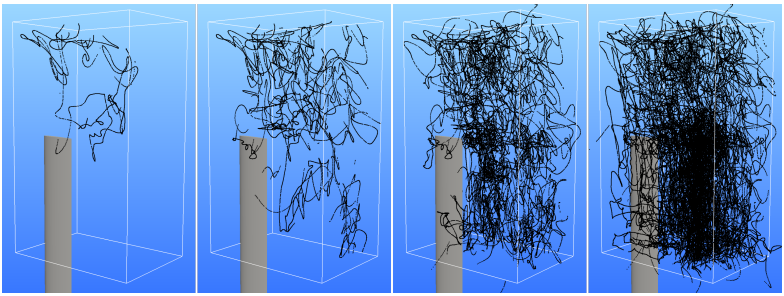


Figure 4.5: Illustration of the data sites distribution during a ProCap measurement. Time increases from left to right.

lity (4.83) holds it is quite possible that the support of the weight function is large. Under the assumption that the previously derived error bound is tight this also implies a large approximation error. This follows not only because the approximation error is proportional to $\delta_{\boldsymbol{x}}^{m+1}$ but also because the input point set is characterized by a small separation distance. The small separation distance arises from the fact that ProCap acquires data along a single path at a relatively high frequency. It follows that the number of data sites in the support of the weight function and therefore the value of the Lebesgue function are large too.

Although the derived error bound (4.86) does not account for measurement noise, one can identify two competing mechanisms related to noise:

- If the noise is random and mutually independent logic would suggest that the variance of the MLS estimate decreases with the number of data sites taken into account (cf. section 4.5.2).

- On the other hand, a widening of the support radius may not only increase the number of included data sites but also the non-uniformity. In this case the Lebesgue function becomes large and amplification of the measurement noise by the approximation process is the logical consequence.

Design decisions and simplifications

In view of this, we are now able to define the guiding principles of our selection scheme:

- Reduce the influence of the number of measurement points on the Lebesgue function.
- Decrease the data sites' non-uniformity.
- Select the bandwidth such that inequality (4.83) is approximately satisfied.
- Keep the computational costs to a minimum.

These goals may be achieved by taking the following measures:

1. **Spatial binning:** The data sites' non-uniformity can be reduced by either decreasing the fill distance or by increasing the separation distance. While the fill distance is automatically decreased in the course of the experiment, the separation distance remains the same. This lends weight to the argument that the approximation quality can be improved by imposing a lower bound on the separation distance. Simply discarding of samples, however, is not advisable as useful information is lost that may be used to reduce the noise level of the input data. In ProCap we pursue a different approach: The domain is divided into cubic subdomains (voxels). We then reduce the data in each voxel by taking the average over the measurement points contained in this voxel. To put it another way, a voxel represents all measurement points within this sampling volume by a single, less noisy data point. As the number of data sites per voxel is not a constant the noise levels of the voxel averages are likely to be different. This issue can be resolved by multiplying the weight of each voxel by an individual certainty value. It seems clear that the certainty value should reflect the number of effective measurement points within a voxel. Details of how to model this certainty function are discussed below.
2. **Reduction of the domain of interest:** A maximum ball radius δ_{max} is defined. If all voxels in a ball with this radius

are empty no approximation at the ball's center is computed. By doing that, we not only restrict the domain of interest, we also lower the fill distance of the data sites which in turn leads to a better approximation.

3. **Limiting the number of considered data sites:** The selection of the optimal shape parameter is always a trade-off between keeping the support size small and bounding the Lebesgue function. As the number of data sites is monotonically increasing with the support radius, logic suggests that the minimum of the Lebesgue function is given by the smallest bandwidth that still satisfies inequality (4.83)**. Since the spatial binning ensures that the effective separation distance is larger or equal than the voxel size it seems reasonable to replace inequality (4.83) by a slightly different criterion: If the ball $B_{\delta_{\boldsymbol{x}}}$ around the evaluation point $\boldsymbol{x}$ contains k or more measurement points (voxel averages) the error bound (4.86) may still hold. k is an integer that has to be specified by the user. From a mathematical standpoint, we cannot be certain that the two criteria are equivalent, but provided that k is selected appropriately the k-nearest neighbor criterion should be sufficient for the majority of evaluation points. Since the certainty of the voxel averages is not a constant, it is better to replace the number of measurement points by the sum of all certainty values within the support of the weight function. In short, instead of an integer k a threshold value s needs to be defined. The bandwidth is then set as small as possible but such that the sum of certainty values does not fall below s. To avoid unsatisfactory results ProCap offers the possibility to visualize the distribution of the Lebesgue function. This allows the operator to rescan regions where the point distribution is not favorable for the approximation scheme.

We make no claims here that the spatial binning as proposed above is the best solution in terms of approximation accuracy but it offers a number of advantages: The separation distance is increased, the noise of the input data is reduced, the amount of data to be

**assuming that the error bound derived in the previous section is tight

transferred to the GPU is significantly smaller than before and as a result of the data sites' regularity the k-nearest-neighbor search becomes simple. For the sake of completeness it is worth noting that the subdivision of the domain into voxels is not only used to reduce the measurement points, it also defines the points at which the MLS approximation is computed. Consequently, the voxel size should be set in accordance with the selected spatial resolution and/or the resolution capability of the probe at hand.

Certainty function

The concept of adding a certainty function c to the input data was introduced by *Knutsson and Westin* (1993) as integral part of the so-called normalized convolution method (NC). NC is a method which, in its simplest form, is identical to the previously introduced MLS method. In most cases, *Knutsson and Westin* (1993) suggested c to be modeled as an indicator function, i.e. $c = 1$ if data at the corresponding location is available and 0 otherwise. In ProCap this binary ansatz seems to be inappropriate as the number of measurement values per voxel and therefore the voxels' certainty levels do vary in space.

In what follows, we assume that the measurements in a voxel $\{\widetilde{u}_i\}_{i=1}^{M_v}$ are discrete values of one observation $\widetilde{u}$ of a continuous time random process $\widetilde{U}(t)$. Naturally, this assumption is only justified if the size of the voxels is sufficiently small, otherwise the spatial dependence cannot be ignored. The random process $\widetilde{U}(t)$ shall capture both: measurement errors and fluctuations due to turbulent motions in the flow. For simplicity, however, we assume that $\widetilde{U}(t)$ is a wide-sense stationary process, i.e.

$$\text{(C1)} \qquad \mathrm{E}\left[\widetilde{U}(t)\right] = u = const. \tag{4.87}$$

$$\text{(C2)} \qquad \mathrm{E}\left[\left(\widetilde{U}(t) - u\right)\left(\widetilde{U}(t+\tau) - u\right)\right] = C(\tau) \tag{4.88}$$

$C(\tau)$ denotes the autocovariance function. The integral time scale

of this process is defined as

$$\mathcal{T} = \frac{1}{C(0)} \int_0^\infty C(\tau)\ \mathrm{d}\tau \tag{4.89}$$

and is assumed to be finite. Furthermore, let the random process be ergodic in the wide sense, i.e.

$$\text{(C3)} \quad u = \lim_{T\to\infty} \frac{1}{T} \int_0^T \widetilde{U}(t)\ \mathrm{d}t \tag{4.90}$$

$$\text{(C4)} \quad C(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_0^T \left(\widetilde{U}(t) - u\right)\left(\widetilde{U}(t+\tau) - u\right)\ \mathrm{d}t \tag{4.91}$$

The question we are interested in is how well the voxel average $1/M_v \sum_{i=1}^{M_v} \widetilde{u}_i$ estimates $\mathrm{E}[\widetilde{U}(t)]$. But before we turn to the discrete case it is helpful to study the properties of the continuous time average estimator:

$$\widehat{u} = \frac{1}{T} \int_0^T \widetilde{u}\ \mathrm{d}t \tag{4.92}$$

Here, T defines the period over which $\widetilde{u}$ is averaged. Clearly, if the conditions (C1)-(C4) hold the estimator is unbiased:

$$\mathrm{E}[\widehat{u}] = \mathrm{E}\left[\frac{1}{T}\int_0^T \widetilde{u}\ \mathrm{d}t\right] = \frac{1}{T}\int_0^T \mathrm{E}\left[\widetilde{U}\right]\ \mathrm{d}t = \frac{T}{T}\mathrm{E}\left[\widetilde{U}\right] = \mathrm{E}\left[\widetilde{U}\right] \ \blacksquare \tag{4.93}$$

The determination of the estimators variance is a bit more involved:

$$\begin{aligned}
\mathrm{Var}[\widehat{u}] &= \mathrm{E}\left[\left(\frac{1}{T}\int_0^T \widetilde{u}\ \mathrm{d}t - u\right)^2\right] = \mathrm{E}\left[\left(\frac{1}{T}\int_0^T \widetilde{u} - u\ \mathrm{d}t\right)^2\right] \\
&= \frac{1}{T^2}\int_0^T\int_0^T \mathrm{E}\left[\left(\widetilde{u}(t) - u\right)\left(\widetilde{u}(t') - u\right)\right]\ \mathrm{d}t\ \mathrm{d}t' \\
&= \frac{1}{T^2}\int_0^T\int_0^T C(t-t')\ \mathrm{d}t\ \mathrm{d}t' = \frac{2}{T}\int_0^T C(\tau)\ \mathrm{d}\tau \\
&\approx \frac{2\mathcal{T}}{T}\mathrm{Var}\left[\widetilde{U}\right]
\end{aligned} \tag{4.94}$$

The last step is only justified if C is rapidly decaying and if T is larger than $\mathcal{T}$. We now turn to the discrete case where the integral can be replaced by a sum

$$\widehat{u} = \frac{1}{M_v \Delta t} \sum_{i=1}^{M_v} \widetilde{u}_i \Delta t = \frac{1}{M_v} \sum_{i=1}^{M_v} \widetilde{u}_i \tag{4.95}$$

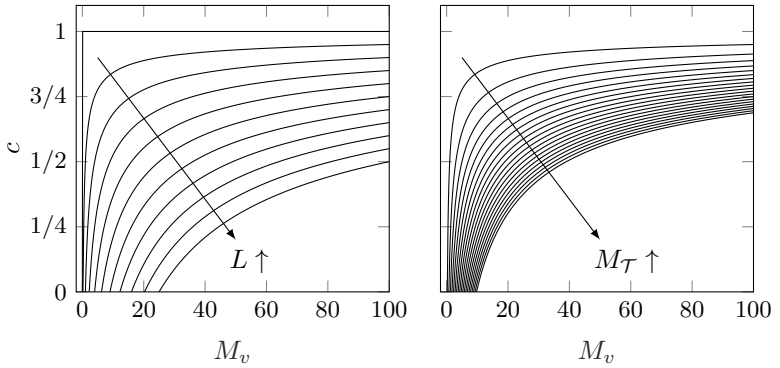
Here, the time between two samples Δt is assumed to be constant. This is reasonable since in most cases the samples contained in one voxel are from the same path segment. Now, with the same reasoning as above, the discrete time average is unbiased and its variance can be approximated by

$$\mathrm{Var}\left[\widehat{u}\right] \approx \frac{2M_{\mathcal{T}}}{M_v} \mathrm{Var}\left[\widetilde{U}\right] \tag{4.96}$$

where $M_{\mathcal{T}}$ is number of samples that fit into the integral time scale $\mathcal{T}$ (i.e. $\mathcal{T}/\Delta t$). If the measurements are uncorrelated the variance of the average is known to be

$$\mathrm{Var}\left[\widehat{u}_{uncor.}\right] = \frac{1}{M_v} \mathrm{Var}\left[\widetilde{U}\right] \tag{4.97}$$

Consequently, $M_v/(2M_{\mathcal{T}})$ can be considered as the effective number of samples in a voxel. In other words, if one is only interested in the mean values sampling with a frequency larger than $1/(2\mathcal{T})$ does not improve the accuracy of the estimated average. For Gaussian noise we know that the width of a confidence interval is proportional to the standard deviation. This strongly suggests that the certainty function should be tied to the standard deviation of the discrete time average estimator. To this end, we define a value σ_0 that acts as a cut-off standard deviation. More specifically, if the standard deviation of the estimator is larger than this value the certainty value shall be zero. On the other hand, if the standard deviation is below σ_0 the certainty function shall take a value between 0 and 1. More precisely, the certainty shall go to 1 as the number of samples M_v goes to infinity. In mathematical terms, we propose the following



(a) Effect of the noise level L **(b)** Effect of the integral time $M_\mathcal{T}$

Figure 4.6: Left: Influence of the noise level L on the certainty function c. $L = 0 \ldots 0.5 \ldots 5$, $M_\mathcal{T} = 0.5$. Right: Influence of the correlation length $M_\mathcal{T}$ on the certainty function c. $M_\mathcal{T} = 0.5 \ldots 1 \ldots 20$, $L = 0.5$

model

$$c\left(M_v\right) = \max\left(0, 1 - \sqrt{\frac{2M_\mathcal{T}}{M_v}\underbrace{\frac{\sigma_u}{\sigma_0}}_{=:L}}\right) \tag{4.98}$$

Here, σ_u is the standard deviation of the measurement signal, while the ratio L describes a measure of the noise level. L is a positive number and goes to zero as the input data become less noisy[††]. The second quantity that has to be known is the correlation length $M_\mathcal{T}$. In the simplest case where one assumes that the measurement points are uncorrelated $M_\mathcal{T}$ reaches its minimum, which is 0.5. In the current version L and $M_\mathcal{T}$ are pre-defined, global values. In the future, they might be deduced from the measurement itself. The effect of the noise level L and the integral time scale $M_\mathcal{T}$ on the certainty function c is illustrated in figure 4.6.

[††]turbulence & measurement noise

4.7 An overview of the implementation of the reconstruction algorithm

This section gives a short overview of the interpolation algorithm and its implementation into the ProCap visualization software. As mentioned before it is essential for the method to generate an up-to-date visualization of the flow. This clearly requires that the interpolation has to be recomputed in real-time. The basic steps to be performed in one update cycle are displayed in figure 4.7 and described below. Steps 1 to 4 are computed on the CPU, while steps 5 and 6 are executed on the GPU by means of a compute shader.

❶ The measurement points collected in the period between two interpolation updates form the input data for the next interpolation step. The exact number depends on the capture rate of the data acquisition (MoCap and probe) and on the frame rate of the ProCap visualization software. Normally, the two rates are of the same order, i.e. there are only a few measurement points to process. If the distance between two consecutive points exceeds a certain threshold the group of points is split into two or even more subgroups. To guarantee a high update rate only one subgroup is processed per frame. Then for every measurement point of the subgroup one has to perform two tasks: First one has to identify the voxel to which the point belongs and second one has to recompute the corresponding voxel averages and the certainty value. Hereafter, we denote the set of voxels being updated as raw-data-region (RDR).

❷ Based on the raw-data region the voxels in which the interpolated values might change have to be identified. More precisely, one has to find all voxels that are within a radius of δ_{max} to a voxel of the raw-data region. This set of voxels is called region-to-interpolate (RTI)

❸ The interpolation is solved by means of a compute shader – a program that runs on the GPU but is not part of the rendering pipeline. As the invocation of a compute shader is decomposed into groups of threads the region-to-interpolate has to be

extended in such a way that the number of voxels in each dimension is a multiple of the 1D size of a thread group.

❹ In the next stage, one has to select the voxels whose data is required to compute the approximation in the extended RTI (eRTI). As depicted in figure 4.7 this set of voxels, named as region of influence (ROI), is determined by adding a layer of thickness δ_{max} to the eRTI. Subsequently, the voxel values of the ROI are sent to the global memory of the GPU.

❺ After the completion of the data transfer, the interpolation compute shader is invoked. A compute shader divides its work into a 3D grid of thread groups. A thread group is simply a set of threads. Ideally, the number of threads per group is a multiple of the warp size[‡‡]. Here, every voxel of the RTI corresponds to a single thread. Threads within one group are not independent but execute the same instruction on different data. In exchange, they can be synchronized and have access to so-called group-shared memory, a fast, user-controlled cache. As group-shared memory can be accessed approximately 100 times faster than global memory[§§], it is crucial to transfer the data used by several threads to the group-shared memory. Consequently, the compute shader's kernel function comprises two stages: First the relevant data of the ROI is transferred to the group-shared memory. Second all threads within a group are synchronized which is then followed by the actual interpolation.

❻ For each voxel in the RTI a moving least squares problem is solved. First of all, the bandwidth/shape parameter of the weight function is determined using the method described in section 4.6. Next, the first-order MLS problem is solved by directly inverting the 4×4 Gram matrix. If at some point the user-defined certainty level is not reached although the maximum bandwidth is applied then the first-order MLS approximation is replaced by the more robust zeroth-order MLS

[‡‡] the warp size for most nVidia GPUs is 32
[§§] assuming that no bank conflicts occur

estimate. The zeroth-order MLS approximation can easily be deduced from the terms used to compute the first-order estimate. Thus, code branching can be kept to a minimum.

The described program was tested on two laptops:

1. CPU: Intel Core i7-4810MQ, $2.8GHz$, 16GB RAM, GPU: nVidia Quadro K2100M, OS: Windows 10 Pro, 64 bit
2. CPU: Intel Core i7-4700MG, $2.4GHz$, 32GB RAM, GPU: nVidia GeForce GTX 770M, OS: Windows 10 Home, 64 bit

In both cases the maximum frame rate of $60Hz$ could be achieved without changing the compute shader parameters. Last but not least, it is worth stressing that the progress of the scanning and the quality of the interpolation can be monitored in real-time by visualizing the distribution of the bandwidth, the distribution of the Lebesgue function and/or the distribution of a flow field quantity.

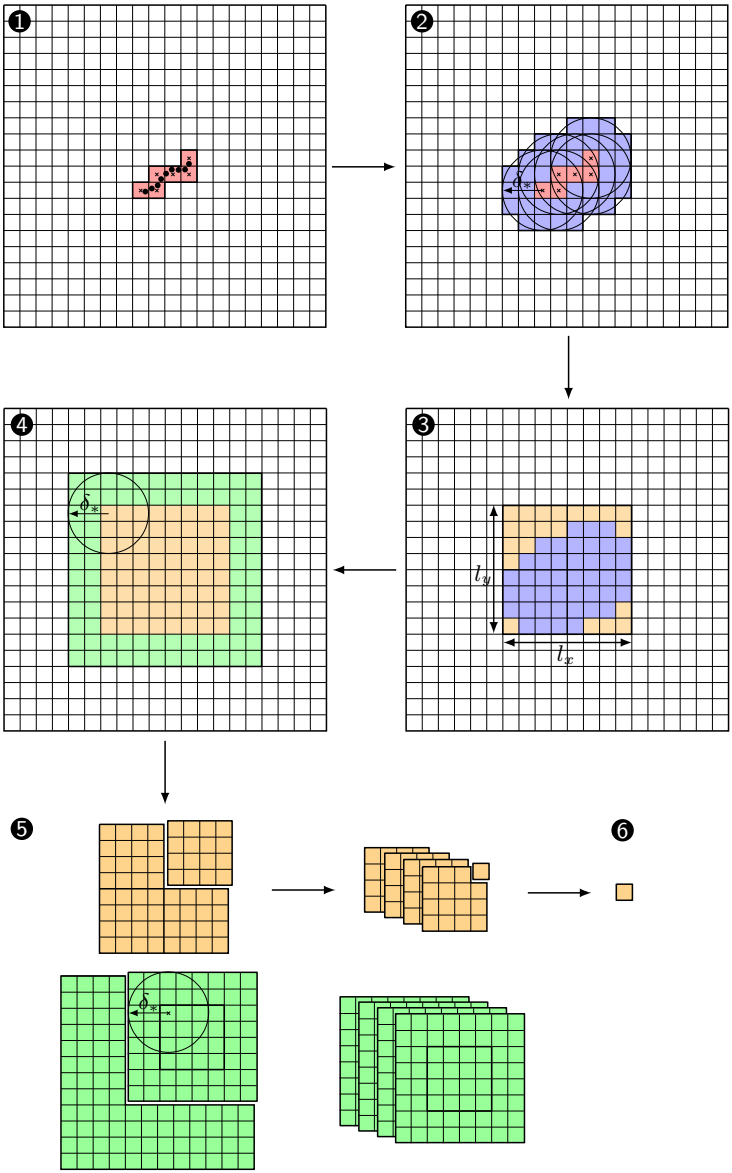


Figure 4.7: Interpolation algorithm in ProCap. $\delta_* = \delta_{max}$
RDR, RTI, eRTI, ROI

4.8 Pseudo-divergence-free moving least squares

In low-speed aerodynamics, the flow under investigation is known to be incompressible. This implies that the divergence of the velocity field has to vanish at all points to satisfy mass conservation. Due to various errors, velocity measurements that resolve all flow structures usually violate this condition. Denoising schemes that enforce mass conservation have proven to perform significantly better than those ignoring the solenoidal nature of the velocity field (*Song et al.*, 1993; *Azijli and Dwight*, 2015). For measurement techniques that do not resolve all flow structures, the task is not only to reduce the measurement noise, but also to fill in the data gaps in a physically consistent way. In general, the incorporation of mass conservation into the flow field reconstruction process can be done in two ways:

(a) *Two-step approach*: 1.) Reconstruct velocity field ignoring mass conservation. 2.) Solenoidal denoising.

(b) *One-step approach*: Direct use of a divergence-free interpolation/approximation scheme to reconstruct the velocity field.

The latter is arguably the more elegant approach. Within the framework of RBF, *Narcowich and Ward* (1994) introduced matrix-valued basis functions that lead automatically to a solenoidal vector field. *Lowitzsch* (2002) extended this concept to locally supported basis functions. In a similar context, *Azijli and Dwight* (2015) recently demonstrated the effectiveness of solenoidal filtering within the framework of Kriging. In MLS, one can think of three direct ways to enforce mass conservation:

(a) *Penalty method:* An additional term is added to the functional to be minimized. This term consists of a measure of the local divergence times a so-called penalty parameter. This parameter steers the priority of getting a divergence-free approximation.

(b) *Exact method:* The exact divergence of the MLS approximation is forced to be zero. This can be done by replacing the penalty parameter by a Lagrangian multiplier.

(c) *Pseudo-divergence-free method:* This approach is identical to the previous one except that instead of the exact, the diffuse derivatives are used to build the divergence.

A disadvantage of the first approach is that a new parameter is introduced. Similarly to the bandwidth, this parameter has to be preselected. As this adds to the complexity of the optimal bandwidth problem, this approach was not further pursued. The second approach leads to a second order differential equation for the Lagrangian multiplier which has to be solved in the entire domain Ω. This is not only computationally expensive, it also ruins the local character of the MLS method. Thus, with regard to the real-time capability of the ProCap system, the third approach is the most promising and has been successfully integrated into the ProCap reconstruction procedure. As the name may suggest, the method does not lead to a proper divergence-free vector field, since the diffuse derivatives and the exact derivatives of the approximation are not equal. However, it is worth to point out that the approximation orders of both estimates, the diffusive and the exact derivatives, are identical and therefore, similar results are expected. The idea to apply the divergence constraint on the diffuse derivatives is not new. In fact, it has been the subject of *Huerta et al.* (2004) and *Hong et al.* (2008). However, in the present work the divergence constraint is treated slightly differently allowing decoupling of the resulting linear system such that the pseudo-divergence-free MLS approximation can be computed in a very efficient way.

4.8.1 Mathematical derivation

Consider a vector-valued function $\boldsymbol{u} : \mathbb{R}^d \to \mathbb{R}^d$, which is known to be divergence-free, i.e.

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0. \tag{4.99}$$

Given a set of samples $\boldsymbol{u}_i = \boldsymbol{u}(\boldsymbol{x}_i)$ at M distinct locations $\boldsymbol{x}_i$, $i = 1, \ldots, M$, the task is to find an approximation $\widehat{\boldsymbol{u}}$ at an arbitrary point $\boldsymbol{x}_0$. For this the MLS approximation ansatz (4.2) is extended

to multi-valued functions:

$$\widehat{\boldsymbol{u}}\left(\boldsymbol{x},\boldsymbol{x}_0\right)=\sum_{j=0}^{N}\boldsymbol{c}_j\left(\boldsymbol{x}_0\right)b_j\left(\boldsymbol{x}-\boldsymbol{x}_0\right), \tag{4.100}$$

with $\boldsymbol{c}_j\left(\boldsymbol{x}_0\right)\in\mathbb{R}^d$. As demonstrated in section 4.3.1, the standard MLS approximation is obtained by solving the following unconstrained minimization problem

$$\sum_{i=1}^{M}w\left(\boldsymbol{x}_i,\boldsymbol{x}_0\right)\left|\boldsymbol{u}\left(\boldsymbol{x}_i\right)-\widehat{\boldsymbol{u}}\left(\boldsymbol{x}_i,\boldsymbol{x}_0\right)\right|^2\rightarrow\min_{\boldsymbol{c}_j}. \tag{4.101}$$

Without adding further constraints this problem can be split into d decoupled minimization problems, each associated with a single component of $\boldsymbol{u}$. Here though, the solution space is limited such that the local approximation is divergence-free with respect to the diffuse derivatives, i.e.

$$\boldsymbol{\nabla}_{\boldsymbol{x}}\cdot\widehat{\boldsymbol{u}}\left(\boldsymbol{x},\boldsymbol{x}_0\right)=0,\ \forall\boldsymbol{x}\in\mathbb{R}^d \tag{4.102}$$

Restricting our analysis on the first-order case with monomial basis functions[¶¶], this constraint simply translates to

$$\sum_{k=1}^{d}c_{k,k}=0. \tag{4.103}$$

Here, the first subscript describes the index of the basis function, while the second subscript labels the component of the vector $\boldsymbol{c}_k$. By introducing a Lagrangian multiplier $\lambda\left(\boldsymbol{x}_0\right)$, the constraint minimization problem can be formulated as

$$\sum_{k=1}^{d}\left\{\sum_{i=1}^{M}w\left(\boldsymbol{x}_i,\boldsymbol{x}_0\right)\left(u_{i,k}-\left[c_{0,k},\ldots,c_{d,k}\right]\widehat{\boldsymbol{x}}_i\right)^2-\lambda c_{k,k}\right\}\rightarrow\min_{\boldsymbol{c},\lambda}, \tag{4.104}$$

where $\widehat{\boldsymbol{x}}_i=\left[1,\boldsymbol{x}_i^T-\boldsymbol{x}_0^T\right]^T$. The Euler-Lagrange equations of this constrained minimization problem lead to the following system of

[¶¶] as defined in section 4.4.2

linear equations:

$$\left[\begin{array}{cccccc|c} \boldsymbol{G} & \boldsymbol{0} & \cdots & & \cdots & \boldsymbol{0} & \boldsymbol{e}_2 \\ \boldsymbol{0} & \ddots & \ddots & & & \vdots & \vdots \\ \vdots & \ddots & & & & & \\ & & & \ddots & \vdots & & \\ \vdots & & \ddots & \ddots & \boldsymbol{0} & \vdots & \\ \boldsymbol{0} & \cdots & & \cdots & \boldsymbol{0} & \boldsymbol{G} & \boldsymbol{e}_{d+1} \\ \hline \boldsymbol{e}_2^T & \cdots & & & \cdots & \boldsymbol{e}_{d+1}^T & 0 \end{array}\right] \left[\begin{array}{c} \check{\boldsymbol{c}}_1 \\ \vdots \\ \\ \\ \vdots \\ \check{\boldsymbol{c}}_d \\ \lambda \end{array}\right] = \left[\begin{array}{c} \boldsymbol{v}_1 \\ \vdots \\ \\ \\ \vdots \\ \boldsymbol{v}_d \\ 0 \end{array}\right] \tag{4.105}$$

Here, $\boldsymbol{G}$ is the first order Gram matrix as defined by equation (4.42), while the definitions of the vectors $\boldsymbol{e}_i$, $\check{\boldsymbol{c}}_i$ and $\boldsymbol{f}_i$ are given below

$$\begin{aligned} \boldsymbol{e}_i &= [\overset{1}{0}, \ldots, \overset{i-1}{0}, \overset{i}{1}, \overset{i+1}{0}, \ldots, \overset{d+1}{0}]^T \\ \check{\boldsymbol{c}}_i &= [c_{0,i}, \ldots, c_{d,i}]^T = \left[\widehat{u_i}(\boldsymbol{x}_0), \widehat{\partial_{x_1} u_i}(\boldsymbol{x}_0), \ldots, \widehat{\partial_{x_d} u_i}(\boldsymbol{x}_0)\right]^T \\ \boldsymbol{v}_i &= \sum_{j=1}^{M} w(\boldsymbol{x}_j, \boldsymbol{x}_0)\, u_{j,i} \widehat{\boldsymbol{x}}_j \end{aligned} \tag{4.106}$$

Evidently, the size of the system grows with the dimension of the space. In $\mathbb{R}^3$ the number of equations is 13 making a direct matrix inversion almost impossible. Fortunately, as indicated in the above equation, the matrix on the left-hand side can be partitioned into four blocks. This partition enables a blockwise matrix inversion of the form

$$\begin{aligned} &\left[\begin{array}{c|c} \boldsymbol{L} & \boldsymbol{E} \\ \hline \boldsymbol{E}^T & 0 \end{array}\right]^{-1} = \\ &\left[\begin{array}{c|c} \boldsymbol{L}^{-1} - \boldsymbol{L}^{-1}\boldsymbol{E}(\boldsymbol{E}^T\boldsymbol{L}^{-1}\boldsymbol{E})^{-1}\boldsymbol{E}^T\boldsymbol{L}^{-1} & \boldsymbol{L}^{-1}\boldsymbol{E}(\boldsymbol{E}^T\boldsymbol{L}^{-1}\boldsymbol{E})^{-1} \\ \hline (\boldsymbol{E}^T\boldsymbol{L}^{-1}\boldsymbol{E})^{-1}\boldsymbol{E}^T\boldsymbol{L}^{-1} & -(\boldsymbol{E}^T\boldsymbol{L}^{-1}\boldsymbol{E})^{-1} \end{array}\right] \end{aligned} \tag{4.107}$$

Note, the only relevant part of the inverted matrix is the upper left block, since

(1) the last entry of the right hand side vector is zero, and

(2) there is no need to know the Lagrangian multiplier explicitly.

Due to the block diagonal structure of the matrix $\boldsymbol{L}$, its inversion is trivial to compute

$$\boldsymbol{L}^{-1} = \begin{bmatrix} \boldsymbol{G}^{-1} & \boldsymbol{0} & \cdots & & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & & & & \\ & & & & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \cdots & & \cdots & \boldsymbol{0} & \boldsymbol{G}^{-1} \end{bmatrix} \tag{4.108}$$

This also implies

(1) $\left(\boldsymbol{E}^T \boldsymbol{L}^{-1} \boldsymbol{E}\right)^{-1} = 1/\sum_{i=2}^{d+1} \left(\boldsymbol{G}^{-1}\right)_{i,i}$

(2) $\boldsymbol{L}^{-1}\boldsymbol{E} = \left[\overline{\boldsymbol{g}}_2^T, \ldots, \overline{\boldsymbol{g}}_{d+1}^T\right]^T$, $\{\overline{\boldsymbol{g}}_i\}_{i=1}^{d+1}$ are the columns of $\boldsymbol{G}^{-1}$.

(3) $\boldsymbol{L}^{-1}\left[\boldsymbol{v}_1^T, \ldots, \boldsymbol{v}_d^T\right]^T = \left[\breve{\boldsymbol{c}}_1, \ldots, \breve{\boldsymbol{c}}_d\right]_{(uncon)}^T$

(4) $\boldsymbol{E}^T\boldsymbol{L}^{-1}\left[\boldsymbol{v}_1^T, \ldots, \boldsymbol{v}_d^T\right]^T = \left[\widehat{\boldsymbol{\nabla}\cdot\boldsymbol{u}}\,(\boldsymbol{x}_0)\right]_{(uncon)}$

Here, the expression $(uncon)$ labels the coefficients one obtains from the unconstrained, component-wise decoupled first-order MLS minimization problem. Finally, with (1)–(4), the solution of the pseudo-divergence-free MLS at $\boldsymbol{x}_0$ can be written in the form:

$$\begin{bmatrix} \widehat{u}_i \\ \widehat{\partial_{x_1} u_i} \\ \vdots \\ \widehat{\partial_{x_d} u_i} \end{bmatrix}_{(sol)} = \begin{bmatrix} \widehat{u}_i \\ \widehat{\partial_{x_1} u_i} \\ \vdots \\ \widehat{\partial_{x_d} u_i} \end{bmatrix}_{(uncon)} - \frac{\left[\widehat{\boldsymbol{\nabla}\cdot\boldsymbol{u}}\right]_{(uncon)}}{\sum_{j=2}^{d+1}\left(\boldsymbol{G}^{-1}\right)_{j,j}} \overline{\boldsymbol{g}}_{i+1} \tag{4.109}$$

where the subscript (*sol*) stands for solenoidal. The above formula shows that the divergence-free MLS approximation can be computed in two steps:

(1) Solve the standard first-order MLS problem for each component of the unknown vector-valued function $\boldsymbol{u}$. Note, $\boldsymbol{G}$ is only a function of the evaluation point $\boldsymbol{x}_0$ and does not depend on $\boldsymbol{u}$. Therefore, one has to perform one matrix inversion per evaluation point only.

(2) To obtain the pseudo-divergence-free MLS approximation one has to subtract a correction term from the unconstrained MLS approximation. This term can be evaluated from the inverted Gram matrix and the unconstrained solution only. Therefore, the additional computational costs are negligible.

4.8.2 Numerical experiment

The effectiveness of the pseudo-divergence-free MLS method is demonstrated on the basis of a simple two-dimensional numerical experiment. A synthetic solenoidal vector field $\boldsymbol{u} : \mathbb{R}^2 \to \mathbb{R}^2$ can be constructed by the derivatives of a scalar-valued function $\psi : \mathbb{R}^2 \to \mathbb{R}$ (a.k.a. stream function):

$$\boldsymbol{u}(x, y) = \left[\frac{\partial \psi}{\partial y}, -\frac{\partial \psi}{\partial x}\right]^T \tag{4.110}$$

For the simulations at hand, the stream function is assumed to be given by Franke's test function***. The resulting flow field, as depicted in fig. 4.8, is characterized by three eddies of different strength and different orientation. Moreover, the domain in which the velocity field is reconstructed is the unit square. For convenience, the velocity field is scaled such that the maximum velocity in the unit square is equal to one.

In order to imitate a ProCap measurement, the sampling points are located along smoothed random paths as exemplified in fig.

***defined in eq. (4.45)

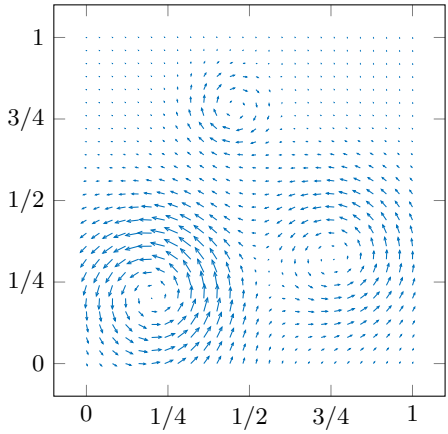


Figure 4.8: Velocity field created by Franke's test function.

4.9. For each numerical experiment a different realization of such a smoothed random path is generated. A short description of how the random paths are constructed is given in appendix E. In accordance with the previous section, the accuracy and numerical stability of the MLS approximation is enhanced by limiting the minimum separation distance. To this end, the input data set is reduced such that the distance between two subsequent measurement points is always larger than 0.01. Furthermore, to be as close to reality as possible, the known velocity data is disturbed by random noise. It is modeled by an independent, Gaussian distributed random vector. Let the noise level (NL) be defined as

$$NL = \frac{\sigma}{|\boldsymbol{u}|_{max}}, \tag{4.111}$$

where σ denotes the standard deviation of the noise in both directions. Note that the maximum velocity magnitude of the undisturbed flow $|\boldsymbol{u}|_{max}$ is one by definition. The noise level is varied between 0 and 50% using 5% increments.

Furthermore, the evaluation points of the MLS approximation lie on a regular grid with spacing of 0.01. Then, the approximation

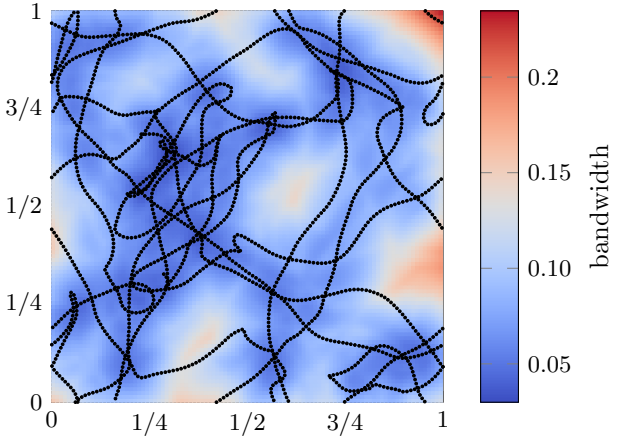


Figure 4.9: Example of a smoothed random path. The color depicts the support size of the weight function so as to contain 30 data points.

error ε may be defined by

$$\varepsilon = \frac{1}{T} \sum_{i=1}^{T} |\boldsymbol{u}(\boldsymbol{x}_i) - \widehat{\boldsymbol{u}}(\boldsymbol{x}_i)| \tag{4.112}$$

where $\boldsymbol{x}_i$ denotes the i-th evaluation point and T is the total number of grid nodes in the unit square. Additionally, let $\bar{\varepsilon}$ denote the error averaged over all random path realizations.

The required weight function w is formed by the Epanechnikov kernel with a locally varying bandwidth. This bandwidth function is defined such that the number of data sites located in the support of every w is fixed by integer M. The contour plot in fig. 4.9 gives an example ($M = 30$) of how the spatial distribution of the bandwidth may look like. As discussed in the previous section, the optimum M depends on the selected approximation order, on the nature of the underlying function, on the distribution of the data sites and last but not least on the noise level. In addition, it is important to note, that

(i) the optimal bandwidth of the standard and the pseudo-divergence-free MLS method do not necessarily have to be the same, and

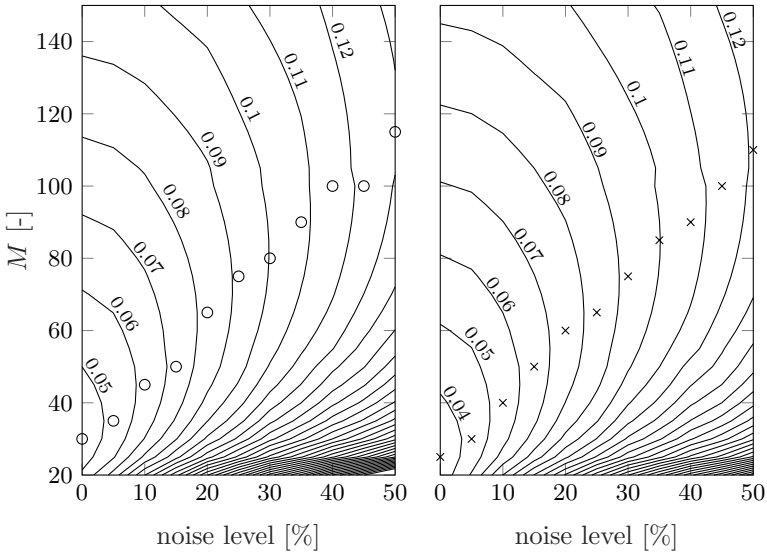
(ii) the integer M, which is in the current ProCap version predefined by the user, is likely to be suboptimal.

Thus, in order to obtain a comprehensive comparison between the standard and the pseudo-divergence-free MLS method, one has to account for the influence of the integer M. So, M is varied between 20 and 150 using an increment of 5. To sum up, for each of the 100 different random paths, both the noise level NL and the integer M are varied, forming a total of 29700 different input data sets. Figure 4.10 shows the distribution of the mean error $\bar{\varepsilon}$ (i.e. averaged over all random path realizations) for the standard and the solenoidal MLS method. The marks $\circ$ and $\times$ reveal that for both methods the optimal bandwidth increases with the level of noise. The contour lines of the average error already suggest that the solenoidal MLS method generally produces better results. To directly compare the two approximation schemes, we introduce an improvement factor denoted as Imp:

$$Imp = 1 - \frac{\varepsilon_{(sol)}}{\varepsilon_{(uncon)}}. \tag{4.113}$$

The subscript (sol) stands for the pseudo-divergence-free MLS method, while $(uncon)$ indicates the standard MLS method. Imp essentially measures the amount of improvement resulting from the solenoidal correction step. In other words, if Imp is positive/negative the pseudo-divergence-free MLS method performs better/worse than the standard MLS method. An Imp of one (100%) means that pseudo-divergence-free MLS method recovers the velocity field exactly. In figure 4.11, the average Imp as a function of NL and M is displayed. The following observations are made:

- On average, the pseudo-divergence-free MLS method reduces the approximation error for all noise levels and all bandwidths. It is important to note that this does not automatically imply that the error always becomes smaller. There might be unfavorable configurations for which the closest pseudo-divergence-free MLS approximation amplifies the noise. Although such a



(a) Standard 1st-order MLS **(b)** Solenoidal 1st-order MLS

Figure 4.10: Contour plot of the average error $\overline{\varepsilon}$ as a function of the noise level (NL) and the number of data sites (M) in the support of the weight function. $\circ$ and $\times$ indicate the optimal integer M for a certain NL

scenario is not impossible, it is highly unlikely. In the 29700 test cases at hand it did not occur once.

- For a fixed noise level, the mean improvement factor $\overline{Imp}$ decreases monotonically with increasing bandwidth. This makes perfect sense as in the limit the reconstructed flow field becomes uniform and therefore divergence-free. This also implies that the optimal bandwidth of the pseudo-divergence-free MLS method is slightly smaller than that of the standard method.

- Interestingly, the highest improvement factor is obtained at zero noise level. A possible but so far untested explanation could be that in the present study, the major error contribu-

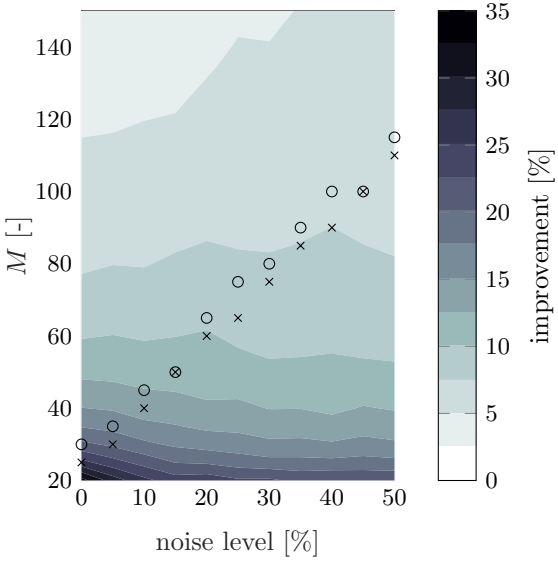


Figure 4.11: Distribution of the improvement factor

tion is induced by the highly non-uniform point distribution and not by the measurement noise. As a consequence, the solenoidal correction step mainly attempts to reduce this error component, which intuitively is more efficient for more accurate data values.

In figure 4.12 the two methods are compared under the assumption that for both cases the optimal bandwidth has been selected. At zero noise level, the average improvement is around 25%, while at 50% noise level the improvement is merely 5%. Note, the filled area depicts the 95% confidence interval of the improvement factor, which as stated above does not cross the zero improvement line. Another strength of the method is indicated by the error bar plot: With the solenoidal approach the variance of the error is slightly reduced. In other words, outliers are expected to occur less often.

To complete this section figure 4.13 shows the result of one realization at zero noise level. The corresponding point distribution

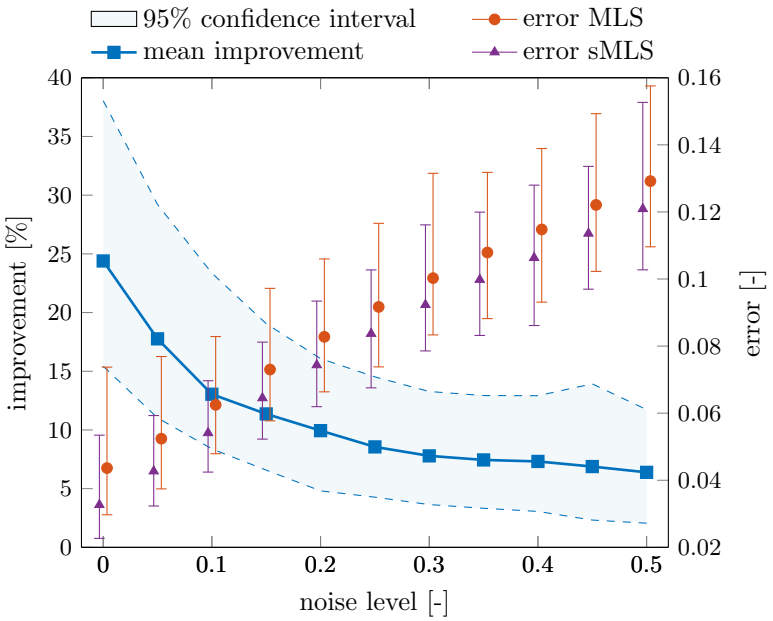


Figure 4.12: Improvement of the pseudo-divergence-free 1st-order MLS over the standard 1st-order MLS method at different noise levels. The filled area and the error bars cover 95% of the realizations.

is given in figure 4.9. The improvement factor is slightly below the average and amounts to approximately 23%. Some regions in which the changes are clearly visible are enlarged by a factor 3. Looking at the point distributions in these selected areas does not give a clear picture of what is causing these changes. A more thorough analysis of the error may shed some new light on that issue.

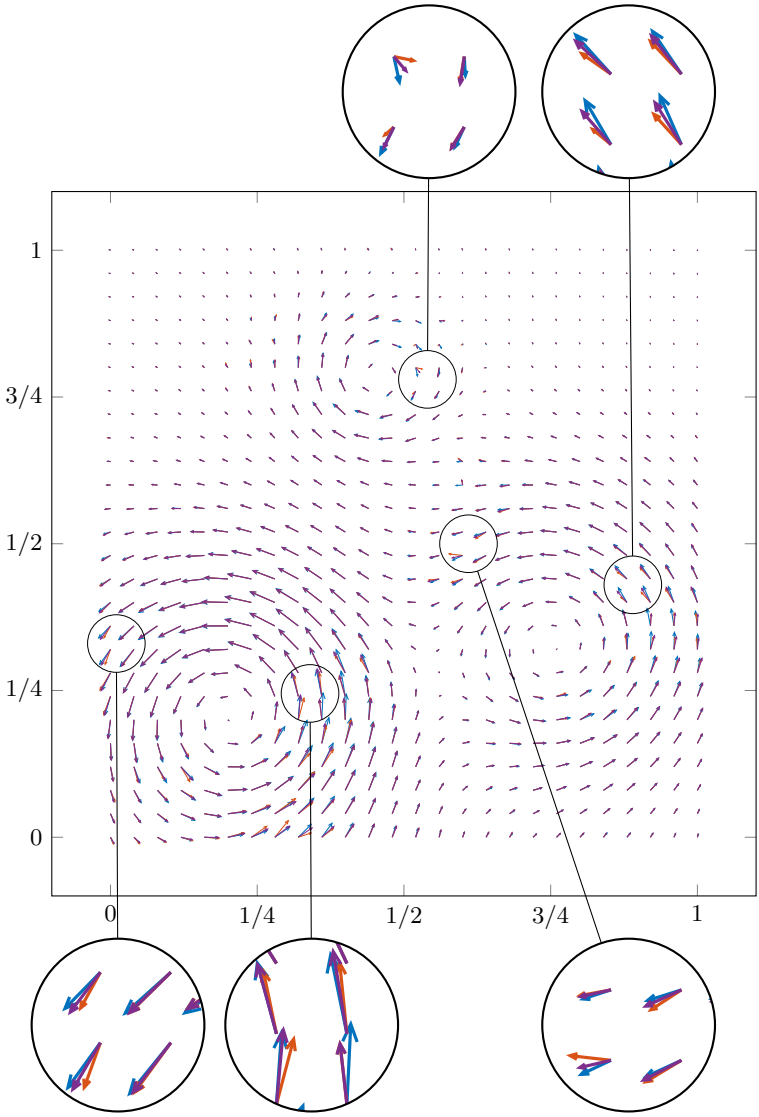


Figure 4.13: → exact, → standard MLS, → solenoidal MLS

Chapter 5

Real-time visualization

The reconstruction of the flow field in real-time is only useful if the acquired data is made accessible to the operator during the experiment to improve the scanning. This refers not only to the measurement accuracy but also to the measurement efficiency (e.g. reduction of the measurement time). In ProCap this is achieved by visualizing the flow topology and the progress of the measurement to the operator. This in turn allows the operator to adapt the handling of the probe in a non-random, determined fashion. Visualization of scientific data is a field of research on its own. ProCap employs simple, well-known visualization concepts such as isosurfaces, streamlines, contour- and vector-slices. Hence, the difficulties to be overcome are not rooted in the visualization methods themselves, but related to the way they are integrated into the ProCap software. Because the spatial interpolation is carried out on the GPU using a compute shader, the interpolated data is stored in a buffer in the global memory of the GPU. Considering the limited data rates between the CPU and the GPU, performing the calculations required for the visualization on the GPU seems to be the most appropriate solution in terms of efficiency. The advantage of this approach is that substantially fewer data has to be exchanged between the two processing units. Thus, this approach allows us to preserve the real-time functionality of the system.

This chapter provides a brief outline of the applied visualization concepts and their implementation into ProCap using Microsoft's DirectX graphics API. The aim of the first part is to shed some light on the DirectX rendering pipeline, on its stages and on the DirectX GPGPU ansatz. As we are just going briefly through these concepts, for details the reader is referred to more comprehensive literature on computer graphics. The second part covers the visualization methods and their implementation into ProCap using the techniques introduced in the first part.

5.1 Shaders

Computer programs that determine how an object is rendered on a display are usually referred to as shaders. Typically, shaders run on a GPU and are attached to a stage of the rendering pipeline. While the first video cards were fixed-function processors, today's GPUs have turned into powerful programmable processors, allowing not only customization of the shading but also general purpose computations (GPGPU). Microsoft's graphics API, Direct3D 11, offers two different shader types (cf. figure 5.1): Shaders that are attached to a stage of the rendering pipeline and shaders that run independently, i.e. that are not attached to any stage of the rendering pipeline. The latter are commonly known as compute shaders. In Direct3D, the language used to program these shaders is usually Microsoft's high-level shader language, commonly abbreviated as HLSL.

5.1.1 Shaders attached to the rendering pipeline

As shown in figure 5.1 the rendering pipeline comprises several stages:

- **Input assembler:** The first stage of the DirectX rendering pipeline is the input assembler, which is essentially responsible for three tasks:
  1. read data from user-filled buffers (geometry data)
  2. assemble data into primitives, e.g. line lists, triangle lists etc.
  3. attach system-generated values such as vertex ID, primitive ID, instance ID etc. to the data

  To date, the input assembler is not programmable.

- **Vertex shader:** As the name implies the second stage in the rendering pipeline processes the vertices passed in by the input assembler. Generally, the main task of the vertex shader is the transformation of the coordinates into the camera space. Not exclusively however, as other, user-defined transformati-

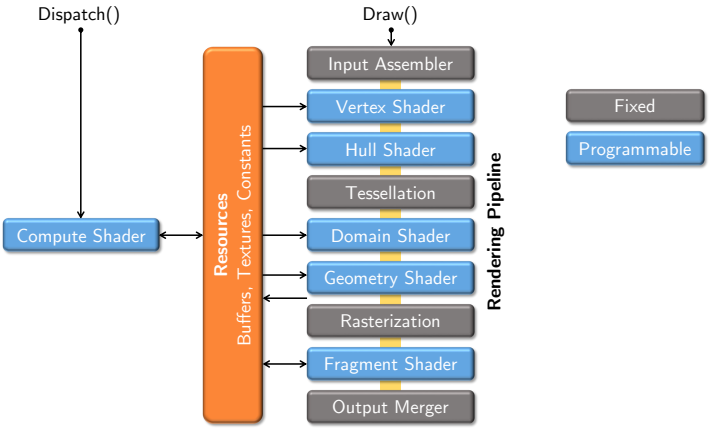


Figure 5.1: Shader types under Direct3D 11

ons such as morphing or skinning can be applied. It is important to recognize that the term vertex is not limited to coordinates but may also contain other per-vertex data such as information about the lighting, the color, the texture and the surface normal. Thus, other per-vertex operations, for instance related to lighting or coloring, may be performed. While the output vertex structures may be different from their input counterparts, the deletion and diversification of vertices are not possible. This means for each input vertex there is exactly one output vertex. Since the number of vertices is normally lower than the number of pixels it is better to perform the computationally expensive operations at this stage rather than in the fragment shader.

- **Tesselation stages (hull shader, tessellation, domain shader):** Unlike the vertex shader, the three tessellation stages are optional. These stages allow you to add more details to low-resolution models by subdividing the primitives into smaller pieces. Improving the model quality by means of these shader stages has a number of advantages:

1. Saving computational costs as expensive calculations (physics, lighting) can be done on the level of the low-resolution model.
2. Saving memory and bandwidth. In particular, the data transmission traffic between the CPU and the GPU is reduced.
3. Support of scalable-rendering, e.g. levels-of-detail can depend on the viewpoint.

As these processing stages have not been used in the ProCap visualization software we do not go further into details and refer the interested reader to other, more comprehensive sources.

- **Geometry shader:** Like the tesselation, the geometry shader stage is optional. The geometry shader function is invoked once for every primitive (e.g. triangle, line, point). Besides the primitive ID provided by the input assembler, one has access to all vertices of the primitive being processed. In addition to that, the vertex data of the edge-adjacent primitives are also accessible. Unlike the vertex shader, the geometry shader stage allows you to define a new set of vertices provided that they belong to the same topology (e.g. triangle strip, line strip, point strip). Thus, contrary to the vertex shader diversification of vertices is possible. However, it is required that the number of output vertices is fixed for all primitives. In a final step, the output of the geometry shader can either be streamed to the rasterization stage or one can activate the so-called stream output stage, where the vertices are simply stored in a vertex buffer on the GPU memory.

- **Rasterization:** The rasterization stage combines the fixed-function steps required to map the vertex and primitive data to a 2D raster image. A raster image is essentially a matrix of pixels (a.k.a. fragments). The rasterization process may include the following operations:

  1. Vertex clipping: Consider a primitive that lies partly inside and partly outside the view frustum. Clipping is the process that first splits this primitive into several smaller

primitives and then discards all primitives that lie outside the view frustum.

2. Face culling: Primitives facing away from the viewer (camera) are discarded without being rendered.
3. Perspective division: The primitives are projected on the 2D viewport. In doing so, the affected fragments are tagged for further processing.
4. Interpolation: The per-vertex data (e.g. color, texture coordinates) is interpolated onto the affected segments. Although not programmable, the user can specify whether and how the interpolation shall be done, e.g. interpolation in the projected space or in the 3D space.

- **Fragment shader:** Another optional and programmable stage of the rendering pipeline is the fragment shader (a.k.a. pixel shader), which enables the programmer to apply per-fragment operations, such as lighting, color mapping. The output of the fragment shader essentially consists of two buffers: the color buffer that stores the color of the fragments and the depth buffer that informs the output merger about the distances of the processed fragments from the camera.

- **Output merger:** In Direct3D the last step in the rendering pipeline is the output merger. This stage determines the final color of a pixel that is written to the frame buffer. First, a fragment has to undergo a number of tests, which determine whether and how a pixel is updated. Commonly, the following tests are done:
  - Pixel-ownership-test
  - Stencil-test
  - Depth-test

  Which tests are actually active depends on the pipeline state, i.e. they are user-selectable. The pixel color in the frame buffer is only changed if the corresponding fragment passes all activated tests. The color assigned to a pixel in the frame

buffer may also depend on the previous pixel color, e.g. translucent surface. This mixing process is commonly known as color blending.

5.1.2 Compute shader

Compute shaders provide a way to carry out work not only in parallel but also detached from the rendering pipeline. An advantage of compute shaders compared to other GPGPU solutions (Cuda C, OpenCV) is that they allow interaction with the rendering process in a straightforward fashion. More precisely, they consume and produce memory resources that can also be accessed by the draw calls.

A compute shader divides its work into a 3D grid of thread groups. Each thread group consists of a 3D grid of threads and each thread executes the compute shader's kernel function once. A thread group is assigned to a processor unit sometimes called multiprocessor. Internally, the thread group is split into so-called warps (a.k.a. wavefronts). On nVidia cards the warp size usually corresponds to 32 threads. Therefore, it is recommended that the number of threads in a thread group is a multiple of the GPU's warp size. Unlike CPU-threads, threads within a warp share a single instruction unit, i.e. they are not independent and at best they execute the exact same instructions on a different set of data. Therefore, in order to fully utilize the computational power of a GPU, code branching within a warp should be avoided. It is worth mentioning that threads belonging to the same thread group can not only be synchronized but also share on-chip memory (with read and write permission) known as shared memory. In today's video cards, the shared memory size is typically limited to 32 KB and is divided into 32 memory banks (on compute shader 5.x). Normally, a memory bank can be accessed only by a single thread per cycle. If this rule is violated the memory access will be serialized. This artifact is commonly referred to as bank conflict. The main advantage of shared memory is that it can be accessed about hundred times faster (i.e. approx. 2-3 cycles) than off-chip global memory provided that no bank conflict occurs.

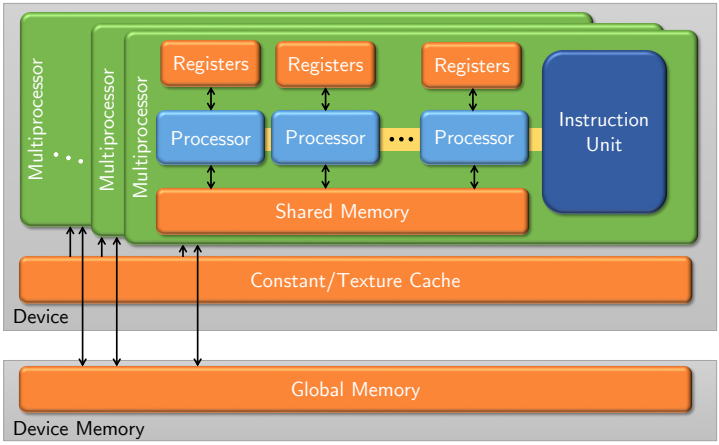


Figure 5.2: Compute shader architecture

5.2 Visualization features

5.2.1 Isosurfaces

Isosurfaces are useful data visualization methods to study the three-dimensional features of a flow. An isosurface is given by the collection of points that have the same value of a scalar function. Today's GPUs are powerful processors mainly in real-time rendering of polygonal surfaces. From this perspective, a method that computes the isosurface as a triangulation is preferred. A simple, but nevertheless efficient approach is the so-called marching cubes algorithm (*Lorensen and Cline*, 1987). As the name suggests the algorithm marches through a set of cubes. Regarding ProCap, the cubes are formed by the regular grid whose nodes are given by the center points of the interpolation voxels (staggered grid). It is vital for the algorithm that the value of the scalar quantity is known at the corners of the cubes. Now, each of the cube's 8 vertices can either be smaller or larger than the predefined isovalue. Thus, there are in total $2^8 = 256$ different configurations how the isosurface can intersect

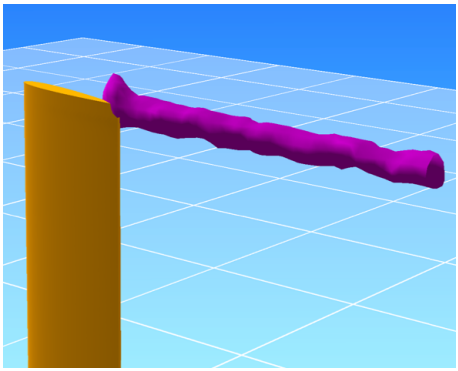


Figure 5.3: Isosurface

a cube. Naturally, this statement is only true if we assume that the scalar function between two edge-connected nodes changes monotonically. By symmetry, one can reduce the number of configurations to 15. If an edge intersects the isosurface (i.e. one vertex value is larger and the other is smaller than the isovalue), the resulting intersection point is used as a vertex of the triangulated isosurface. The exact position of the intersection point is determined by linearly interpolating the scalar function between the two corresponding corners of the cube.

Responsible for the efficiency of the marching cubes algorithm is the fact that apart from the interpolation almost all operations can be bypassed by look-up tables.

The marching cubes algorithm is embedded into ProCap by means of a compute shader that is invoked once per frame. To render the isosurface one has to call a second shader that is attached to the rendering pipeline and uses the triangle data produced by the compute shader. This segregated design allows you to restrict the marching cubes algorithm to the cubes that have been affected by the previous interpolation step. While this approach may save computing time it is not optimal regarding the memory consumption. To prevent memory overflow ProCap limits the number of displayed isosurfaces to one. Alternatively, instead of using a compute shader the marching cubes calculations could also be carried out by a cu-

stomized geometry shader. This ansatz possibly reduces the access requirements on the memory but increases computing time. This is due to the fact that in the most natural algorithm design the geometry shader will update all cubes instead of only those affected by the interpolation.

Improvements in the rendering quality of the isosurface (e.g. lighting, smoothness) often require that the normal vectors at the triangles' vertices are known. A simple way of estimating the normal vector at a vertex is to interpolate the gradient field of the scalar quantity along the cube's edge. In ProCap the gradient field can be created either by using the diffuse derivatives or by applying a finite difference scheme on the interpolated data (additional compute shader). To be less dependent on the interpolation method the latter approach was selected.

5.2.2 Contour slices

As shown in figure 5.4 contour slices display the color-coded distribution of a scalar quantity in a plane. In ProCap they are created by adjusting the vertex and fragment shader accordingly. The only inputs of the vertex shader are the vertex positions that describe a user-defined triangulated plane. This triangulated plane is built beforehand on the CPU by a layer of regularly distributed points. To prevent oversampling, the grid distance of these points is set equal to the size of the voxels used for the interpolation.

Before we turn our attention to the operations carried out by the vertex shader, we look at the format of the vertex shader's output. The per-voxel output of the vertex shader is a structure consisting of two variables:

- P: position of the vertex with respect to the camera-view coordinate system

- UV: 2D vector, where the first component indicates whether the vertex lies in the domain of interest or not and the second component is simply the value of the scalar quantity at that particular location.

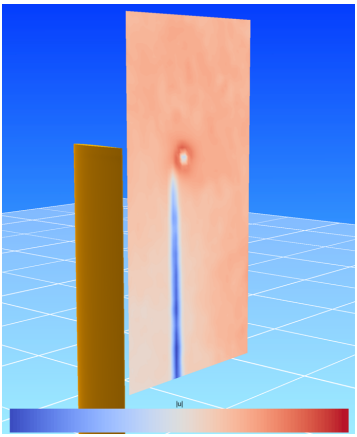


Figure 5.4: Contour slice

In the vertex shader itself the following per-vertex operations are performed:

1. **Coordinate transform 1:** To ensure fast access to the interpolation data and to allow clipping, the coordinates of the vertex position are transformed to the normalized domain space. In the normalized domain space, the domain of interest is the unit cube.

2. **Clipping:** In the normalized domain space is it trivial to detect whether the vertex at hand lies in the domain of interest or outside. If it is inside, the first component of the output variable UV is set to zero, else to one.

3. **Identification of the relevant interpolation nodes:** As described in the previous chapter, the domain of interest is decomposed into a grid of voxels. As the interpolated data is stored at the center of these voxels one can define a second, staggered grid where the interpolated data points are not the centers but the corners of the grid cells. Assuming that the vertex is contained in the domain of interest the shader has to accomplish two simple tasks: Find the cell in which the vertex

is located and then extract the corresponding corner values from the interpolation data buffer.

4. **Trilinear interpolation:** By means of these eight data points at the corners of the cell, the value at the vertex is calculated by trilinear interpolation. The computed value is then assigned to the second component of the output variable UV.

5. **Coordinate transform 2:** Subsequent to the trilinear interpolation, the vertex position is transformed to the camera-view space and then passed to the output object P.

Right after the vertex shader the vertex and primitive data go through the rasterization stage. Since the input plane is only single-layered but has to be visible regardless of the viewing direction, face culling is deactivated.

On completion of the rasterization the fragment shader is invoked. First of all, all fragments are discarded for which the first component of the UV variable is nonzero. Next, the color of the fragment is determined. Thereby, the second component of UV is linearly mapped to the colors of the colorbar. This mapping is fully described by the user-specified minimum and maximum value and by the selected 1D colorbar texture. The reason why the color mapping is performed in the high resolution fragment shader rather than in the low resolution vertex shader is that with this approach the boundary between two contour levels remains sharp and does not become blurred by the interpolation in the rasterization stage.

5.2.3 Vector slices

In ProCap vector slices are visualization objects that display the direction of the flow (cf. figure 5.5). Naturally, this requires that the probe used is capable of measuring all three components of the velocity. Similar to contour slices vector slices are generated by adaptation of the vertex and fragment shader. In addition to that, a few computational steps have to be performed in the geometry shader stage.

In contrast to contour slices, the input primitives are not triangles but just individual points. The output of the vertex shader comprises:

- the start- and endpoint of the vector associated with the vertex being processed (represented in camera-view coordinates)
- the size and the orientation of the arrow head.

Briefly, the vertex shader is responsible for the following operations:

1. **Vector extraction:** At each vertex the vector to be visualized is determined. To this end, we employ the same approach as in the contour slice shader, i.e. find the cell in which the vertex is located, extract the corresponding corner values from the interpolated data buffer and use for each component the trilinear interpolation scheme to get a local estimate of the vector. If the vertex does not lie in the domain of interest or no interpolated data is available a degenerated arrow is returned.
2. **Vector manipulation:** For visualization reasons the user can turn on or specify several vector manipulations: 1.) Projection of the vector on the plane, 2.) normalization and 3.) scaling of the vector's length. If the vector is projected on the plane the orientation of the arrow's head is given by the normal of the plane otherwise the normal of the head is set parallel to the viewing direction.
3. **Coordinate transform:** Transformation of the per-vertex data to the camera-view space.

In the geometry shader, each vertex primitive is replaced by a line strip that forms the arrow to be rendered. To prevent backface culling the line strip is two-layered. After the rasterization, the fragment shader simply sets the color of the involved pixels to black.

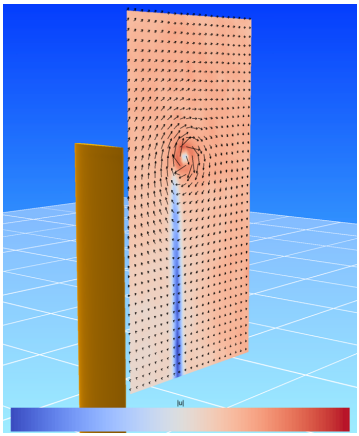


Figure 5.5: Vector slice

5.2.4 Streamlines

Another way to visualize the three-dimensional topology of the flow field are streamlines (cf. figure 5.6). In case of a stationary flow, a streamline corresponds to the trajectory of a fluid particle denoted as $\boldsymbol{\xi}(t)$. $\boldsymbol{\xi}(t)$ is described by an initial value problem

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\xi}(t) = \boldsymbol{u}(\boldsymbol{\xi}(t)), \quad \boldsymbol{\xi}(0) = \boldsymbol{\xi}_0, \tag{5.1}$$

where $\boldsymbol{u}$ is the velocity as a function of space, t is the time and $\boldsymbol{\xi}_0$ is the position of the fluid particle at $t = 0$. From a numerical point of view, parallelization of algorithms that solve this kind of initial value problem is not trivial (*Burrage*, 1993). Hence, The decision was made to parallelize not the computation of a single trajectory but the computations of several trajectories. The calculation of the streamlines itself is completed by means of a compute shader. In the current design, the number of threads per group is set to 32, which corresponds to the warp size of most of today's GPUs. Each thread of the compute shader is responsible for the calculation of one streamline. To keep the computational costs low the time inte-

gration relies on an explicit Euler scheme. Moreover, the number of time steps and therefore the number of streamline segments is limited to one hundred. Per streamline segment one particle is drawn. For animation reasons its position is not fixed but moves along the segment. The speed of this movement can be adjusted by the user. The output of the compute shader is a buffer storing the particle positions at a certain time. To render the particles the point data has to pass the stages of the rendering pipeline. The vertex shader uses the per-vertex IDs to retrieve the particle positions from the output buffer of the compute shader. At the geometry shader stage each point primitive is replaced by two triangles forming a square. The normal of this square is set parallel to the viewing direction while the size of the square can be adjusted by the user. Additionally, to each corner of the square a 2D texture vector is ascribed. These texture vectors simply correspond to the corner coordinates of the unit square and are subsequently used in the fragment shader to generate the *pseudo-spherical* shape of the particles. An advantage of the fact that the streamlines are updated once per frame is that the initial point $\boldsymbol{\xi}_0$ can change its position. For instance to enhance the functionality of the probe one can attach the initial point of a trajectory (or even more) to a point on the probe. By doing so the probe is both a flow sensing instrument but also some sort of a virtual smoke probe that can be used for flow diagnostics.

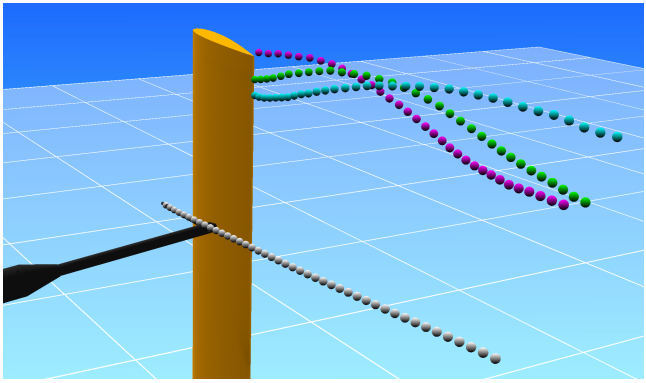


Figure 5.6: Streamlines

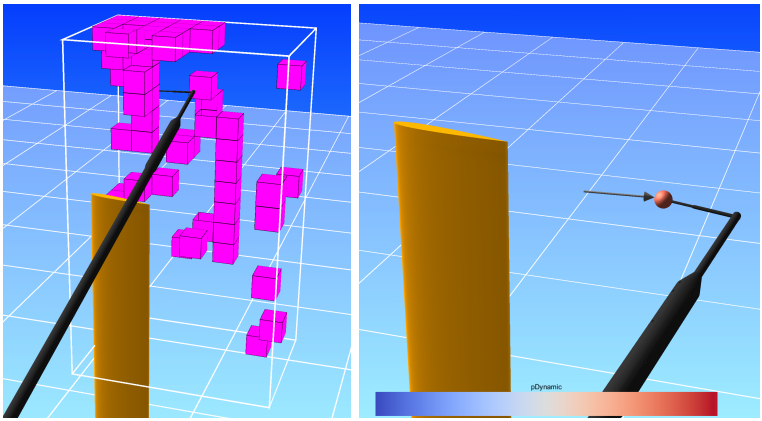


Figure 5.7: Left: Voxel eraser. Right: Current-state-visualizer

5.2.5 Visualization techniques decoupled from interpolation

Independent of the interpolation, two other helpful measurement visualization techniques are available in ProCap (cf. figure 5.7). Since these two methods do not depend on the interpolated flow field the underlying geometries are created on the CPU. For the rendering standard, built-in shaders are applied.

- **Voxel eraser**: The voxel eraser splits the domain of interest into a set of uniform voxels. The size of these voxels can be adjusted by the user and is typically larger than those used for the interpolation. A voxel disappears as soon as the number of samples within this volume exceeds a user-defined threshold. To put it another way, a rendered voxel indicates a region where the measurement point density is low. Thus, a possible measurement strategy for uniform scanning is to erase all voxels. At first, the size of the voxels shall be set large but as the measurement progresses one is encouraged to reduce the size in order to spot under-resolved zones.

- **Current-state-visualizer**: By means of the current-state-visualizer ProCap provides a way of visualizing the latest sensor data. A colored sphere attached to the probe's head displays the most recent value of a measured, scalar quantity, e.g. static or dynamic pressure. In addition to the colored sphere, the probe is also equipped with a virtual arrow that indicates the direction of a user-selected vector quantity (e.g. velocity).

Chapter 6

Results/Measurements

To demonstrate the performance and versatility of the ProCap system, this chapter presents measurement results for three different test cases, namely:

- Tip vortex of a symmetric NACA wing
- C-pillar vortex of the Ahmed body
- Flow field around a sailing yacht

The first two measurements were carried out in the large subsonic wind tunnel of the Institute of Fluid Dynamics, ETH Zurich. To assess the accuracy of the ProCap measurements, the reconstructed flow fields are compared to flow data acquired with the built-in traversing system. The third experiment showing the flow around a sailing yacht was conducted in the twisted flow wind tunnel (TFWT) of the University of Auckland.

6.1 Test facilities

6.1.1 Large subsonic wind tunnel

The large wind tunnel of the Institute of Fluid Dynamics is a closed return wind tunnel built in 1935 under the direction of Prof. Jakob Ackeret. The side view of the tunnel is illustrated in schematic form in figure 6.1. The air flow is generated by two fans, each powered by a $175kW$ electric motor. Before the air reaches the test section, it passes a screen and a flow straightener (honeycomb) to create a uniform laminar inflow. In the corners, where the flow changes its direction, an array of vanes prevents the growth of flow instabilities. The converging nozzle located upstream of the test section not

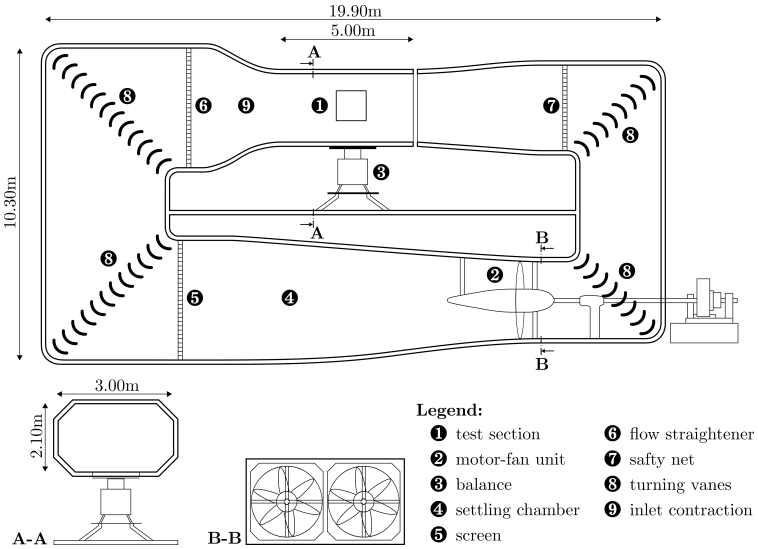


Figure 6.1: Large subsonic wind tunnel at ETH Zürich

only increases the wind speed but also has a stabilizing effect on the flow (favorable pressure gradient). Normally, the wind speed is determined by means of the pressure drop across the inlet contraction. Under normal conditions, the maximum flow velocity is around $60m/s$, which corresponds to a Mach number of approximately 0.17. The test section itself has the dimensions $2.1 \times 3 \times 5m$ (height×width×length). To reduce the effect of the growth of the boundary layer on the core flow, the cross-sectional area of the test section widens slightly in the flow direction. The test objects are usually mounted on a turn-table which allows the operator to change the angle of flow incidence. Moreover, for the precise measurement of the aerodynamic forces, the turn-table is mounted on a stationary 6-component piezoelectric force balance. The test section is closed, but is, if required, accessible through three windows on the top, front and rear wall. In addition, antireflective glass windows in the sloped part of the ceiling allow for optical access without disturbing the flow. Furthermore, to obtain volumetric flow data from point

measurements, the wind tunnel is equipped with an integral 3-axes traversing system.

The ETH wind tunnel can be operated in two different modes, either stagnation pressure regulated or rotation-speed regulated. All experiments carried out for this work are based on the latter operational mode.

6.1.2 Twisted flow wind tunnel

As the air flow around a sailing yacht is rather complex, wind tunnel testing is of crucial importance for design optimization. To make wind tunnel measurements of sailing yachts more reliable, the Yacht Research Unit of the University of Auckland built in 1994 the first twisted flow wind tunnel (*Flay* (1996)). As indicated in figure 6.2 the wind experienced by a sailing boat (a.k.a apparent wind) is the superposition of the true wind and the head wind ($\widehat{=}$ inverse of the boat velocity). The magnitude of the true wind varies with the distance to the surface of the water (atmospheric boundary layer) resulting in a vertical twist of the apparent wind profile. To simulate this twist the twisted flow wind tunnel (TFWT) at the University of Auckland uses an array of flexible vanes, which are located $1.5m$ up-

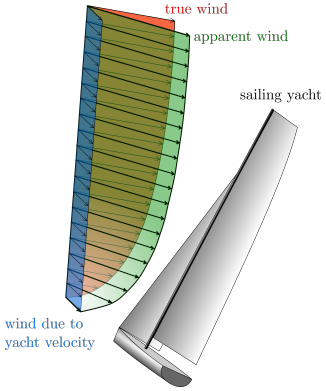


Figure 6.2: Twisted flow profile

stream of the test section (see figure 6.3). The shape of these vanes can be mechanically adjusted allowing one to change the strength of the vertical flow twist (max. 40°). Moreover, to properly simulate the atmospheric boundary layer the TFWT contains a comparatively long settling chamber ($18m$) and a set horizontal bars spanning the width of the tunnel. By adjusting their heights one can change the flow such that it conforms to the target flow profile. Typically, the boundary layer has a height of about $50cm$ and a turbulence intensity between 15 and 5%. In the bulk flow the turbulence level is approximately 4%.

The TFWT is operated as a blow down, open return wind tunnel. The two $75kW$ fans can produce wind speeds of up to $8m/s$. The course and heel angle of the boat can be adjusted by the remote controlled turn table. Moreover, to eliminate artifacts caused by the air flow around the hull of the ship, the model is immersed in a basin of water. The forces acting on the model are measured by a built-in 6-component force balance.

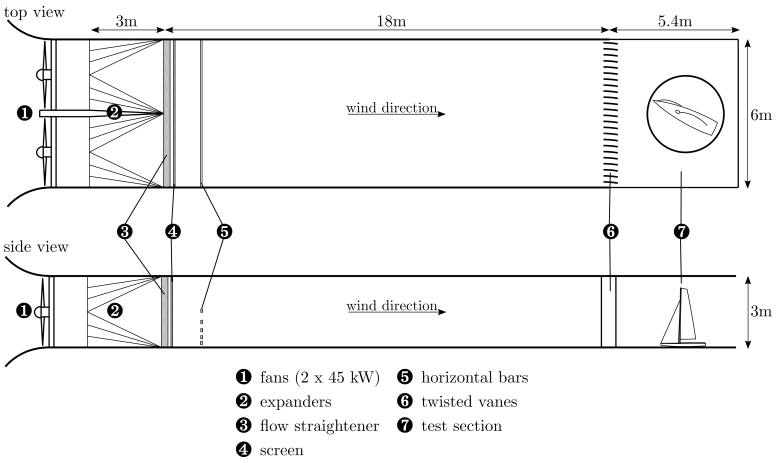


Figure 6.3: Twisted flow wind tunnel at University of Auckland

6.2 Test cases

6.2.1 Tip vortex of a NACA airfoil

An interesting and well-studied flow phenomenon is the tip vortex generated by a wing of finite size. Here, a symmetric wing of length $1m$ is installed vertically at the center of the wind tunnel's test section (see figure 6.4). The profile has a constant chord length of $200mm$ and its shape corresponds to a standard profile from the NACA series, namely the NACA0012 airfoil. The number 0012 signifies that the profile is symmetric and that the ratio of thickness to chord is 12%. For this experiment, the wing model is tilted at an angle of 10° to the flow inducing a strong tip vortex. As shown in figure 6.4, the domain of interest is set about $150mm$ downstream of the trailing edge and measures $100\times420\times800mm$ (length×width×height). The wind speed u_∞ is $12m/s$. Thus, the five-hole probe is fitted with the HCLA02X5EB pressure transducers which have a range $\pm250Pa$.

For the ProCap measurement the domain is divided into 268'800

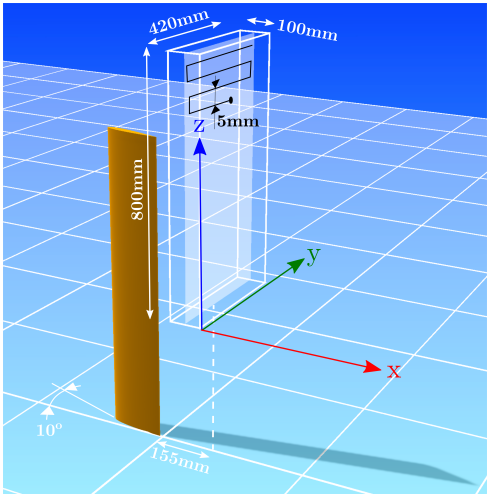


Figure 6.4: Measurement domain for NACA0012

cubic voxels of edge length $5mm$. The technique used to interpolate the data onto a regular grid is the adaptive first-order moving least squares method described in chapter 4. The interpolation parameters are set as follows:

- The samples are considered to be uncorrelated, therefore the integral time scale $M_{\mathcal{T}}$ is equal 0.5.
- The noise level L is assumed to be 0.5.
- The certainty threshold value s defining the kernel bandwidth is set to be 20.

The data is acquired at a frequency of $100Hz$. To cover the whole domain of interest, the measurement takes about 1200 seconds to complete. For a ProCap scan this is relatively long, and can be explained by the time it takes to resolve the fine structures found in the flow field.

Similar to the ProCap measurement, the traverse is used in conjunction with the same five-hole probe (HCLA02X5EB pressure transducers). However, only one y-z-plane at the center of the domain is scanned. To further reduce the measurement time, the traversing aperture is programmed to scan the plane continuously at a speed of $10mm/s$. The scanning paths are vertically layered with a separation distance of $5mm$. Hence, the measurement time for one scan adds up to approximately 7200 seconds. Although only one plane is measured, this is about six times longer than the volumetric ProCap measurement. As before the acquisition rate of the probe signal is $100Hz$ resulting in 50 samples per $5mm$. To reduce the noise of the measurement these 50 samples are averaged together and replaced by a single measurement point.

Figure 6.5 compares the distribution of the scaled velocity magnitude (i.e. $|u|/u_\infty$). Not surprisingly, both methods are capable of capturing the two dominant flow structures present in the domain of interest, namely the tip vortex and the sharp vertical wake in the velocity profile. While the locations of these structures are practically identical, some differences are found with respect to their strengths. The ProCap measurement tends to smooth out sharp features (roll-up of the tip vortex, depth and thickness of the wake).

This behavior becomes evident by examining the distribution along a line as shown in the two graphs in the lower half of figure 6.5). The effect can be explained by the following facts:

- due to the shorter measurement time and the larger volume to be covered, the density of the measurement points with ProCap is considerably lower than that of the traversing scan
- the selection of the interpolation parameters (certainty threshold and voxel size) is based on a trade-off between robustness and spatial resolution. The spatial resolution can be improved by using more aggressive interpolation settings. However, this leads to considerably longer measurements and bears the risk of creating ill-conditioned interpolation problems.

As illustrated in figures 6.6 to 6.8, the smoothing effect appears to have a larger impact on the main-flow component of the velocity than on the cross-flow components. When expressed in absolute numbers, the differences in the cross-flow components are insignificant. In closing, it is worth mentioning that the probe tip size is not ideal considering the flow structures being observed. Moreover, the scan speed of the traverse should be reduced since the average values in the interesting flow regions still seem to be biased by noise (measurement noise, flow fluctuations).

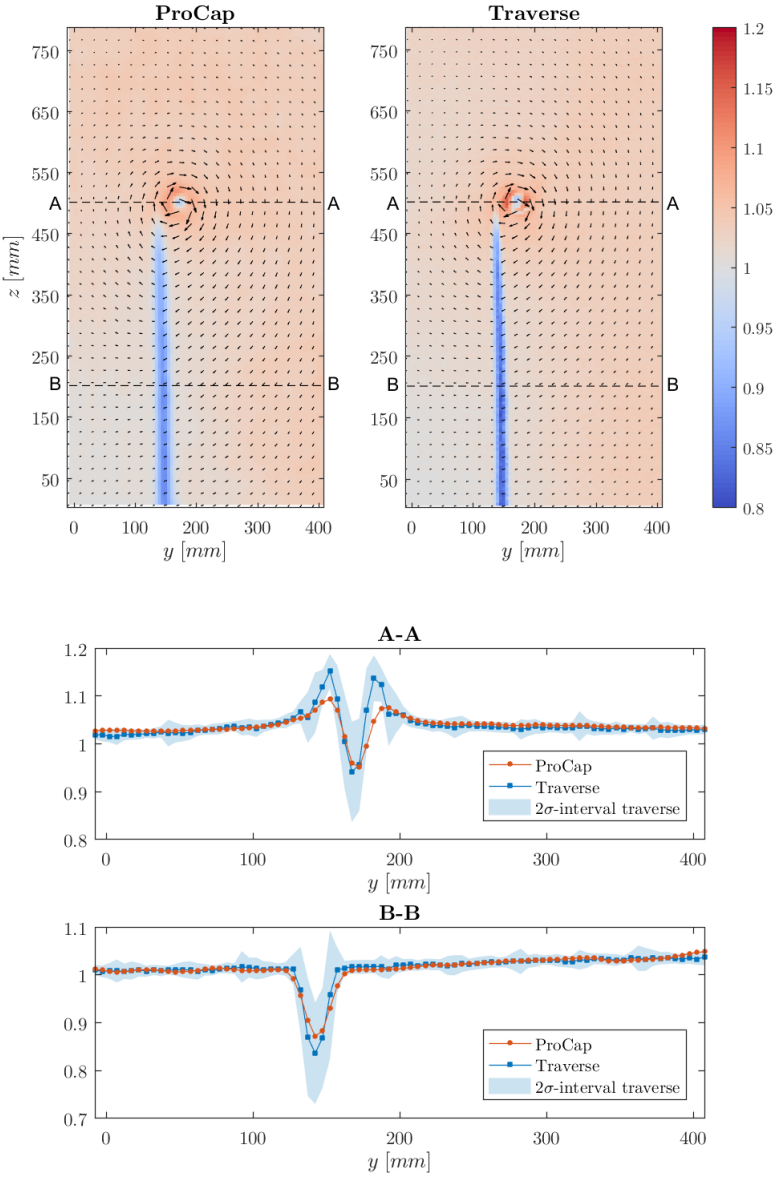


Figure 6.5: Comparison of the scaled velocity magnitude, i.e. $|u|/u_\infty$

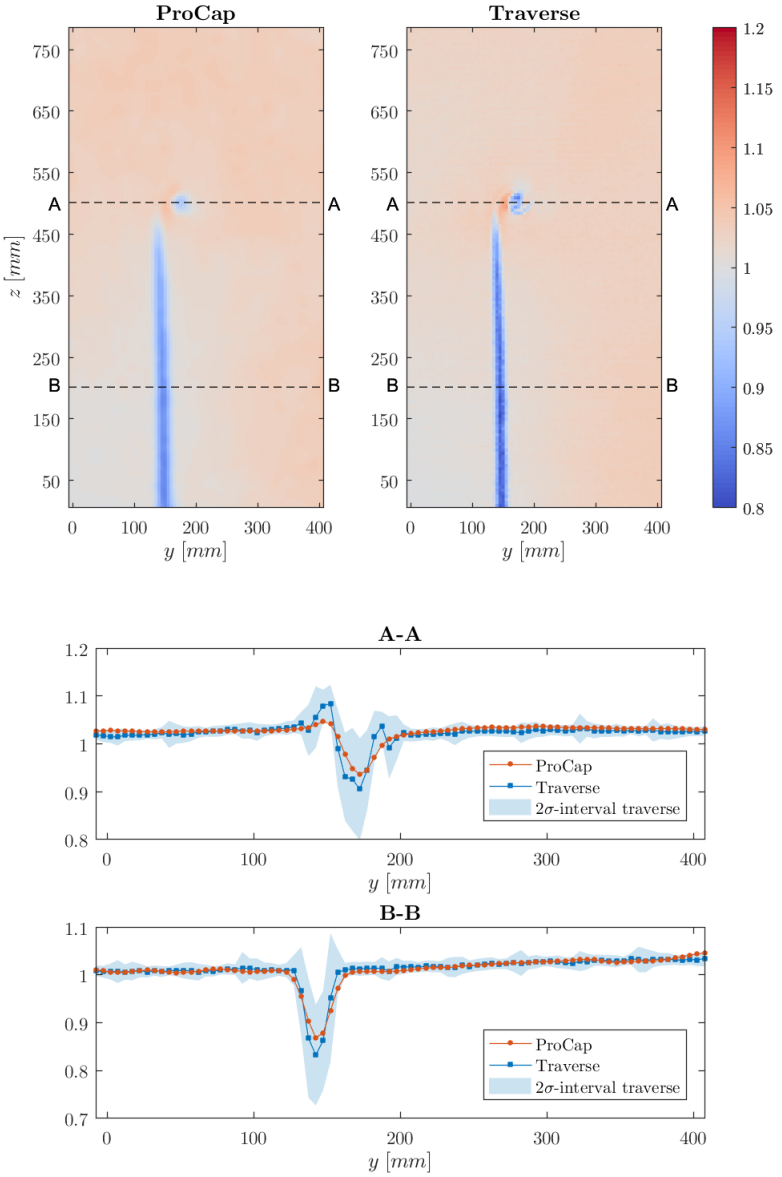


Figure 6.6: Comparison of the scaled velocity in x-direction, i.e. u/u_∞

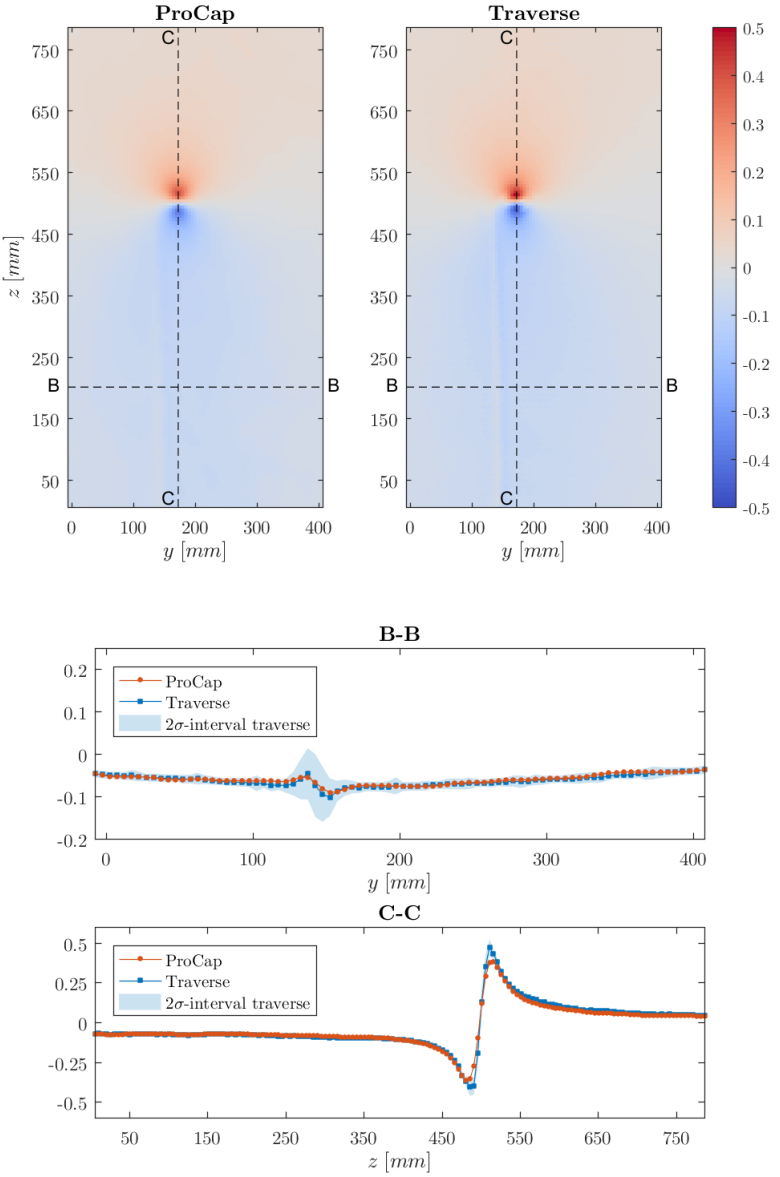


Figure 6.7: Comparison of the scaled velocity in y-direction, i.e. v/u_∞

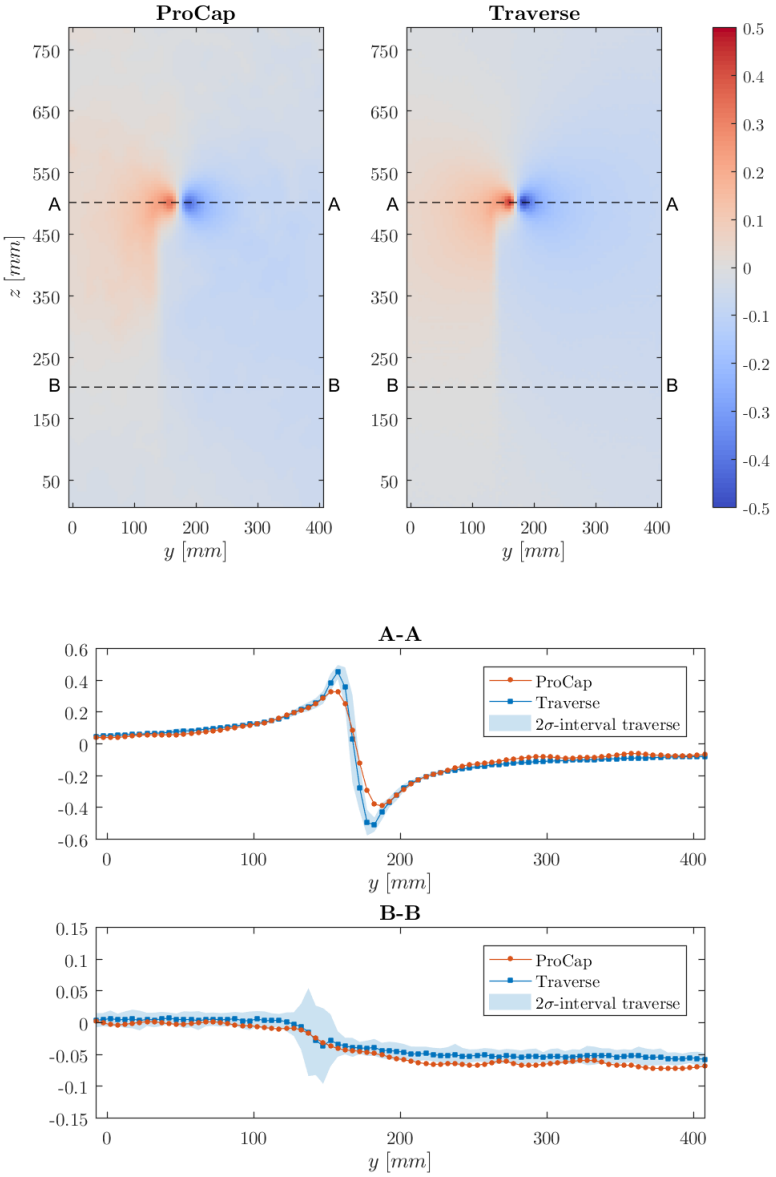


Figure 6.8: Comparison of the scaled velocity in z-direction, i.e. w/u_∞

6.2.2 C-pillar vortex of the Ahmed body

Often the scan of a single plane using a traversing system is insufficient to fully understand the three-dimensional topology of the flow. This is particularly true for flows around complex geometries such as the Ahmed body (*Ahmed et al.*, 1984). The Ahmed body is a standardized bluff body shape specifically designed to simulate the flow around a road vehicle. The schematic in figure 6.9 shows the geometry and dimensions of the body. It is known that the angle of the rear slant φ has a major influence on the global flow pattern. In this work φ measures 25° resulting in a rather complex, highly three-dimensional flow field. If no angle of incidence is applied the flow direction coincides with the x-axis introduced in figure 6.9. For both measurements the incident flow velocity is fixed at $18m/s$. The wake flow is characterized by two vortices generated at the lateral edges of the slanted surface (a.k.a. C-pillar vortices). In addition, the flow over the slanted surface is partially separated, to be precise, there is a thin separation bubble near the centerline. At the rear edge of the slanted surface the flow becomes completely detached, thereby two counter-rotating separation zones are formed. Close to the ground the flow contains a number of smaller, relatively complex flow structures generated by the legs of the body, the ground floor and the rear end of the body. However, this is irrelevant as the measurement domain covers only one C-pillar vortex and part of the separated flow (cf. figure 6.10).

To accommodate the prevailing flow conditions, the five-hole probe is fitted with the $250Pa$ pressure transducers (HCLA02X5EB). Since the acceptance angle of the five-hole probe at hand is limited to $\pm 60°$, reverse flow, as it is seen in the separated flow regions, cannot be properly captured. This implies that the probe processes only pressure data that fall into the valid range of operation, while "invalid" pressure readings are simply ignored. Consequently, the measurement in these regions is strongly biased. This circumstance underlines the need for fully omnidirectional flow sensors.

For the measurement using the traversing system, the domain of interest is split into 19 parallel y-z-planes (see figure 6.10). The distance between two neighboring planes is $25mm$. The five-hole probe is traversed continuously along horizontal paths with a scan speed of

$10mm/s$. The distance between two adjacent paths is $5mm$. Every scan path is divided into $5mm$ segments over which the average of the recorded data is computed. The probe is operated at $100Hz$. Thus, at best, there are 50 samples per segment. However, in the separated flow region the number of "valid" data points may be reduced considerably. The net time to scan all 19 planes is about 12 hours.

The ProCap data is collected in two measurement runs, each with a length of 20 minutes. Compared to the measurement with the traversing system, this corresponds to a speed-up factor of 18. It is worth noting that this value does not account for the time it takes to program/teach the traverse. For the ProCap measurement the domain is split into voxels measuring $5mm$ in each direction ($\widehat{=}$ 473'760 voxels). The acquisition rate of the tracking and of the pressure signals is $100Hz$. To reconstruct the flow field the approximation method introduced in chapter 4 is applied. The parameters to be selected are set as follows:

- As the samples are considered to be uncorrelated, the integral time scale $M_{\mathcal{T}}$ is 0.5.
- The noise level L is assumed to be 0.5.
- The adaptive kernel width is determined by the certainty threshold s. For this measurement scenario, s is set rather conser-

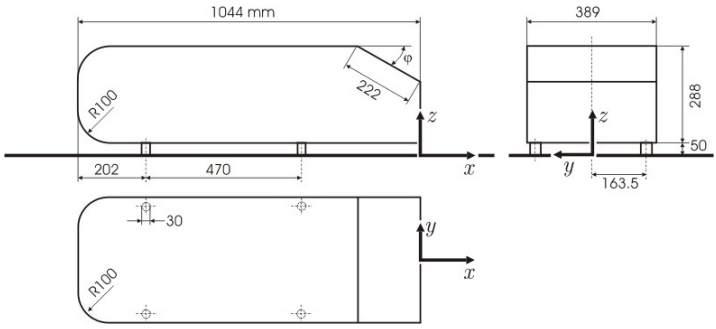


Figure 6.9: Geometry and dimensions of the Ahmed body (in mm). Figure from *Hinterberger et al.* (2004).

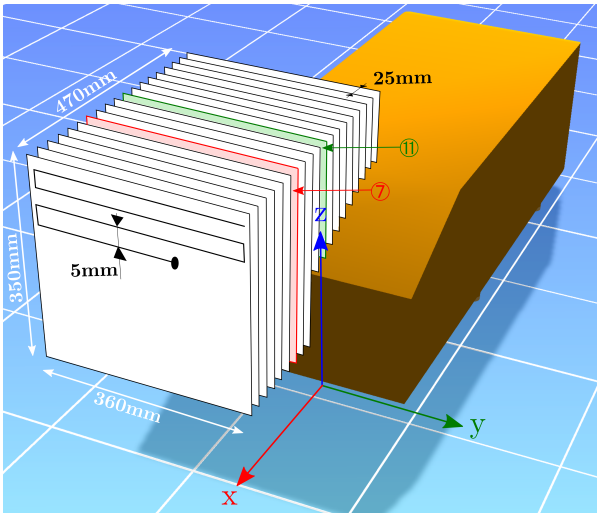


Figure 6.10: Measurement domain for Ahmed body

vative to a numerical value of 30.

Below, the results of the two measurement approaches are compared at two different positions, namely at $x = 371mm$ (Plane 7) and at $x = 271mm$ (Plane 11). Figure 6.12a shows the distribution of the total pressure loss (scaled by the bulk dynamic pressure) at $x = 371mm$. The arrows indicate the direction and magnitude of the projected velocity field. Qualitatively, the distributions of the total pressure loss look quite similar. In particular, the C-pillar vortex and the separation zone are clearly visible in both cases. As expected, away from the low-pressure zones, both approaches predict marginal losses of total pressure (inviscid flow). ProCap, however, seems to perform worse in regions where the total pressure field has a high curvature (cf. chapter 4). The tendency to smooth the underlying function becomes more marked if the distributions are plotted along a line (cf. figure 6.12a). ProCap's smoothing behavior can be partly explained by the comparatively conservative approximation settings (voxel size and certainty threshold s). To achieve better accuracy one could reduce the voxel size and/or the certainty threshold s.

But this comes at the cost of a longer measurement time.

Figures 6.12b, 6.13a and 6.13b depict the scaled distributions of the velocity components u, v and w in the same plane (at $x = 371mm$). The results support the findings described above, i.e. regarding the ProCap measurement regions with a large Hessian are less accurately resolved due to the smoothing behavior of the approximation scheme. In addition to that, the figures also show that for the measurement with the traverse the noise in the detached flow zone is significantly higher than in the surrounding area. This has two causes: On the one hand, the flow fluctuations in this region are stronger, therefore more samples are required to reduce the uncertainty of the average estimator. On the other hand, there is a good chance that the mean flow direction falls outside the application range of the five hole probe. Hence, the number of samples used to determine the average may be substantially less than usual. In the ProCap case, the noise is less conspicuous as the lower sampling density is compensated by a kernel with a larger support.

If the test model intersects the domain of interest, scanning with a traverse is challenging. One has to make sure that the probe/sting and the model do not come in contact with one another. Since the flow leads to slight probe and model vibrations, measurements close to the model surface are almost impossible. By comparison, in ProCap the movement of the probe can be precisely controlled by the operator allowing one to measure in the proximity of the model*. By examining figures 6.14 and 6.15 the benefit of the hand-guided approach is immediately visible in that the scan with the traverse covers the interesting flow region only partially. As before, these figures show the distributions of the total pressure loss and the three velocity components, but this time at $x = 271mm$. At first sight, there seems to be a systematic difference in the curves depicting the distribution of the quantity being studied along the line C-C, particularly near the centerline of model. However, this is an artifact and can be explained by the fact that a large part of the line lies in a zone where the function has a high curvature. To put it another way, the observations made at $x = 271mm$ coincide with

*It is worth pointing out that the probe calibration does not account for the wall effects

those made at $x = 371mm$.

As seen above, with the selected interpolation parameters the in-plane resolution of the ProCap measurement is worse than that obtained by the traversing system. However, it may also be noted that for the traverse scan the spatial resolution in the x-direction is a factor 5 lower due to the relatively large gap between two planes. ProCap, on the other hand, is not subject to this limitation. Another advantage of the ProCap system is that the recorded data is inherently interpolated onto a 3D grid allowing for a truly volumetric visualization of the flow topology (see figure 6.11).

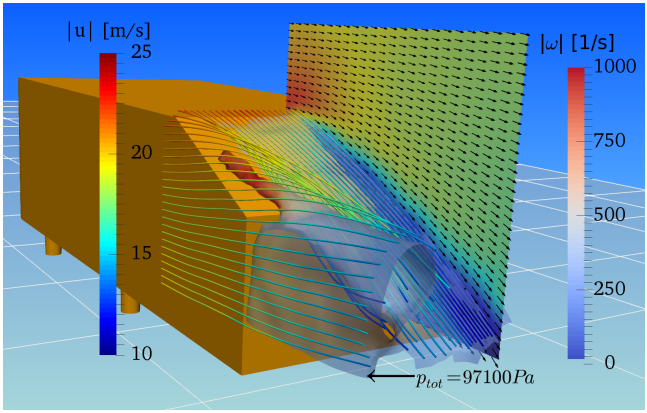
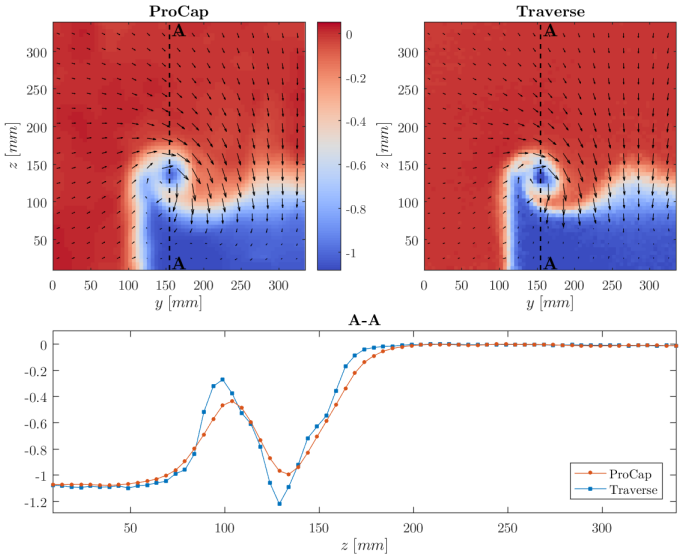
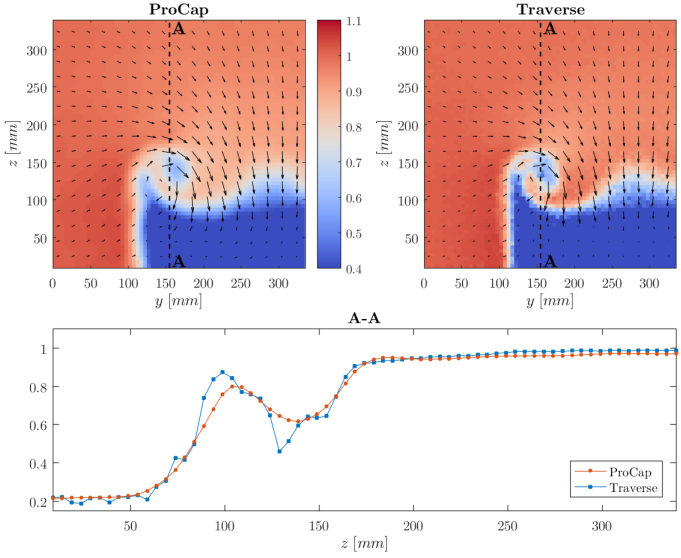


Figure 6.11: ProCap provides visualization capabilities so far only known from CFD.

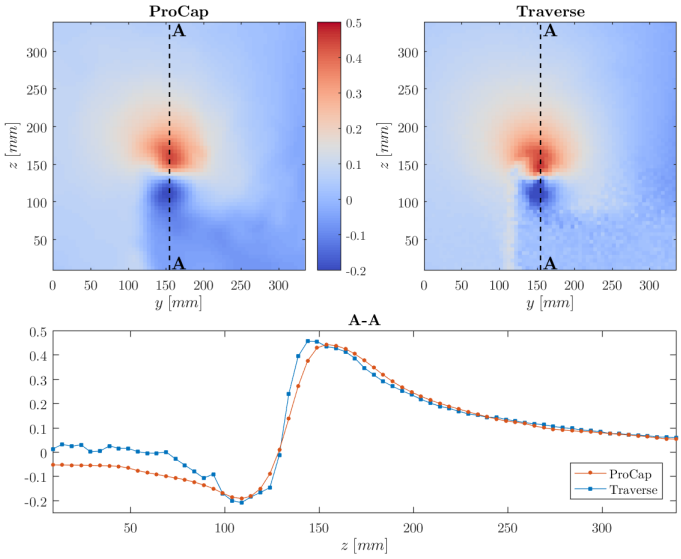


(a) $\Delta p_{tot}/(0.5\rho u_\infty^2)$

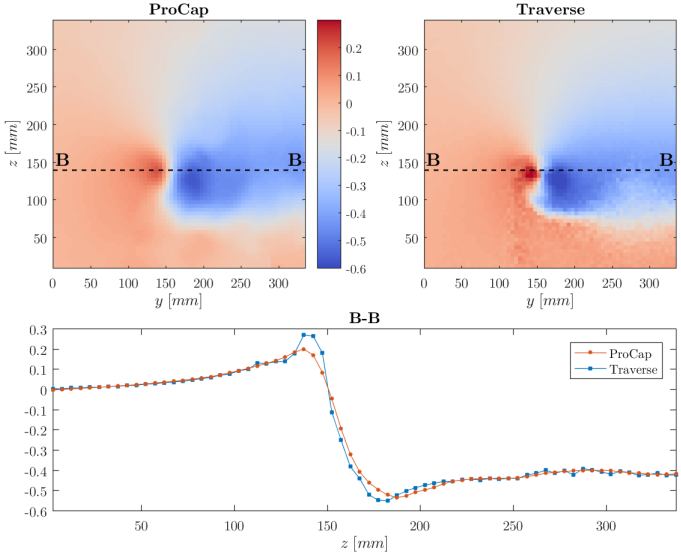


(b) u/u_∞

Figure 6.12: Comparison between ProCap measurement and traversing scan at $x = 371mm$ (Plane 7)

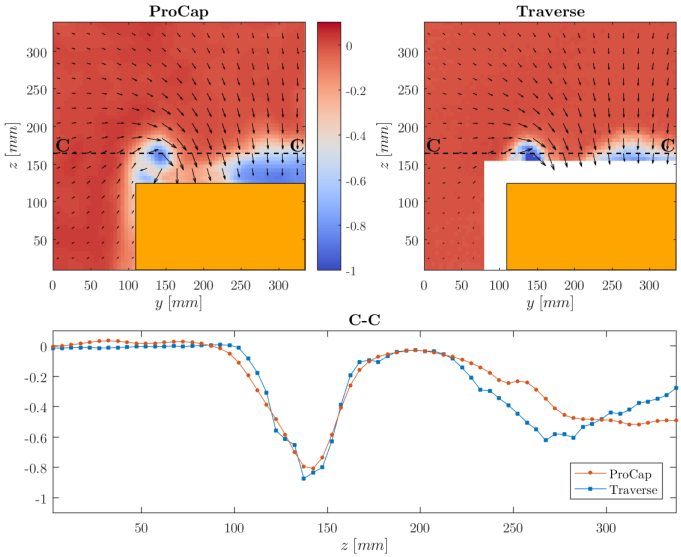


(a) v/u_∞

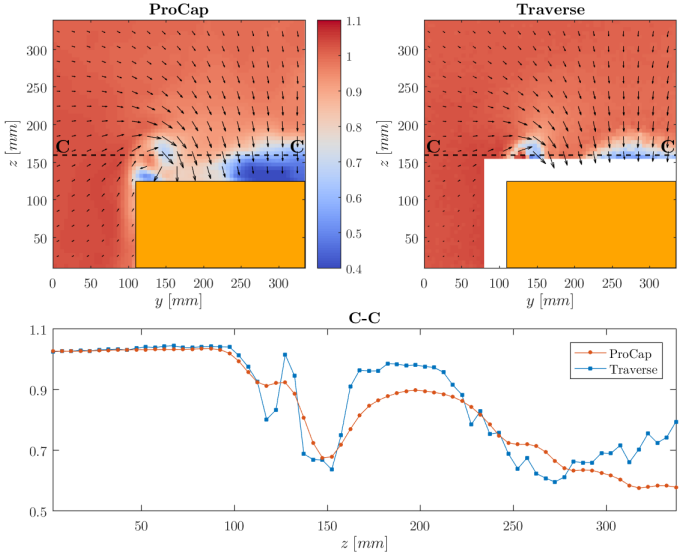


(b) w/u_∞

Figure 6.13: Comparison between ProCap measurement and traversing scan at $x = 371mm$ (Plane 7).

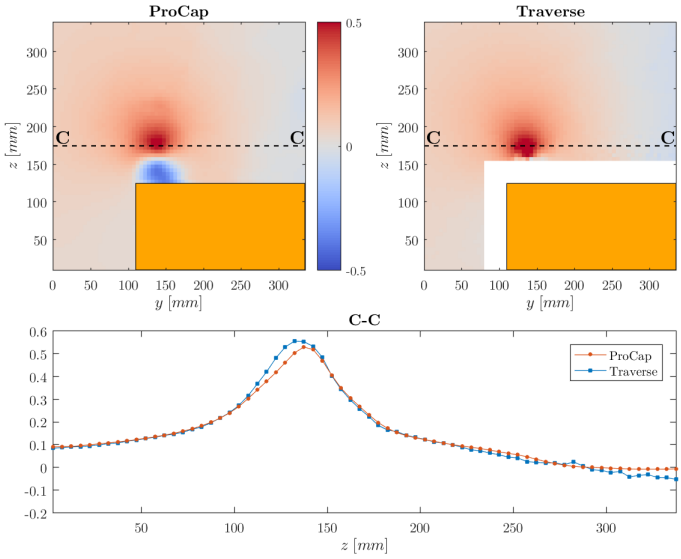


(a) $\Delta p_{tot}/(0.5\rho u_\infty^2)$

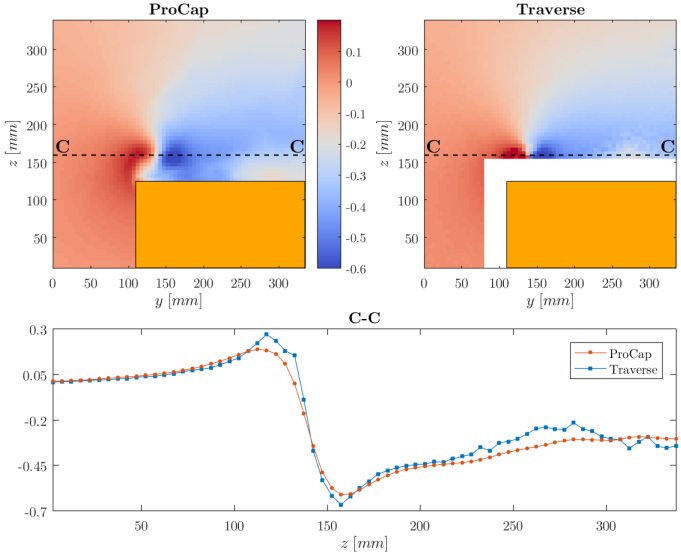


(b) u/u_∞

Figure 6.14: Comparison between ProCap measurement and traversing scan at $x = 271mm$ (Plane 11)



(a) v/u_∞



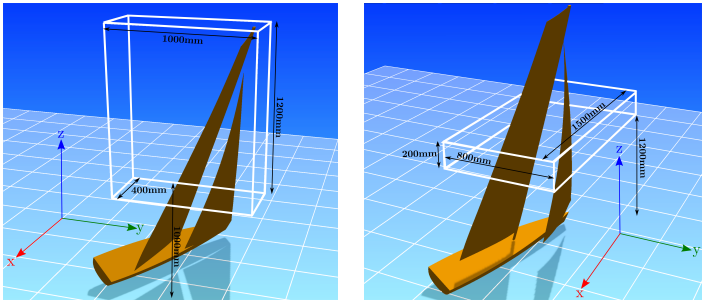
(b) w/u_∞

Figure 6.15: Comparison between ProCap measurement and traversing scan at $x = 271mm$ (Plane 11)

6.2.3 Sailing yacht

The preceding examples have demonstrated how ProCap performs compared to conventional traversing measurements. In doing so, no attention was paid to the fact that measurements using a traversing system are often not practical. This is particularly true if the shape of the test article depends on the flow conditions or if the domain of interest is not accessible due to the model blocking the traversing path. Considering this aspect, ProCap is much more flexible, as demonstrated in the next example. A custom race yacht in close-hauled configuration (1:12.5 scale) is installed in the twisted flow wind tunnel at the University of Auckland. As shown in figure 6.16, ProCap is used to scan two measurement volumes. One is intended to study the rather complex vortex system in the wake of the jib and the main sail (see figure 6.16a), while the other focuses on the flow around and between the two sails (see figure 6.16b). The wind speed measures $4.2m/s$, therefore, the five-hole-probe is fitted with the $25Pa$ pressure transducers (FirstSensor LBAS025BE).

The first volume is $400mm$ long, $1000mm$ wide and $1200mm$ high and is decomposed into cubic voxels of $10mm$ edge length (i.e. 480'000 voxels). The ProCap scan takes about 40 minutes to complete. As a result of the low wind speed ($u \sim 4m/s$) and the relative large model size ($l \sim 1m$), the integral time scale of the



(a) Domain in the wake **(b)** Domain around the sails

Figure 6.16: Sailing yacht in close-hauled setup

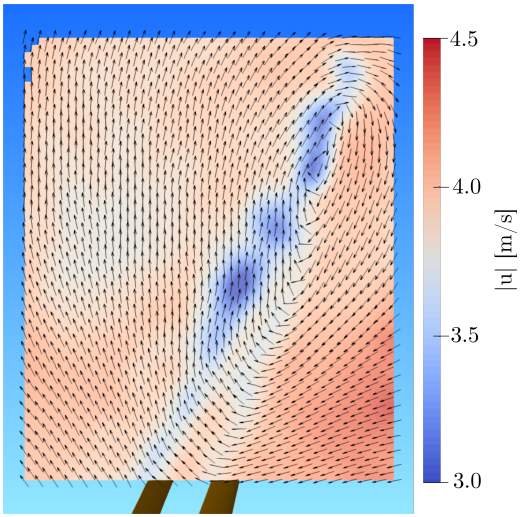


Figure 6.17: Wake measurement (plane $x = 250mm$): The colors indicate the distribution of the velocity magnitude, whereas the vectors show the in-plane velocity.

flow fluctuations is relative large and can be estimated by the large-eddy-turnover time ($l/u \sim 0.25s$). Since the recording frequency is $100Hz$, the integral time scale $M_{\mathcal{T}}$ used for the approximation can be estimated to be 25 samples. The noise level L and the certainty threshold s are set as in the previous cases, i.e. $L = 0.5$ and $s = 20$.

Figure 6.17 displays the distribution of the velocity magnitude in the y-z center plane. The arrows illustrate the projected velocity. On the leeward side the air is displaced downward. In contrast, an upwash can be detected on the windward side. Moreover, the wakes of both sails are visible; the wake of the jib sail is considerably weaker than that of the main sail. Compared to the NACA wing, the wake profile is less regular. This can be explained by the higher complexity of the "wing" shapes, the interaction between the two wakes and the disturbances introduced by the rigging (standing and running).

For the second domain, the same approximation settings are app-

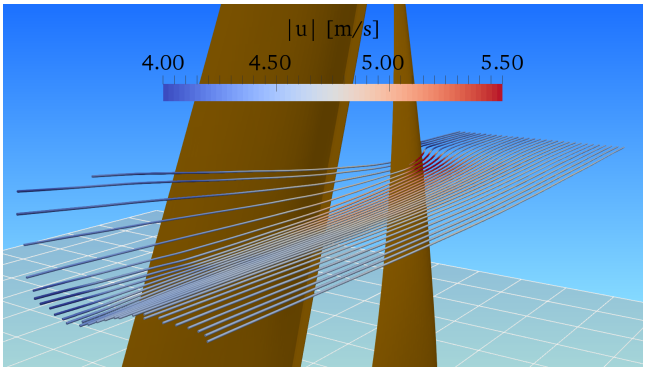


Figure 6.18: Streamlines around the jib and the main sail. The color maps the velocity magnitude

lied, except that the edge length of the voxels is reduced by a factor 2 ($\widehat{=}5mm$). The whole domain, consisting of 1'920'000 voxels, takes about 40 minutes to scan. The streamlines in figure 6.18 illustrate the twist of the flow caused by the jib sail. The local wind speed is reflected by the color of the streamlines. Clearly, the flow is accelerated on the leeward side of the jib and main sail (see also figure 6.19). Furthermore, from figure 6.19, one can make the following observations:

- At this height, the wake of the main sail is significantly stronger than that of the jib.
- The mast creates a separation bubble on the leeward side of the main sail. Interestingly, this flow structure can be partially captured by the ProCap system, despite the short distance to the sail (wall effects) and the limited angle range of the probe.

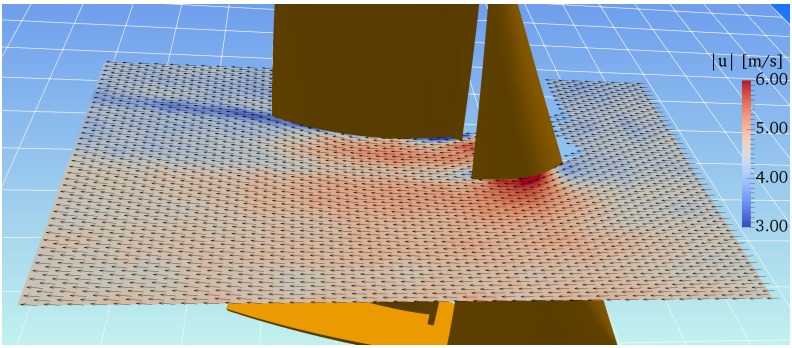
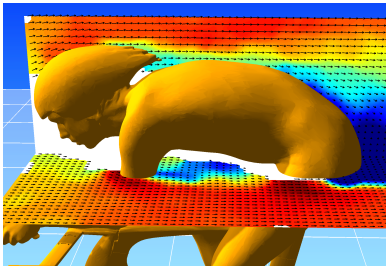


Figure 6.19: Velocity distribution around the jib and the main sail.

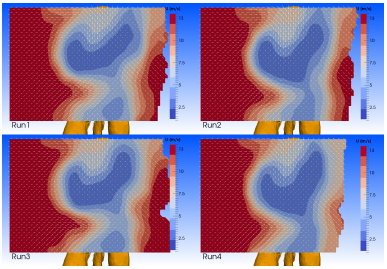
6.3 Application examples

The aim of this section is to demonstrate the versatility of ProCap and its many applications. We forego a comprehensive description of the examples since our only goal here is to provide a brief overview of the wide range of applications:

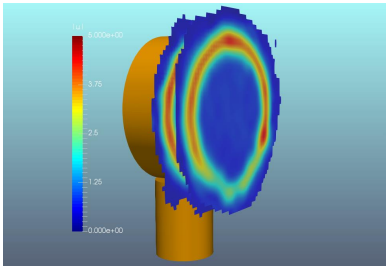
- Figure 6.20a illustrates the flow around a 1:1 scale model of a cyclist. The tests aimed at assessing the effect of different fabrics and helmets on the wake.
- Figure 6.20b shows the velocity distribution in the wake of a cyclist. The goal of these measurements was to demonstrate repeatability. In the first and the third run there was a small gap between the mannequin's back and the saddle. Whereas in the second and the forth run the gap was plugged with a piece of rubber foam. The position of the rider and the bike was not changed.
- In another test, ProCap was used to measure the flow created by a bladeless table fan (Dyson Cool™AM06). Figure 6.20c shows the velocity distribution in two planes.
- ProCap can be operated with different probe types. To study the ventilation system of a passenger car (see figure 6.20d), ProCap was used in conjunction with thermoelectric anemometer (Schiltknecht ThermoAir I).
- Due to the complex geometry, setting up and running a CFD calculation of a flow around a tree is challenging, particularly regarding the amount of modeling involved. On the other hand, the time it takes to set up and run ProCap is comparable to that of any other test case. Figure 6.20e shows the vorticity distribution resulting from a Christmas tree exposed to wind.
- Figure 6.20f displays the flow created by an array of computer fans.



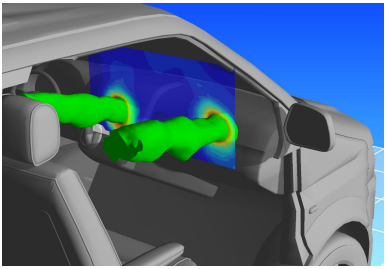
(a) Flow around a cyclist (velocity distribution)



(b) Wake of a cyclist (velocity distribution). Run1 → upper left; Run2 → upper right; Run3 → lower left; Run4 → lower right

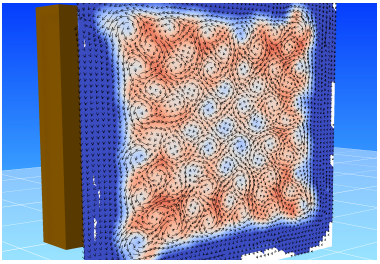


(c) Air flow created by the Dyson Cool™AM06 (velocity distribution)



(d) Ventilition of a VW T5 (velocity distribution)

(e) Air flow around a Christmas tree (vorticity distribution)



(f) Flow created by an array of computer fans (velocity distribution)

Figure 6.20: Gallery

Chapter 7

Conclusions and Outlook

The presented work differs from previous research in a number of respects. Most quantitative flow visualization techniques focus on measurement accuracy only. ProCap, on the other hand, extends its focus towards applicability and efficiency. This different viewpoint bears its own difficulties of which a number have been addressed in this study:

1. Combining data from different sources in an efficient manner
2. Real-time data processing
3. Efficient interpolation of highly irregularly distributed data
4. Interactive, real-time visualization of the measured data

The multi-threaded design of the software enables fast data acquisition and processing. Of fundamental importance to the system's usability is the real-time interpolation of the non-uniformly distributed measurement data onto a regular grid. To preserve the real-time functionality of the system, the interpolation is carried out in parallel on the GPU. A moving least-squares approximation scheme that adapts to the local point distribution proved to perform well both in terms of reconstruction quality and computing time. Tests with two ordinary GPUs have revealed that the interpolation is able to keep up with the target frame rate of $60 fps$ without difficulty. However, a challenge for future research will be to incorporate a more sophisticated bandwidth selection scheme without losing the real-time capability. To ensure smooth visualization of the interpolated data, ProCap makes use of the programmable shader functionality of Microsoft's DirectX framework. Currently, ProCap supports the following flow visualization tools: Contour slices, vector plots, streamlines and isosurfaces.

Provided that an appropriate flow sensor is available, the current system is capable of determining the time-averaged, volumetric distribution of any measurable flow quantity. In this work, the system was tested with a conventional five-hole probe. For a working distance of approximately $2m$ the maximum tracking error of the probe was estimated to be smaller than $1.9mm$ in position and $1.9°$ in orientation. A novel, flow-based calibration method was developed and implemented into ProCap with the aim of extending the measuring range of the probe. In addition, corrections were made to eliminate errors caused by the motion of the probe.

While the measurement quality is comparable to that of a scanning traverse system, the real benefit of ProCap lies in the simplicity of the setup, the comparatively short measurement time, the capability of covering relatively large volumes and the possibility of providing a visual feedback of high-level measurement data in real-time. The way of operation closely resembles that of hand-operated (smoke) probes. This parallel does not, however, extend to the nature of data being collected. While ProCap provides high-quality, quantitative flow data, smoke probes usually serve only as qualitative flow visualization tools. Compared to measurements with a traversing measurement, the overall speed up factor may be as high as 20. Thereby, the method demonstrates its potential to fill the gap between simple yet purely qualitative and quantitative but highly complex flow visualization techniques.

Although the current system is fully functional, there are areas that offer opportunities for further research and development, such as:

- Error quantification in ProCap is a difficult task as interpolation errors are mixed with measurement errors and temporal changes of the underlying flow. Due to the "one measurement per point and time" paradigm statistical quantification is almost impossible or requires assumptions about the flow (e.g. spatial and temporal correlation properties). Having said that, displaying the distribution of a measure that accounts for the reliability of the measurement at a certain location would be of real value to the system.

- In complex low-speed flow scenarios outside the controlled

wind tunnel environment (e.g. in-cabin room ventilation, clean room ventilation) the use of a five-hole probe is inadequate (sensitivity, Re-number dependence, responsiveness). The lack of commercial available solutions will require the development of a new low speed, omni-directional flow sensing probe.

- The integration of a 3D head-mounted augmented reality display is suggested to improve the ease of operation considerably.
- To shorten the measurement time and to improve the large-scale capability of the method, the single-probe design could be changed to a multi-probe design.
- Other potential areas of applications can be identified, e.g. replacing the hand-guided probe by a drone-mounted flow sensor would allow scanning considerably larger volumes (cf. figure 7.1). Similarly, the hand-guided probe may be replaced by a remotely and interactively controlled probe arm, enabling measurements in high speed wind tunnel flows without human access.

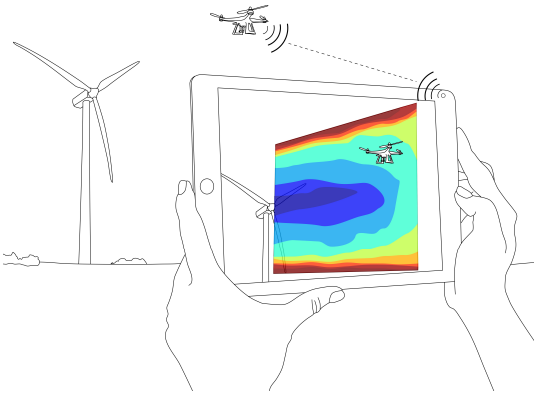


Figure 7.1: Possible possible application in the future: Large-scale wake surveys using drones

Appendices

Appendix A

Quaternions

Quaternions are a number system that can be considered as an extension of complex numbers. The concept of quaternions was introduced by the Irish mathematician William Rowan Hamilton in 1843. At the beginning of the 19th century, complex numbers were studied extensively by mathematicians such as Cauchy and Riemann. Inspired by the power and usefulness of complex numbers, Hamilton's intention was to find an number system that could be used not only for 2D problems but also for 3D problems, e.g. describing rotations. His first attempt were numbers that consisted of one real part and two imaginary parts, i.e. $z = a + b\,i + c\,j$. The problem with this natural approach is that closeness under multiplication cannot be achieved. Hamilton's solution to this problem was the addition of a third imaginary number k, i.e. $z = a + b\,i + c\,j + d\,k$. Then the algebraic structure is closed under multiplication by following the rules

$$i^2 = j^2 = k^2 = ijk = -1. \tag{A.1}$$

In mathematical terms, quaternions possess all properties of a field except that the multiplication does not fulfill the commutative property. Such an algebraic structure is called skew field or division ring. The skew field of quaternions is often denoted as $\mathbb{H}$ in honor of Hamilton.

Shortly after their discovery, quaternions became very popular in physics and mathematics. However, with the advent of vector analysis in the second half of the 19th century, the importance of quaternions in science decreased rapidly and they faced the risk of falling into oblivion. However, by the end of the 20th century, quaternions found their way back into science and technology and proved to be very useful in computer graphics, robotics and altitude control. There are mainly two reasons responsible for this revival:

(1) Compared to rotation matrices, quaternions are more com-

pact and the calculation of rotations is computationally less expensive.

(2) Quaternions are not/less susceptible to gimbal lock.

A.1 Notation and Properties of Quaternions

A quaternion $z_q \in \mathbb{H}$ can either be written as a sum of four terms (one real part and three imaginary parts)

$$z_q = a + b\,i + c\,j + d\,k, \quad a, b, c, d \in \mathbb{R} \tag{A.2}$$

or simply as quadruple

$$z_q = (a, \boldsymbol{a}), \quad a \in \mathbb{R},\ \boldsymbol{a} = [b, c, d]^T \in \mathbb{R}^3. \tag{A.3}$$

We use the subscript q to indicate that number at hand is a quaternion. Below we list a number of important properties and calculation rules of quaternions. Given are three quaternions a_q, b_q, $c_q \in \mathbb{H}$ that can be expressed either in sum or quadruple notation:

Sum notation	*Quadruple notation*
$a_q = a_0 + a_1 i + a_2 j + a_3 k$	$a_q = (a_0, \boldsymbol{a})$ with $\boldsymbol{a} = [a_1, a_2, a_3]^T$
$b_q = b_0 + b_1 i + b_2 j + b_3 k$	$b_q = (b_0, \boldsymbol{b})$ with $\boldsymbol{b} = [b_1, b_2, b_3]^T$
$c_q = c_0 + c_1 i + c_2 j + c_3 k$	$c_q = (c_0, \boldsymbol{c})$ with $\boldsymbol{c} = [c_1, c_2, c_3]^T$

(A.4)

1.) Addition of quaternions

Sum notation:

$$a_q + b_q = (a_0 + b_0) + (a_1 + b_1)\,i + (a_2 + b_2)\,j + (a_3 + b_3)\,k$$

Quadruple notation:

$$a_q + b_q = (a_0 + b_0, \boldsymbol{a} + \boldsymbol{b})$$

Summation of quaternions is *commutative*, i.e.

$$a_q + b_q = b_q + a_q$$

and *associative*, i.e.

$$(a_q + b_q) + c_q = a_q + (b_q + c_q)$$

2.) Subtraction of quaternions

Sum notation:

$$a_q - b_q = (a_0 - b_0) + (a_1 - b_1)\, i + (a_2 - b_2)\, j + (a_3 - b_3)\, k$$

Quadruple notation:

$$a_q - b_q = (a_0 - b_0, \boldsymbol{a} - \boldsymbol{b})$$

3.) Multiplication of quaternions

In the sum notation, the following identities are used

$$i^2 = j^2 = k^2 = ijk = -1,$$
$$ij = k, \quad ji = -k,$$
$$ki = j, \quad ik = -j,$$
$$jk = i, \quad kj = -i,$$

to deduce the quaternion product:

$$\begin{aligned} a_q b_q = {} & a_0 b_0 - a_1 b_1 - a_2 b_2 - a_3 b_3 \\ & + (a_0 b_1 + b_0 a_1 + a_2 b_3 - a_3 b_2)\, i \\ & + (a_0 b_2 + b_0 a_2 + a_3 b_1 - a_1 b_3)\, j \\ & + (a_0 b_3 + b_0 a_3 + a_1 b_2 - a_2 b_1)\, k \end{aligned}$$

Quadruple notation:

$$a_q b_q = (a_0 b_0 - \boldsymbol{a} \cdot \boldsymbol{b}, a_0 \boldsymbol{b} + b_0 \boldsymbol{a} + \boldsymbol{a} \times \boldsymbol{b})$$

Quaternion multiplication is *non-commutative*, i.e.

$$a_q b_q \neq b_q a_q$$

However, the associative law still holds:

$$(a_q b_q)\, c_q = a_q\, (b_q c_q)$$

4.) Distributive property

$$a_q\, (b_q + c_q) = a_q b_q + a_q c_q,$$
$$(b_q + c_q)\, a_q = b_q a_q + c_q a_q$$

5.) Length of quaternions

The length of a quaternion is given by:

$$|a_q| = \sqrt{a_0^2 + a_1^2 + a_2^2 + a_3^2}$$

Alternatively it can be expressed in quadruple notation:

$$|a_q| = \sqrt{a_0^2 + \boldsymbol{a} \cdot \boldsymbol{a}}$$

An important property of quaternions is that the length is *multiplicative*, i.e. the length of a product is given by the product of the lengths:

$$|a_q b_q| = |a_q||b_q|$$

6.) Conjugation of quaternions

Similar to complex numbers, one can define a conjugate of a quaternion:

$$\begin{aligned}\overline{a}_q &= a_0 - a_1 i - a_2 j - a_3 k \\ &= (a_0, -\boldsymbol{a})\end{aligned}$$

In the context of quaternion conjugation, one can deduce a number of rules:

$$\begin{aligned}\overline{a_q b_q} &= \overline{b}_q \overline{a}_q \\ a_q \overline{a}_q &= |a_q|^2\end{aligned}$$

7.) Inverse elements of quaternions

The inverse element of a quaternion a_q with respect of addition is simply $-a_q$, since the neutral element is zero.
The neutral element of the quaternion multiplication is one, thus it is easy to verify that the inverse element of a_q is defined as

$$a_q^{-1} = \frac{\overline{a}_q}{|a_q|^2}$$

8.) Division of quaternions

Since quaternion multiplication is non-commutative, there are two ways to define quaternion division. We call them either right division $a_q b_q^{-1}$ or left division $b_q^{-1} a_q$.

9.) Unit quaternions

A quaternion a_q is called *unit quaternion* if and only if the length $|a_q|$ is one.

10.) Pure quaternions

A quaternion a_q is called *pure quaternion* if and only if the scalar component a_0 is zero. Commonly points in the three-dimensional Euclidean space are expressed by pure quaternions, i.e.

$$\boldsymbol{x} \in \mathbb{R}^3 \rightarrow (0, \boldsymbol{x}) \in \mathbb{H}$$

11.) Rotation with quaternions

In the Euclidean space $\mathbb{R}^3$ the rotation of a point vector $\boldsymbol{x}$ is defined by a matrix multiplication

$\boldsymbol{y} = \boldsymbol{R}\boldsymbol{x}, \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^3$ and $\boldsymbol{R} \in \mathbb{R}^{3\times 3}$.

Here, $\boldsymbol{R}$ denotes the rotation matrix. In terms of quaternions a rotation of a point vector can be expressed by two quaternion multiplications

$y_q = q_q x_q \overline{q}_q, \quad x_q, y_q, q_q \in \mathbb{H}$

where x_q and y_q are the pure quaternions of the point $\boldsymbol{x}$ and the point $\boldsymbol{y}$ respectively. The quaternion q_q is a unit quaternion being defined by the rotation axis $\boldsymbol{c}$ and the rotation angle ϕ:

$q_q = (\cos(\phi/2), \sin(\phi/2)\,\boldsymbol{c})$

The rotation matrix $\boldsymbol{R}$ is linked to the rotation quaternion q_q by the following formula:

$$\boldsymbol{R} = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix}$$

By expressing rotations based on quaternions, it is relatively easy to concatenate rotations. This may be illustrated by applying two consecutive rotations q_q and p_q

$$z_q = p_q y_q \overline{p}_q = \underbrace{p_q q_q}_{=:r_q} x_q \underbrace{\overline{q}_q \overline{p}_q}_{=\overline{r}_q}$$

12.) Multiplication in matrix notation

Alternatively, the multiplication of two quaternions $a_q b_q$ can be formulated by a product of a 4×4-Matrix and a 4-component column vector. Depending on which quaternion is used to construct the matrix, there are two ways to introduce this reformulation:

$$a_q b_q = \begin{cases} \boldsymbol{A}_L(a_q)\,\boldsymbol{b}(b_q) \\ \boldsymbol{A}_R(b_q)\,\boldsymbol{a}(a_q) \end{cases},$$

where "L" and "R" stands for left and right. While $\boldsymbol{a}$ and $\boldsymbol{b}$ are the natural interpretation of the quaternions a_q and b_q as 4×1-vector, i.e.

$\boldsymbol{a} = [a_0, a_1, a_2, a_3]^T$ and $\boldsymbol{b} = [b_0, b_1, b_2, b_3]^T$

the construction of the matrices $\boldsymbol{A}_L$ and $\boldsymbol{A}_R$ is less obvious:

$$\boldsymbol{A}_L(a_q) = \begin{bmatrix} a_0 & -a_1 & -a_2 & -a_3 \\ a_1 & a_0 & -a_3 & a_2 \\ a_2 & a_3 & a_0 & -a_1 \\ a_3 & -a_2 & a_1 & a_0 \end{bmatrix} \tag{A.5}$$

$$\boldsymbol{A}_R(b_q) = \begin{bmatrix} b_0 & -b_1 & -b_2 & -b_3 \\ b_1 & b_0 & b_3 & -b_2 \\ b_2 & -b_3 & b_0 & b_1 \\ b_3 & b_2 & -b_1 & b_0 \end{bmatrix} \tag{A.6}$$

Appendix B

Eigenvalue Perturbation

Let $\boldsymbol{A}$ be a symmetric, positive definite $N \times N$-Matrix. The eigenvalues $\{\lambda_i\}_{i=1}^N$ are known to be real. It is further assumed that all eigenvalues are mutually distinct. Thus, the normalized eigenvectors $\{\boldsymbol{u}_i\}_{i=1}^N$ form an orthonormal basis. The eigenvalues and eigenvectors are linked by the following set of equations:

$$\boldsymbol{A}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i, \quad i = 1, \ldots, N. \tag{B.1}$$

Suppose the matrix $\boldsymbol{A}$ is contaminated with noise

$$\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{\delta A}. \tag{B.2}$$

The level of noise $\boldsymbol{\delta A}$ is small compared to $\boldsymbol{A}$, i.e. $\boldsymbol{\delta A} << \boldsymbol{A}$. Furthermore, it is assumed that the noise does not affect the properties of $\boldsymbol{A}$, i.e. $\widetilde{\boldsymbol{A}}$ remains symmetric and positive definite. The eigenvalues $\{\widetilde{\lambda}_i\}_{i=1}^N$ and normalized eigenvectors $\{\widetilde{\boldsymbol{u}}_i\}_{i=1}^N$ of the noisy matrix $\widetilde{\boldsymbol{A}}$ are the solutions of

$$\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{u}}_i = \widetilde{\lambda}_i \widetilde{\boldsymbol{u}}_i, \quad i = 1, \ldots, N. \tag{B.3}$$

The goal of this section is to deduce estimates for the perturbations of the eigenvalues

$$\delta\lambda_i = \widetilde{\lambda}_i - \lambda_i, \quad i = 1, \ldots, N \tag{B.4}$$

and eigenvectors

$$\boldsymbol{\delta u}_i = \widetilde{\boldsymbol{u}}_i - \boldsymbol{u}_i, \quad i = 1, \ldots, N \tag{B.5}$$

caused by the noise $\boldsymbol{\delta A}$. Substituting expression (B.4) and (B.5) into equation (B.3) yields

$$(\boldsymbol{A}+\boldsymbol{\delta A})\,(\boldsymbol{u}_i+\boldsymbol{\delta u}_i)=(\lambda_i+\delta\lambda_i)\,(\boldsymbol{u}_i+\boldsymbol{\delta u}_i)$$

$$\begin{aligned}\Leftrightarrow\ & \boldsymbol{A u}_i+\boldsymbol{A\delta u}_i+\boldsymbol{\delta A u}_i+\boldsymbol{\delta A\delta u}_i\\ & =\lambda_i\boldsymbol{u}_i+\lambda_i\boldsymbol{\delta u}_i+\delta\lambda_i\boldsymbol{u}_i+\delta\lambda_i\boldsymbol{\delta u}_i,\quad i=1,\ldots,N\end{aligned}\tag{B.6}$$

As the perturbations are considered to be small, the last term on both the left and right-hand side of the equation above are neglected. Subtracting equation (B.1) from this first order approximation leads to

$$\boldsymbol{A\delta u}_i+\boldsymbol{\delta A u}_i=\lambda_i\boldsymbol{\delta u}_i+\delta\lambda_i\boldsymbol{u}_i,\quad i=1,\ldots,N\tag{B.7}$$

As previously mentioned, the noise-free eigenvectors $\{\boldsymbol{u}_i\}_{i=1}^{N}$ build an orthonormal basis. Thus, the perturbation vectors can be expressed in terms of this basis

$$\boldsymbol{\delta u}_i=\sum_{j=1}^{N}\epsilon_{ij}\boldsymbol{u}_j,\quad i=1,\ldots,N.\tag{B.8}$$

Inserting these linear combinations into equation (B.7) yields

$$\boldsymbol{A}\sum_{j=1}^{N}\epsilon_{ij}\boldsymbol{u}_j+\boldsymbol{\delta A u}_i=\lambda_i\sum_{j=1}^{N}\epsilon_{ij}\boldsymbol{u}_j+\delta\lambda_i\boldsymbol{u}_i$$

$$\Leftrightarrow\sum_{j=1}^{N}\epsilon_{ij}\underbrace{\boldsymbol{A u}_j}_{=\lambda_j\boldsymbol{u}_j}+\boldsymbol{\delta A u}_i=\lambda_i\sum_{j=1}^{N}\epsilon_{ij}\boldsymbol{u}_j+\delta\lambda_i\boldsymbol{u}_i\tag{B.9}$$

To determine the first order approximation of the eigenvalue perturbations, equation (B.9) is multiplied from the left by $\boldsymbol{u}_i^T$:

$$\sum_{j=1}^{N} \epsilon_{ij}\lambda_j \underbrace{\boldsymbol{u}_i^T \boldsymbol{u}_j}_{=\delta_{ij}} + \boldsymbol{u}_i^T \boldsymbol{\delta A} \boldsymbol{u}_i = \lambda_i \sum_{j=1}^{N} \epsilon_{ij} \underbrace{\boldsymbol{u}_i^T \boldsymbol{u}_j}_{=\delta_{ij}} + \delta\lambda_i \underbrace{\boldsymbol{u}_i^T \boldsymbol{u}_i}_{=1}$$

$$\Leftrightarrow \ \epsilon_{ii}\lambda_i + \boldsymbol{u}_i^T \boldsymbol{\delta A} \boldsymbol{u}_i = \epsilon_{ii}\lambda_i + \delta\lambda_i$$

$$\Leftrightarrow \ \boxed{\delta\lambda_i = \boldsymbol{u}_i^T \boldsymbol{\delta A} \boldsymbol{u}_i}, \quad i = 1, \ldots, N \tag{B.10}$$

In a similar way, the eigenvector perturbations can be obtained. Equation (B.9) is multiplied from the left by $\boldsymbol{u}_k^T$ $(k \neq i)$:

$$\sum_{j=1}^{N} \epsilon_{ij}\lambda_j \underbrace{\boldsymbol{u}_k^T \boldsymbol{u}_j}_{=\delta_{kj}} + \boldsymbol{u}_k^T \boldsymbol{\delta A} \boldsymbol{u}_i = \lambda_i \sum_{j=1}^{N} \epsilon_{ij} \underbrace{\boldsymbol{u}_k^T \boldsymbol{u}_j}_{=\delta_{kj}} + \delta\lambda_i \underbrace{\boldsymbol{u}_k^T \boldsymbol{u}_i}_{=0}$$

$$\Leftrightarrow \ \epsilon_{ik}\lambda_k + \boldsymbol{u}_k^T \boldsymbol{\delta A} \boldsymbol{u}_i = \epsilon_{ik}\lambda_i$$

$$\Leftrightarrow \ \epsilon_{ik} = \frac{\boldsymbol{u}_k^T \boldsymbol{\delta A} \boldsymbol{u_i}}{\lambda_i - \lambda_k}, \quad i \neq k \text{ and } \ i, k = 1, \ldots, N \tag{B.11}$$

The missing coefficient ϵ_{ii} is determined by the condition that the eigenvector has unit length:

$$\widetilde{\boldsymbol{u}}_i^T \widetilde{\boldsymbol{u}}_i = 1 \Leftrightarrow \left(\boldsymbol{u}_i + \boldsymbol{\delta u_i}\right)^T \left(\boldsymbol{u}_i + \boldsymbol{\delta u_i}\right) = 1$$

$$\Leftrightarrow \ \underbrace{\boldsymbol{u}_i^T \boldsymbol{u}_i}_{=1} + \underbrace{2\boldsymbol{u}_i^T \boldsymbol{\delta u}_i}_{=2\epsilon_{ii}} + \underbrace{\boldsymbol{\delta u}_i^T \boldsymbol{\delta u}_i}_{\approx 0} = 1 \tag{B.12}$$

Thus, assuming small changes, the solution for the missing coefficient is:

$$\epsilon_{ii} = 0 \tag{B.13}$$

To sum up, the first order approximation of the eigenvector perturbations is given by

$$\boxed{\delta\boldsymbol{u}_i = \sum_{\substack{j=1 \\ j\neq i}}^{N} \frac{\boldsymbol{u}_j^T \boldsymbol{\delta A} \boldsymbol{u}_i}{\lambda_i - \lambda_j} \boldsymbol{u}_j} \tag{B.14}$$

Appendix C

Meshless scattered data approximation techniques

This chapter provides a summary of selected meshless scattered data interpolation methods, namely radial basis functions, Kriging and partition of unity. The methods are only outlined and not described in full detail. If more information is requested, the reader is referred to more comprehensive sources quoted in the text below.

C.1 Radial basis function Interpolation

C.1.1 Linear interpolation problem

There are an infinite number of functions $\widehat{u}$ solving the scattered data interpolation problem:

$$\widehat{u}(\boldsymbol{x}_i) = u_i, \quad i = 1, ..., M \tag{C.1}$$

To obtain a unique solution additional constraints have to be imposed. Basically, two interpolation methods differ from each other in the choice of these constraints. An often-used constraint that have a lot of methods in common (e.g. RBF, Kriging etc.) is the assumption that $\widehat{u}$ belongs to a linear M-dimensional function space $\mathcal{U}$, which is typically a subspace of the space the unknown function u belongs to. Suppose that $\mathcal{U}$ has a basis $\{a_i\}_{i=1}^{M}$, i.e. $\mathcal{U} = \text{span}\{a_i\}_{i=1}^{M}$, then $\widehat{u}$ may be expressed as

$$\widehat{u}(\boldsymbol{x}) = \sum_{i=1}^{M} c_i a_i(\boldsymbol{x}) \tag{C.2}$$

This ansatz is often referred to as the *linear interpolation approach*. Taking the interpolation constraints C.1 into account, the interpolation problem yields a system of M linear equations:

$$\boldsymbol{A}\boldsymbol{c} = \boldsymbol{u} \tag{C.3}$$

with

$$\begin{aligned} \boldsymbol{A}_{i,j} &= a_j\left(\boldsymbol{x}_i\right),\ i,j = 1,...,M \\ \boldsymbol{c} &= \left[c_1,...,c_M\right]^T \\ \boldsymbol{u} &= \left[u_1,...,u_M\right]^T \end{aligned} \tag{C.4}$$

The matrix $\boldsymbol{A}$ is the so called *interpolation matrix*.

C.1.2 Well-posedness of the linear interpolation problem

The linear interpolation problem above is well-posed if and only if $\boldsymbol{A}$ is invertible, i.e. $\boldsymbol{A}$ has full rank. As $\boldsymbol{A}$ only depends on the basis functions $\{a_i\}_{i=1}^M$ and on the data sites $\{\boldsymbol{x}_i\}_{i=1}^M$, the well-posedness of the linear interpolation problem is independent of the measured data values $\{u_i\}_{i=1}^M$. From a mathematical point of view, an interesting question is whether there exists a function space $\mathcal{U}$ such that $\boldsymbol{A}$ is non-singular for any set of M distinct data sites. Such a function space is known as *Haar space*. The *Maierhuber-Curtis Theorem* provides an answer to this question:

Theorem 2 (Maierhuber-Curtis Theorem)
On $\mathbb{R}^d, d \geq 2$ there exist no Haar spaces of continuous functions expect the one-dimensional ones.

In other words there exist Haar spaces of continuous functions if either $d = 1$ or $M = 1$. The proof of this theorem can be found in *Fasshauer* (2007).

From the Maierhuber-Curtis theorem one can deduce that in order to guarantee a solvable linear interpolation problem for $d \geq 1$ and $M \geq 1$, the interpolation space $\mathcal{U}$ has to depend on the data sites.

C.1.3 The relevance of strictly positive definite functions

Now, we look at a special subclass of data-sites dependent basis functions that guarantee well-posedness of the linear interpolation problem. Real positive definite matrices are defined as follows

Definition 4 (Positive Definite Matrix)
A real symmetric $M \times M$ matrix $\boldsymbol{A}$ is called positive definite *if*

$$\boldsymbol{c}^T \boldsymbol{A}\boldsymbol{c} \geq 0, \tag{C.5}$$

for every non-zero vector $\boldsymbol{c} \in \mathbb{R}^N$.

Positive definite matrices are known to be invertible. Similarly, one can define *strictly positive definite functions*:

Definition 5 (Strictly Positive Definite Function)
A real-valued continuous function $\Phi : \mathbb{R}^d \to \mathbb{R}$ is strictly positive definite *if and only if the function is even and if*

$$\sum_{i=1}^{M} \sum_{j=1}^{M} c_i c_j \Phi\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right) \geq 0, \tag{C.6}$$

for every set of M distinct nodes $\boldsymbol{x}_1, ..., \boldsymbol{x}_M \in \mathbb{R}^d$ and every non-zero vector $\boldsymbol{c} \in \mathbb{R}^M$.

Thus, if we use $\mathcal{U} = \operatorname{span}\left\{\Phi\left(\boldsymbol{x} - \boldsymbol{x}_i\right)\right\}_{i=1}^{M}$ with Φ being a strictly positive definite function, the resulting linear interpolation matrix $\boldsymbol{A}$ is positive definite and therefore non-singular. More specifically, the linear interpolation ansatz transforms into

$$\widehat{u}\left(\boldsymbol{x}\right) = \sum_{i=1}^{M} c_i \Phi\left(\boldsymbol{x} - \boldsymbol{x}_i\right) \tag{C.7}$$

resulting in a system of M linear equations:

$$\boldsymbol{A}\boldsymbol{c} = \boldsymbol{u} \tag{C.8}$$

with

$$\begin{aligned} \boldsymbol{A}_{i,j} &= \Phi\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right),\ i,j = 1,...,M \\ \boldsymbol{c} &= \left[c_1,...,c_M\right]^T \\ \boldsymbol{u} &= \left[u_1,...,u_M\right]^T. \end{aligned} \tag{C.9}$$

C.1.4 Linear Interpolation with polynomial precision

Let us consider the case where the function to be interpolated is a polynomial of order m. If the interpolation matrix is constructed on the basis of a strictly positive definite function, the linear interpolation does not reproduce the underlying polynomial exactly. For some applications, however, it is required that the interpolation method is capable of exactly reproducing such polynomials. As shown below, this requirement can be met by an extension of the linear interpolation approach. Nonetheless, it is worth mentioning that this technique does not improve the quality of the interpolant in terms of approximation power, it only allows to reproduce global low order polynomials exactly.

In order to reproduce a polynomial of degree m it is required that the data sites form a *m-unisolvent point set*.

Definition 6 (m-unisolvent point set)
Given a polynomial basis $\mathcal{P} = span\left\{p_i\right\}_{i=1}^{N}$ *of order* m *on* $\mathbb{R}^d$, *i.e.* $N = \binom{m+d}{m}$, *a set of* $M \leq N$ *data sites* $\mathcal{X} = \left\{\boldsymbol{x}_i\right\}_{i=1}^{M}$ *on* $\mathbb{R}^d$ *is said to be* m*-unisolvent if*

$$rank\left(\boldsymbol{A}\right) = N \tag{C.10}$$

with

$$\boldsymbol{A}_{i,j} = p_j\left(x_i\right),\ i = 1,...,M,\ j = 1,...,N \tag{C.11}$$

In other words, if the data values are zero, i.e. $u_i = 0,\ i = 1,...,M$, the only interpolating polynomial of degree m is the zero polynomial. Assuming that $\mathcal{X}$ is m-unisolvent (if m and d is small and N is rather large, this is nearly always the case), the linear interpolation

approach above can be modified accordingly:

$$\widehat{u}(\boldsymbol{x}) = \sum_{i=1}^{M} c_i a_i(\boldsymbol{x}) + \sum_{i=1}^{N} d_i p_i(\boldsymbol{x}), \tag{C.12}$$

where $\{a_i\}_{i=1}^{M}$ is a basis of $\mathcal{U}$ and $\{p_i\}_{i=1}^{N}$ of the polynomial vector space $\mathcal{P}_m$. The interpolation requirement $\widehat{u}(\boldsymbol{x}_i) = u_i,\ i = 1, ..., M$ yields the following system of linear equations:

$$\boldsymbol{A}\boldsymbol{c} + \boldsymbol{P}\boldsymbol{d} = \boldsymbol{u} \tag{C.13}$$

with

$$\begin{aligned}
&\boldsymbol{A}_{i,j} = a_j(\boldsymbol{x}_i),\ i, j = 1, ..., M \\
&\boldsymbol{P}_{i,j} = p_j(\boldsymbol{x}_i),\ i = 1, ..., M,\ j = 1, ..., N \\
&\boldsymbol{c} = [c_1, ..., c_M]^T \\
&\boldsymbol{d} = [d_1, ..., c_N]^T \\
&\boldsymbol{u} = [u_1, ..., u_M]^T.
\end{aligned} \tag{C.14}$$

Clearly, the number of equations (M) is less than the number of unknowns ($M + N$), and thus at least N additional equations are required to solve the problem. Besides uniqueness of the solution, we require that interpolation method reproduces the polynomials of order $\leq m$ exactly. Or to put it in other words, if $u(\boldsymbol{x}) = \sum_{j=1}^{N} e_j p_j(\boldsymbol{x})$ we wish to get the solution $\boldsymbol{c} = \boldsymbol{0}$ and $\boldsymbol{d} = \boldsymbol{e} := \{e_1, ..., e_N\}^T$. As shown below, this condition provides the M additional equations required to assure a unique solution. First, we rewrite the system of linear equations for the case where u is an arbitrary polynomial of order m:

$$\boldsymbol{A}\boldsymbol{c} + \boldsymbol{P}\boldsymbol{d} = \boldsymbol{P}\boldsymbol{e}. \tag{C.15}$$

Assuming that $\boldsymbol{c} = \boldsymbol{0}$ this equations guarantee that $\boldsymbol{d} = \boldsymbol{e}$, because due to the unisolvence condition the rank of $\boldsymbol{P}$ is N. Thus, the additional equations and the choice of basis functions $\{\boldsymbol{a}\}_{i=1}^{M}$ have to make sure that $\boldsymbol{c} = \boldsymbol{0}$ for u being an arbitrary polynomial of

order m. Usually, one picks $\boldsymbol{P}^T\boldsymbol{c} = \boldsymbol{0}$ as additional equations. By multiplying eq.(C.15) from the left with $\boldsymbol{c}^T$ yields

$$\boldsymbol{c}^T\boldsymbol{A}\boldsymbol{c} + \boldsymbol{c}^T\boldsymbol{P}\boldsymbol{d} = \boldsymbol{c}^T\boldsymbol{P}\boldsymbol{e} \xrightarrow{\boldsymbol{P}^T\boldsymbol{c}=\boldsymbol{0}} \boldsymbol{c}^T\boldsymbol{A}\boldsymbol{c} = 0. \tag{C.16}$$

As required, this equation is fulfilled in the trivial case $\boldsymbol{c} = \boldsymbol{0}$. However, in order to ensure that the trivial solution is the only solution the basis functions have to be selected accordingly. Provided that $\boldsymbol{P}^T\boldsymbol{c} = \boldsymbol{0}$, basis functions for which the quadratic form $\boldsymbol{c}^T\boldsymbol{A}\boldsymbol{c}$ is only zero in the trivial case $\boldsymbol{c} = \boldsymbol{0}$ and larger zero otherwise are called *strictly conditionally positive definite*.

Definition 7 (Strictly conditionally positive definite function of order $m+1$)
A real-valued continuous function $\Phi : \mathbb{R}^d \to \mathbb{R}$ *is known to be* strictly conditionally positive definite of order $m+1$ *if*

$$\sum_{i=1}^{M}\sum_{j=1}^{M} c_i c_j \Phi\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right) \geq 0, \tag{C.17}$$

for every set of M *distinct nodes* $\boldsymbol{x}_1, ..., \boldsymbol{x}_M \in \mathbb{R}^d$ *and all non-zero vectors* $\boldsymbol{c} \in \mathbb{R}^M$ *satisfying*

$$\boldsymbol{P}^T\boldsymbol{c} = \boldsymbol{0}. \tag{C.18}$$

$\boldsymbol{P}$ *is constructed by a basis* $\{p_i\}_{i=1}^N$ *of the polynomial vector space* $\mathcal{P}_m$, *i.e.*

$$\boldsymbol{P}_{i,j} = p_j\left(\boldsymbol{x}_i\right),\ i = 1, ..., M,\ j = 1, ..., N \tag{C.19}$$

Consequently, if $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^M$ is a m-unisolvent point set and $\boldsymbol{A}$ is determined by

$$\boldsymbol{A}_{i,j} = \Phi\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right),\ i, j = 1, ..., M, \tag{C.20}$$

with Φ being a strictly conditionally positive definite function of order $m+1$, the linear interpolation problem defined above is well-

posed. The resulting linear system is then given by

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{0} \end{bmatrix} \tag{C.21}$$

C.1.5 Radial basis function

Often, the linear approximation approach is used in conjunction with radial basis functions. In this case, the method is referred to as radial basis function (RBF) interpolation. The basis functions $\{\Phi(\boldsymbol{x} - \boldsymbol{x}_i)\}_{i=1}^{M}$ depend only on distance between the evaluation point $\boldsymbol{x}$ and the center of the function $\boldsymbol{x_i}$ and are defined by a scalar function $\varphi(x)$, i.e.

$$\Phi(\boldsymbol{x} - \boldsymbol{x}_i) = \varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|_2), \quad i = 1, \ldots, M \tag{C.22}$$

To guarantee solvability, the function φ has to be chosen such that the point-dependent basis function Φ is strictly positive definite.

C.1.6 Least squares approximation

The linear interpolation approach C.7 can also be used for least squares approximation. Assuming that the samples $\{u\}_{i=1}^{M}$ are the function values at the data sites $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{M}$. Now, we introduce a second point set $\mathcal{Y} = \{\boldsymbol{y}_j\}_{j=1}^{N}$ with $N \leq M$. These mutually distinct points shall define the N basis functions $\Phi(\boldsymbol{x} - \boldsymbol{y}_j)$, $j = 1, \ldots, N$ used in the linear approximation ansatz

$$\widehat{u}(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \Phi(\boldsymbol{x} - \boldsymbol{y}_j). \tag{C.23}$$

As in the interpolation case, the basis functions are often radial functions centered at the collocation points $\mathcal{Y} = \{\boldsymbol{y}_j\}_{j=1}^{N}$. The coefficients $[c_j]_{j=1}^{N}$ can be found by solving the following least squares

problem:

$$\sum_{i=1}^{M} \left[u_i - \sum_{j=1}^{N} c_j \Phi \left(\boldsymbol{x} - \boldsymbol{y}_j \right) \right]^2 \rightarrow \min \tag{C.24}$$

This approximation problem is well-defined if the matrix

$$\boldsymbol{A}_{j,i} = \Phi \left(\boldsymbol{x}_i - \boldsymbol{y}_j \right), \quad i = 1, \ldots, M, \; j = 1, \ldots, N \tag{C.25}$$

has full rank, i.e. $\operatorname{rank}(\boldsymbol{A}) = N$.

C.2 Kriging/Gaussian Process Regression

In principle, Kriging is identical to the RBF interpolation, as it will be outlined later. However, it differs from RBF by its stochastic-based formalism and interpretation. Furthermore, it has the advantage of providing an estimate of the interpolation precision. The basic concept of Kriging is that the unknown function $u(\boldsymbol{x})$ is assumed to be one realization of a random function $U(\boldsymbol{x})$. Roughly speaking, at every point $\boldsymbol{x}$ in Ω we construct a random variable $U(\boldsymbol{x})$ that depends on the random variables at other locations, and the sampled data $\{u_i\}_{i=1}^{M}$ are the point evaluations of one realization of the random function $U(\boldsymbol{x})$ at the data sites $\{\boldsymbol{x}_i\}_{i=1}^{M}$. The goal is to find an estimate $\widehat{U}(\boldsymbol{x})$ for the random function $U(\boldsymbol{x})$ in terms of the random variables $\{U(\boldsymbol{x}_i)\}_{i=1}^{M}$. In a second step, these random variables are replace by their realization counterparts $\{u_i\}_{i=1}^{M}$ to obtain an estimate $\widehat{u}(\boldsymbol{x})$ for the function $u(\boldsymbol{x})$.

In Geostatistics, Kriging, sometimes also called Gaussian Process Regression or Wiener-Kolmogorov prediction, is a collective term for a variety of multivariate interpolation techniques. Following three common representative of this group are briefly outlined, namely simple Kriging, ordinary Kriging and universal Kriging.

C.2.1 Simple Kriging

Simple Kriging is based on the assumption that the underlying random function $U(\boldsymbol{x})$ is second-order stationary. By definition, this is fulfilled if

(i) the expectation is not a function of space, i.e. $\mathrm{E}[U(\boldsymbol{x})] = m,\ \forall \boldsymbol{x} \in \mathbb{R}^d$

(ii) the covariance of two points is only a function of the separation vector, i.e. $\mathrm{Cov}\left[U(\boldsymbol{x}+\boldsymbol{h}), U(\boldsymbol{x})\right] = c(\boldsymbol{h}),\ \forall \boldsymbol{x}, \boldsymbol{h} \in \mathbb{R}^d$

An integral part of simple Kriging is the assumption that the expectation value m is known a priori. An good estimate is the average of all sampled values. Another practical problem involved in simple Kriging is the modeling of the covariance function $c(\boldsymbol{h})$. Possible strategies are properly covered in *Wackernagel* (2013). Frequently, not the covariance function but the semi-variogram $\gamma(\boldsymbol{h})$ is modeled, which is defined as follows

$$\gamma(\boldsymbol{h}) = \frac{1}{2}\mathrm{Var}\left[U(\boldsymbol{x}+\boldsymbol{h}) - U(\boldsymbol{x})\right] \tag{C.26}$$

As second-order stationarity is applied, a simple relation between the semi-variogram and the covariance function can be found

$$\gamma(\boldsymbol{h}) = c(\boldsymbol{0}) - c(\boldsymbol{h}) \tag{C.27}$$

For the estimate of the unknown random function a linear ansatz is proposed

$$\widehat{U}(\boldsymbol{x}) = m + \sum_{i=1}^{M} w_i(\boldsymbol{x})\left(U(\boldsymbol{x}_i) - m\right) \tag{C.28}$$

Under the assumption of second-order stationarity, it is straight forward to show that this linear estimator is unbiased, i.e. $\mathrm{E}[\widehat{U}(\boldsymbol{x})] = \mathrm{E}[U(\boldsymbol{x})] = m$. The unknown weight functions $\{w_i\}_{i=1}^{M}$ are determined by minimizing the variance of the estimation error $\varepsilon(\boldsymbol{x}) = \widehat{U}(\boldsymbol{x}) - U(\boldsymbol{x})$. In the literature, this variance is usually called estimation variance and denoted as σ_E^2. Taking the previous assumptions into account, the following expression for the estimation variance

can be derived

$$\sigma_E^2 = c(\mathbf{0}) + \sum_{i=1}^{M}\sum_{j=1}^{M} w_i w_j c(\boldsymbol{x}_i - \boldsymbol{x}_j) - 2\sum_{i=1}^{M} w_i c(\boldsymbol{x} - \boldsymbol{x}_i) \quad \text{(C.29)}$$

or in terms of the semi-variogram

$$\sigma_E^2 = -\sum_{i=1}^{M}\sum_{j=1}^{M} w_i w_j \gamma(\boldsymbol{x}_i - \boldsymbol{x}_j) + 2\sum_{i=1}^{M} w_i \gamma(\boldsymbol{x} - \boldsymbol{x}_i) \quad \text{(C.30)}$$

Setting the functional derivatives with respect to the weight functions equal to zero yields a linear system

$$\boldsymbol{C}\boldsymbol{w}(\boldsymbol{x}) = \boldsymbol{c}(\boldsymbol{x}) \quad \text{(C.31)}$$

with

$$\begin{aligned} \boldsymbol{C}_{i,j} &= c(\boldsymbol{x}_j - \boldsymbol{x}_i), \quad i,j = 1,\ldots,M \\ \boldsymbol{w}(\boldsymbol{x}) &= [w_1(\boldsymbol{x}),\ldots,w_M(\boldsymbol{x})]^T \\ \boldsymbol{c}(\boldsymbol{x}) &= [c(\boldsymbol{x} - \boldsymbol{x}_1),\ldots,c(\boldsymbol{x} - \boldsymbol{x}_M)]^T \end{aligned} \quad \text{(C.32)}$$

Or equivalently in terms of the semi-variogram

$$\boldsymbol{\Gamma}\boldsymbol{w}(\boldsymbol{x}) = \boldsymbol{\gamma}(\boldsymbol{x}) \quad \text{(C.33)}$$

with

$$\begin{aligned} \boldsymbol{\Gamma}_{i,j} &= \gamma(\boldsymbol{x}_j - \boldsymbol{x}_i), \quad i,j = 1,\ldots,M \\ \boldsymbol{w}(\boldsymbol{x}) &= [w_1(\boldsymbol{x}),\ldots,w_M(\boldsymbol{x})]^T \\ \boldsymbol{\gamma}(\boldsymbol{x}) &= [\gamma(\boldsymbol{x} - \boldsymbol{x}_1),\ldots,\gamma(\boldsymbol{x} - \boldsymbol{x}_M)]^T \end{aligned} \quad \text{(C.34)}$$

We note that the solution of this linear system defines the global minimum of the estimation variance. This minimum is called simple kriging variance σ_{sk}^2 and often serves as an precision estimator of the kriged interpolation value at point $\boldsymbol{x}$:

$$\sigma_{sk}^2 = c(\mathbf{0}) - \boldsymbol{w}^T\boldsymbol{c} \quad \text{or} \quad \sigma_{sk}^2 = \boldsymbol{w}^T\boldsymbol{\gamma} \quad \text{(C.35)}$$

As mentioned above, in the ansatz (C.28) the random variables $(U(\boldsymbol{x}_i))_{i=1}^M$ are replaced by their sampled values $(u_i)_{i=1}^M$ to get an estimate for $u(\boldsymbol{x})$. From the analysis above it becomes clear why Kriging is also known as best linear unbiased estimator.

C.2.2 Ordinary Kriging

Ordinary kriging is closely related to simple kriging. Whereas second-order stationarity still holds, it is not assumed that the expectation value m is known a priori. Consequently, the linear estimation ansatz must be slightly modified

$$\widehat{U}(\boldsymbol{x}) = \sum_{i=1}^{M} w_i(\boldsymbol{x}) U(\boldsymbol{x}_i) \tag{C.36}$$

To guarantee an unbiased estimator, it is required that

$$\sum_{i=1}^{M} w_i = 1 \tag{C.37}$$

It can be shown that this modification does not imply a change of the estimation variance σ_E^2, i.e. the expressions (C.29) and (C.30) are still valid. However, what does change is the minimization problem since the constraint from above has to be satisfied:

$$\begin{aligned} &\sigma_E^2 \to \min \\ &\text{subject to } \sum_{i=1}^{M} w_i = 1 \end{aligned} \tag{C.38}$$

To handle this constraint minimization problem, a Lagrangian multiplier $\lambda(\boldsymbol{x})$ is introduced, yielding a system of Euler-Lagrange equations

$$\begin{bmatrix} \boldsymbol{C} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \lambda \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} \\ 1 \end{bmatrix} \tag{C.39}$$

1 represents a $N \times 1$ vector with each element being 1. The other terms are defined as in section (C.2.1). Alternatively, the system can also be expressed in terms of the semi-variogram:

$$\begin{bmatrix} \boldsymbol{\Gamma} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \lambda \end{bmatrix} = \begin{bmatrix} \boldsymbol{\gamma} \\ 1 \end{bmatrix} \tag{C.40}$$

C.2.3 Universal Kriging

Essentially, universal Kriging can be considered as a generalization of ordinary Kriging. Instead of assuming a constant expectation value m of the random function, m is allowed to vary in space. To this end, a so-called trend model is introduced:

$$m(\boldsymbol{x}) = \sum_{k=1}^{L} g_k p_k(\boldsymbol{x}), \tag{C.41}$$

where $\{p_k\}_{k=1}^{L}$ is a basis of a predefined function space (usually, polynomials). Clearly, the first condition of the second-order stationarity is violated. Thus, a relaxed formulation of the second-order stationarity is introduced by replacing the violated condition with the trend model from above. This assumption forms the basis of universal Kriging. Interestingly, it does not affect the formulas for the estimation variance (eq. (C.29) and (C.29)). However, the linear estimator ansatz (C.36) is only unbiased, if

$$\sum_{i=1}^{M} w_i(\boldsymbol{x}) p_k(\boldsymbol{x}_i) = p_k(\boldsymbol{x}),; \quad k = 1, \ldots, L \tag{C.42}$$

Imposing these additional constraints leads to the following minimization problem:

$$\begin{aligned} &\sigma_E^2 \to \min \\ \text{s.t.} &\sum_{i=1}^{M} w_i(\boldsymbol{x}) p_k(\boldsymbol{x}_i) = p_k(\boldsymbol{x}), \quad k = 1, \ldots, L \end{aligned} \tag{C.43}$$

As in ordinary Kriging, we reformulate the problem using L Lagrangian multipliers (i.e. $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_L]^T$) and compute the Euler-Lagrange equations:

$$\begin{bmatrix} \boldsymbol{C} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{p} \end{bmatrix}, \tag{C.44}$$

or rewritten in terms of the semi-variogram function γ:

$$\begin{bmatrix} \boldsymbol{\Gamma} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\gamma} \\ \boldsymbol{p} \end{bmatrix}. \tag{C.45}$$

Here, the matrix $\boldsymbol{P}$ and the vector $\boldsymbol{p}$ are given by

$$\boldsymbol{P}_{i,j} = p_j(\boldsymbol{x}_i) \tag{C.46}$$

$$\boldsymbol{p}(\boldsymbol{x}) = [p_1(\boldsymbol{x}), \ldots, p_L(\boldsymbol{x})]^T, \tag{C.47}$$

while the rest is defined as in the previous sections. For an in-depth analysis of the various Kriging methods the interested reader is referred to *Wackernagel* (2013).

C.2.4 Dual formalism and connection to RBF interpolation

For most scattered data interpolation problems the number of data sites is quite large. Thus, to solve the resulting linear system iterative methods are preferred to direct solution methods. In the standard formalism of Kriging, the right-hand side of the linear system to be solved is a function of the evaluation point $\boldsymbol{x}$. From a computational point of view this is rather inefficient as for every evaluation point one has to solve a slightly different linear system. Fortunately, there exists a dual formalism of the Kriging method resulting in a linear system with a fixed right-hand side. Below, this alternative form is derived.

We begin by recalling the standard interpolation ansatz in vector

notation:

$$\widehat{u}(\boldsymbol{x}) = \boldsymbol{u}^T \boldsymbol{w}(\boldsymbol{x}). \tag{C.48}$$

In the case of universal Kriging, the generating functions $\boldsymbol{w}$ are found by solving the following linear system

$$\begin{bmatrix} \boldsymbol{C} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{p} \end{bmatrix} \tag{C.49}$$

In the first step, we solve this system for $\boldsymbol{w}$ symbolically:

1.) $\boldsymbol{C}\boldsymbol{w} + \boldsymbol{P}\boldsymbol{\lambda} = \boldsymbol{c} \qquad \left| \boldsymbol{P}^T\boldsymbol{C}^{-1}\cdot \right.$

$$\underbrace{\boldsymbol{P}^T\boldsymbol{w}}_{=\boldsymbol{p}} + \boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{P}\boldsymbol{\lambda} = \boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{c} \quad \left| -\boldsymbol{p},\ \left[\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{P}\right]^{-1}\cdot \right.$$

$$\boldsymbol{\lambda} = \left[\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{P}\right]^{-1}\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{c} - \left[\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{P}\right]^{-1}\boldsymbol{p} \tag{C.50}$$

2.) $\boldsymbol{C}\boldsymbol{w} + \boldsymbol{P}\boldsymbol{\lambda} = \boldsymbol{c} \qquad \left| \boldsymbol{C}^{-1}\cdot \right.$

$$\boldsymbol{w} = \boldsymbol{C}^{-1}\boldsymbol{c} - \boldsymbol{C}^{-1}\boldsymbol{P}\boldsymbol{\lambda} \tag{C.51}$$

3.) insert 1.) into 2.)

$$\boldsymbol{w} = \left[\boldsymbol{C}^{-1} - \boldsymbol{C}^{-1}\boldsymbol{P}\left[\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{P}\right]^{-1}\boldsymbol{P}^T\boldsymbol{C}^{-1}\right]\boldsymbol{c} + \boldsymbol{C}^{-1}\boldsymbol{P}\left[\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{P}\right]^{-1}\boldsymbol{p} \tag{C.52}$$

In the second step, the vector $\boldsymbol{w}$ in the ansatz C.48 is replaced by the expression above, i.e.

$$\widehat{u} = \underbrace{\boldsymbol{u}^T\left[\boldsymbol{C}^{-1} - \boldsymbol{C}^{-1}\boldsymbol{P}\left[\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{P}\right]^{-1}\boldsymbol{P}^T\boldsymbol{C}^{-1}\right]}_{=:\boldsymbol{e}^T}\boldsymbol{c} + \underbrace{\boldsymbol{u}^T\boldsymbol{C}^{-1}\boldsymbol{P}\left[\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{P}\right]^{-1}}_{=:\boldsymbol{d}^T}\boldsymbol{p} \tag{C.53}$$

As indicated above, we can rewrite the interpolation ansatz as

$$\widehat{u}(\boldsymbol{x}) = \boldsymbol{c}^T(\boldsymbol{x})\,\boldsymbol{e} + \boldsymbol{p}^T(\boldsymbol{x})\,\boldsymbol{d} \tag{C.54}$$

where

$$\boldsymbol{e} = \boldsymbol{C}^{-1}\boldsymbol{u} - \boldsymbol{C}^{-1}\boldsymbol{P}\left[\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{P}\right]^{-1}\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{u} \tag{C.55}$$

and

$$\boldsymbol{d} = \left[\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{P}\right]^{-1}\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{u} \tag{C.56}$$

Equation C.55 and C.56 can be rearranged as follows

$$\text{1.)}\quad \boldsymbol{e} = \boldsymbol{C}^{-1}\boldsymbol{u} - \boldsymbol{C}^{-1}\boldsymbol{P}\underbrace{\left[\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{P}\right]^{-1}\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{u}}_{=\boldsymbol{d}} \quad \left|\boldsymbol{C}\cdot,\ +\boldsymbol{P}\boldsymbol{d}\right.$$

$$\underline{\boldsymbol{C}\boldsymbol{e} + \boldsymbol{P}\boldsymbol{d} = \boldsymbol{u}} \tag{C.57}$$

$$\text{2.)}\quad \boldsymbol{d} = \left[\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{P}\right]^{-1}\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{u} \quad \left|\boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{P}\cdot\right.$$

$$\boldsymbol{P}^T\boldsymbol{C}^{-1}\underbrace{\boldsymbol{P}\boldsymbol{d}}_{=\boldsymbol{u}-\boldsymbol{C}\boldsymbol{e}} = \boldsymbol{P}^T\boldsymbol{C}^{-1}\boldsymbol{u}$$

$$\underline{\boldsymbol{P}^T\boldsymbol{e} = \boldsymbol{0}} \tag{C.58}$$

To put it another way, the unknown coefficient vectors $\boldsymbol{e}$ and $\boldsymbol{d}$ can be determined by solving the following linear system

$$\begin{bmatrix} \boldsymbol{C} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{bmatrix}\begin{bmatrix} \boldsymbol{e} \\ \boldsymbol{d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{0} \end{bmatrix} \tag{C.59}$$

To sum up, in the dual formalism of Kriging, the esimation is expressed by a sum of two terms (see eq. C.54). The unknown coefficients are found by solving the linear system given in equation C.59. Comparison of equation C.54 and C.12 shows that Kriging and RBF with polynomial precision are identical provided that the correlation function c and the radial function Φ are equal.

C.3 Partition of Unity Method

The partition of unity approach is an efficient interpolation and approximation method introduced by *Babuška and Melenk* (1997) as part of a meshfree solution algorithm for partial differential equations. As shown above, most global interpolation/approximation methods (e.g. RBF, Kriging) result in solving a large linear system. The partition of unity method (PUM) offers a simple alternative by splitting the global problem into many smaller subproblems which are independent of each other and therefore can be solved in parallel. A two-step strategy is applied:

(I) **Decomposition:** The domain of interest $\Omega \in \mathbb{R}^d$ is split into S subdomains $\{\Omega_i\}_{i=1}^{S}$. The subdomains need to fulfill two conditions:

(i) The union of the subdomains must contain the underlying domain, i.e. $\bigcup_{i=1}^{S} \Omega_i \subseteq \Omega$

(ii) To guarantee continuity of the global interpolant the subdomains should slightly overlap among each other.

For each subdomain a separate interpolation or approximation problem is solved considering only the data points located within this subdomain. One point worth noting here is that the method to solve these subproblems is not restricted to a certain type, i.e. all interpolation/approximation techniques (e.g. RBF, Kriging, polynomial fit) are equally valid. For this reason, we denote the solution of subdomain Ω_i as $\widehat{u}_i(\boldsymbol{x})$ without further clarification.

(II) **Assembly:** Every subdomain is assigned to a personal weight function $w_i(\boldsymbol{x})$. These functions are continuous, bounded between 0 and 1 and their support is limited to the region of the corresponding subdomain. Moreover, it is required that at every point in Ω the sum of these functions is 1, i.e.

$$\sum_{i=1}^{S} w_i(\boldsymbol{x}) = 1, \quad \forall \boldsymbol{x} \in \Omega \tag{C.60}$$

A common way to design these weight functions is to use the Shepard method. In mathematical terms this means

$$w_j(\boldsymbol{x}) = \frac{\chi_{\Omega_j}(\boldsymbol{x})}{\sum_{i=1}^{S} \chi_{\Omega_i}(\boldsymbol{x})} \tag{C.61}$$

with χ_{Ω_i} being the indicator function of the subdomain Ω_i, i.e.

$$\chi_{\Omega_i}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in \Omega_i \\ 0 & \text{otherwise} \end{cases} \tag{C.62}$$

Appendix D

Derivation of a simple error formula

As shown in chapter 4.5.1 the generalized approximation error is defined by

$$e_{\boldsymbol{\gamma}}(\boldsymbol{x}_0) = \widehat{\partial^{\boldsymbol{\gamma}} u}(\boldsymbol{x}_0) - \partial^{\boldsymbol{\gamma}} u(\boldsymbol{x}_0) \tag{D.1}$$

To derive an analytical expression for this error it is assumed that the unknown function $u(\boldsymbol{x})$ is $(m+1)$-times differentiable, i.e. u belongs to $\mathcal{C}^{m+1}$. By this assumption and by Taylor's theorem, $u(\boldsymbol{x})$ can be expressed as follows

$$u(\boldsymbol{x}) = \sum_{|\boldsymbol{\alpha}| \leq m} \frac{\partial^{\boldsymbol{\alpha}} u(\boldsymbol{x}_0)}{\boldsymbol{\alpha}!} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\alpha}} + \sum_{|\boldsymbol{\beta}| = m+1} R_{\boldsymbol{\beta}}(\boldsymbol{x}) (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\beta}} \tag{D.2}$$

where the remainder term is

$$R_{\boldsymbol{\beta}}(\boldsymbol{x}) = \frac{1}{\boldsymbol{\beta}!} \partial^{\boldsymbol{\beta}} u(\boldsymbol{x}_0 + \eta_{\boldsymbol{\beta}} (\boldsymbol{x} - \boldsymbol{x}_0)), \quad \eta_{\boldsymbol{\beta}} \in [0,1]. \tag{D.3}$$

As before the multi-index notation is used. The derivative of the generating functions can be deduced from eq. (4.35):

$$\partial^{\boldsymbol{\gamma}} \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{x}_0) = \boldsymbol{W} \boldsymbol{B} \boldsymbol{G}^{-1} \partial^{\boldsymbol{\gamma}} \boldsymbol{b}(\boldsymbol{x} - \boldsymbol{x}_0) \tag{D.4}$$

The MLS approximant around $\boldsymbol{x}_0$ can be retrieved from formula (4.28):

$$\widehat{\partial^{\boldsymbol{\gamma}} u}(\boldsymbol{x}, \boldsymbol{x}_0) = \sum_{i=1}^{M} \tilde{u}_i \partial^{\boldsymbol{\gamma}} \psi_i(\boldsymbol{x}, \boldsymbol{x}_0), \tag{D.5}$$

where $\{\boldsymbol{x}_i\}_{i=1}^{M}$ are the sampling points in the support of the weight function. Further as indicated by the tilde, the measured values $\{\tilde{u}_i\}_{i=1}^{M}$ are assumed to be contaminated by arbitrary noise:

$$\tilde{u}_i = u(\boldsymbol{x}_i) + \epsilon_i, \quad i = 1, \ldots, M. \tag{D.6}$$

Replacing $u(\boldsymbol{x}_i)$ by the Taylor series introduced above and substituting expression (D.6) into equation (D.5) yields

$$\begin{aligned}\widehat{\partial^{\boldsymbol{\gamma}} u}(\boldsymbol{x},\boldsymbol{x}_0) = {} & \partial^{\boldsymbol{\gamma}} \sum_{i=1}^{M} \psi_i(\boldsymbol{x},\boldsymbol{x}_0) \sum_{|\boldsymbol{\alpha}|\le m} \frac{\partial^{\boldsymbol{\alpha}} u(\boldsymbol{x}_0)}{\boldsymbol{\alpha}!} (\boldsymbol{x}_i - \boldsymbol{x}_0)^{\boldsymbol{\alpha}} \\ & + \sum_{i=1}^{M} \partial^{\boldsymbol{\gamma}} \psi_i(\boldsymbol{x},\boldsymbol{x}_0) \left[\epsilon_i + \sum_{|\boldsymbol{\beta}|=m+1} R_{\boldsymbol{\beta}}(\boldsymbol{x}_i) (\boldsymbol{x}_i - \boldsymbol{x}_0)^{\boldsymbol{\beta}} \right]\end{aligned} \tag{D.7}$$

Due to the polynomial reproduction property, this equation can be simplified to

$$\begin{aligned}\widehat{\partial^{\boldsymbol{\gamma}} u}(\boldsymbol{x},\boldsymbol{x}_0) = {} & \sum_{\substack{|\boldsymbol{\alpha}|\le m| \\ \boldsymbol{\alpha}\ge\boldsymbol{\gamma}}} \frac{\partial^{\boldsymbol{\alpha}} u(\boldsymbol{x}_0)}{(\boldsymbol{\alpha}-\boldsymbol{\gamma})!} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\alpha}-\boldsymbol{\gamma}} \\ & + \sum_{i=1}^{M} \partial^{\boldsymbol{\gamma}} \psi_i(\boldsymbol{x},\boldsymbol{x}_0) \left[\epsilon_i + \sum_{|\boldsymbol{\beta}|=m+1} R_{\boldsymbol{\beta}}(\boldsymbol{x}_i) (\boldsymbol{x}_i - \boldsymbol{x}_0)^{\boldsymbol{\beta}} \right]\end{aligned} \tag{D.8}$$

By evaluating this equation at $\boldsymbol{x}_0$, the first term on the right hand side becomes $\partial^{\boldsymbol{\gamma}} u(\boldsymbol{x}_0)$. Consequently, the formula for the approximation error at $\boldsymbol{x}_0$ reduces to:

$$e_{\boldsymbol{\gamma}}(\boldsymbol{x}_0) = \sum_{i=1}^{M} \partial^{\boldsymbol{\gamma}} \psi_i(\boldsymbol{x}_0,\boldsymbol{x}_0) \left[\epsilon_i + \sum_{|\boldsymbol{\beta}|=m+1} R_{\boldsymbol{\beta}}(\boldsymbol{x}_i) (\boldsymbol{x}_i - \boldsymbol{x}_0)^{\boldsymbol{\beta}} \right]. \tag{D.9}$$

Since only the sampling points $\{\boldsymbol{x}_i\}_{i=1}^{M}$ in the support of the weight function $\Omega_0 = \operatorname{supp}[w(\cdot,\boldsymbol{x}_0)]$ are taken into account, the upper bound of $\left|R_{\boldsymbol{\beta}}(\boldsymbol{x}_i)\right|$ can be written as follows

$$\left|R_{\boldsymbol{\beta}}(\boldsymbol{x}_i)\right| \le \frac{1}{(m+1)!} \max_{|\boldsymbol{\alpha}|=m+1} \left\|\partial^{\boldsymbol{\alpha}} u(\boldsymbol{x})\right\|_{L_\infty(\Omega_0)}, \quad \left|\boldsymbol{\beta}\right| = m+1 \tag{D.10}$$

Further, let the magnitude of the data errors be limited by ϵ, i.e.

$$\left|\epsilon_i\right| \le \varepsilon, \quad i = 1,\ldots,M. \tag{D.11}$$

Then, it follows that the error $e_{\boldsymbol{\gamma}}$ is bounded by

$$\begin{aligned} \left|e_{\boldsymbol{\gamma}}\left(\boldsymbol{x}_0\right)\right| \leq\ & \Lambda_{m,\boldsymbol{\gamma}}(\boldsymbol{x}_0)\,\varepsilon \\ & + \frac{C\left(\boldsymbol{x}_0\right)}{(m+1)!}\sum_{i=1}^{M}\left|\partial^{\boldsymbol{\gamma}}\psi_i\left(\boldsymbol{x}_0,\boldsymbol{x}_0\right)\right| \sum_{|\boldsymbol{\beta}|=m+1}\left|\boldsymbol{x}_i-\boldsymbol{x}_0\right|^{\boldsymbol{\beta}} \end{aligned} \tag{D.12}$$

with

$$C\left(\boldsymbol{x}_0\right) = \max_{|\boldsymbol{\alpha}|=m+1}\left\|\partial^{\boldsymbol{\alpha}} u\left(\boldsymbol{x}\right)\right\|_{L_\infty(\Omega_0)} \tag{D.13}$$

and $\Lambda_{m,\boldsymbol{\gamma}}$ being the Lebesgue function

$$\Lambda_{m,\boldsymbol{\gamma}}(\boldsymbol{x}_0) = \sum_{i=1}^{M}\left|\partial^{\boldsymbol{\gamma}}\psi_i(\boldsymbol{x}_0,\boldsymbol{x}_0)\right| \tag{D.14}$$

Appendix E

Smoothed random paths

This section contains a brief description of how the smoothed random paths, which are used for demonstration purposes in chapter 4, are generated. If compared with the probe path of a ProCap measurement, a 'standard' random walk seems to be a poor imitation, since the changes in direction are rather sharp. A smoother random walk is obtained by introducing a specific force $\boldsymbol{f}$ (acceleration) that undergoes a Wiener process with zero drift and an adjustable volatility $\sigma_{\Delta t} = \sigma\sqrt{\Delta t}$. Let Δt be the time step, integrating the specific force twice by using an explicit Euler scheme leads to the new position of the virtual particle. The initial position, velocity and acceleration are denoted by $\boldsymbol{x}_0$, $\boldsymbol{u}_0$ and $\boldsymbol{f}_0$, respectively. To better control the random walk, two corrective actions are applied:

(i) the magnitude of the specific force is limited by a parameter f_{max}

(ii) a decelerating force (drag) that is proportional to the velocity square is implemented. The proportionality factor is denoted by β.

To restrict the particle coordinates to the simulating box, periodic boundary conditions are applied. For the numerical experiments in chapter 4, the following settings were chosen:

$$\begin{array}{ll} \Delta t = 0.01 & \boldsymbol{x}_0 \sim \text{uniform in } [0,1] \times [0,1] \\ \#\text{ steps } N = 25000 & \boldsymbol{u}_0 = \boldsymbol{0} \\ \sigma_T = 0.01 & \boldsymbol{f}_0 = \boldsymbol{0} \\ \beta = 5 & f_{max} = 0.1 \end{array} \tag{E.1}$$

The algorithm for a smoothed random path in a unit square is outlined below.

Smoothed Random Path

Input:

time step size Δt	initial position $\boldsymbol{x}_0$
number of steps N	initial velocity $\boldsymbol{u}_0$
force variance σ_T^2	initial force $\boldsymbol{f}_0$
drag coefficient β	force limit f_{max}

Output:

random path $\{\boldsymbol{x}_k\}_{k=1}^{R}$

begin

  for $k \leftarrow 1$ **to** N **do**

    $\boldsymbol{f}_k \leftarrow$random vector $\sim \mathcal{N}(\boldsymbol{f}_{k-1}, \sigma_T^2 \boldsymbol{I})$

    if $|\boldsymbol{f}_k| > \boldsymbol{f}_{max}$ **then**

      $\boldsymbol{f}_k \leftarrow f_{max} \frac{\boldsymbol{f}_k}{|\boldsymbol{f}_k|}$

    end

    $\boldsymbol{a}_k \leftarrow \boldsymbol{f}_k - \beta |\boldsymbol{u}_{k-1}| \boldsymbol{u}_{k-1}$

    $\boldsymbol{u}_k \leftarrow \boldsymbol{u}_{k-1} + \Delta t\, \boldsymbol{a}_k$

    $\boldsymbol{x}_k \leftarrow \boldsymbol{x}_{k-1} + \Delta t\, \boldsymbol{u}_k$

    $\boldsymbol{x}_k \leftarrow \boldsymbol{x}_k$ modulo 1

  end

end

Bibliography

Ahmed, S., G. Ramm, and G. Faltin (1984), Some salient features of the time-averaged ground vehicle wake, *Tech. rep.*, SAE Technical Paper.

Alfeld, P. (1989), Scattered data interpolation in three or more variables, *Mathematical methods in computer aided geometric design*, pp. 1–33.

Azijli, I., and R. P. Dwight (2015), Solenoidal filtering of volumetric velocity measurements using gaussian process regression, *Experiments in Fluids*, *56*(11), 1–18.

Babuška, I., and J. M. Melenk (1997), The partition of unity method, *International journal for numerical methods in engineering*, *40*(4), 727–758.

Backus, G., and F. Gilbert (1967), Numerical applications of a formalism for geophysical inverse problems, *Geophysical Journal International*, *13*(1-3), 247–276.

Backus, G., and F. Gilbert (1968), The resolving power of gross earth data, *Geophysical Journal International*, *16*(2), 169–205.

Backus, G., and F. Gilbert (1970), Uniqueness in the inversion of inaccurate gross earth data, *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, *266*(1173), 123–192.

Bauer, M. (2007), Tracking errors in augmented reality, Ph.D. thesis, Technische Universität München.

Bird, J. D., and D. R. Riley (1952), *Some Experiments on Visualization of Flow Fields Behind Low-Aspect-Ratio Wings by Means of a Tuft Grid*, National Advisory Committee for Aeronautics.

Borer, D. J. (2014), 4d flow visualization with dynamic vision sensors, Ph.D. thesis, ETH Zürich.

Bos, L., and K. Salkauskas (1989), Moving least-squares are backus-gilbert optimal, *Journal of Approximation Theory*, *59*(3), 267–275.

Bryer, D. W., and R. C. Pankhurst (1971), *Pressure-probe methods for determining wind speed and flow direction*, Her Majesty's Stationery Office.

Burrage, K. (1993), Parallel methods for initial value problems, *Applied Numerical Mathematics*, *11*(1-3), 5–25.

Cleveland, W. S. (1979), Robust locally weighted regression and smoothing scatterplots, *Journal of the American statistical association*, *74*(368), 829–836.

Coutsias, E. A., C. Seok, and K. A. Dill (2004), Using quaternions to calculate rmsd, *Journal of computational chemistry*, *25*(15), 1849–1857.

Dominy, R., and H. Hodson (1992), An investigation of factors influencing the calibration of 5-hole probes for 3-d flow measurements, in *ASME 1992 International Gas Turbine and Aeroengine Congress and Exposition*, American Society of Mechanical Engineers.

Dudzinski, T. J., and L. N. Krause (1969), Flow-direction measurement with fixed-position probes.

Fan, J., and I. Gijbels (1996), *Local polynomial modelling and its applications: monographs on statistics and applied probability 66*, vol. 66, CRC Press.

Fasshauer, G. E. (2007), *Meshfree approximation methods with MATLAB*, vol. 6, World Scientific.

Flay, R. G. (1996), A twisted flow wind tunnel for testing yacht sails, *Journal of Wind Engineering and Industrial Aerodynamics*, *63*(1-3), 171–182.

Franke, R. (1982), Scattered data interpolation: Tests of some methods, *Mathematics of computation*, *38*(157), 181–200.

Franke, R., and G. M. Nielson (1991), Scattered data interpolation and applications: A tutorial and survey, in *Geometric Modeling*, pp. 131–160, Springer.

Gallington, R. (1980), Measurement of very large flow angles with non-nulling seven-hole probes, *USAFA-TR-80-17*, pp. 60–88.

Hall, B. F., and T. Povey (2017), The oxford probe: an open access five-hole probe for aerodynamic measurements, *Measurement Science and Technology*, *28*(3), 035,004.

Hinterberger, C., M. Garcia-Villalba, and W. Rodi (2004), Large eddy simulation of flow around the ahmed body, in *The aerodynamics of heavy vehicles: trucks, buses, and trains*, pp. 77–87, Springer.

Hong, J.-M., J.-C. Yoon, and C.-H. Kim (2008), Divergence-constrained moving least squares for fluid simulation, *Computer Animation and Virtual Worlds*, *19*(3-4), 469–477.

Huerta, A., Y. Vidal, and P. Villon (2004), Pseudo-divergence-free element free galerkin method for incompressible fluid flow, *Computer Methods in Applied Mechanics and Engineering*, *193*(12), 1119–1136.

Kabsch, W. (1976), A solution for the best rotation to relate two sets of vectors, *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, *32*(5), 922–923.

Knutsson, H., and C.-F. Westin (1993), Normalized and differential convolution, in *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on*, pp. 515–523, IEEE.

Kundu, P. K., I. M. Cohen, and D. R. Dowling (2016), *Fluid Mechanics*, 6th edition ed., Academic Press.

Lancaster, P., and K. Salkauskas (1981), Surfaces generated by moving least squares methods, *Mathematics of computation*, *37*(155), 141–158.

Landolt, A., D. Borer, A. Meier, and T. Roesgen (2016), Quantitative flow visualization applied to a passive wake control problem, in *The Aerodynamics of Heavy Vehicles III*, pp. 413–425, Springer.

Levin, D. (1998), The approximation power of moving least-squares, *Mathematics of Computation of the American Mathematical Society*, *67*(224), 1517–1531.

Lienhart, H., C. Stoots, and S. Becker (2002), Flow and turbulence structures in the wake of a simplified car model (ahmed modell), in *New Results in Numerical and Experimental Fluid Mechanics III*, pp. 323–330, Springer.

Lipman, Y. (2009), Stable moving least-squares, *Journal of Approximation Theory*, *161*(1), 371–384.

Lipman, Y., D. Cohen-Or, and D. Levin (2006), Error bounds and optimal neighborhoods for mls approximation, in *Proceedings of the fourth Eurographics symposium on Geometry processing*, pp. 71–80, Eurographics Association.

Liu, X.-H. (2001), Kernel smoothing for spatially correlated data, Ph.D. thesis, Iowa State University.

Lorensen, W. E., and H. E. Cline (1987), Marching cubes: A high resolution 3d surface construction algorithm, in *ACM siggraph computer graphics*, vol. 21, pp. 163–169, ACM.

Lowitzsch, S. (2002), Approximation and interpolation employing divergence–free radial basis functions with applications, Ph.D. thesis, Citeseer.

Mirzaei, D., R. Schaback, and M. Dehghan (2011), On generalized moving least squares and diffuse derivatives, *IMA Journal of Numerical Analysis*.

Mitas, L., and H. Mitasova (1999), Spatial interpolation, *Geographical information systems: principles, techniques, management and applications*, *1*, 481–492.

Mueller, A., A. Landolt, and T. Roesgen (2012), Probe capture for quantitative flow visualization in large scale wind tunnels, in *28th Aerodynamic Measurement Technology, Ground Testing, and Flight Testing Conference including the Aerospace T&E Days Forum*, p. 3317.

Narcowich, F. J., and J. D. Ward (1994), Generalized hermite interpolation via matrix-valued conditionally positive definite functions, *Mathematics of Computation*, *63*(208), 661–687.

Nayroles, B., G. Touzot, and P. Villon (1992), Generalizing the finite element method: diffuse approximation and diffuse elements, *Computational mechanics*, *10*(5), 307–318.

Paul, A. R., R. R. Upadhyay, and A. Jain (2011), A novel calibration algorithm for five-hole pressure probe, *International Journal of Engineering, Science and Technology*, *3*(2), 89–95.

Pham, T. Q., L. J. Van Vliet, and K. Schutte (2006), Robust fusion of irregularly sampled data using adaptive normalized convolution, *EURASIP Journal on Advances in Signal Processing*, *2006*(1), 1–12.

Pisasale, A., and N. Ahmed (2002), A novel method for extending the calibration range of five-hole probe for highly three-dimensional flows, *Flow Measurement and Instrumentation*, *13*(1), 23–30.

Ruppert, D., and M. P. Wand (1994), Multivariate locally weighted least squares regression, *The annals of statistics*, pp. 1346–1370.

Shepard, D. (1968), A two-dimensional interpolation function for irregularly-spaced data, in *Proceedings of the 1968 23rd ACM national conference*, pp. 517–524, ACM.

Song, S. M., S. Napel, G. H. Glover, and N. J. Pelc (1993), Noise reduction in three-dimensional phase-contrast mr velocity measurementsl, *Journal of Magnetic Resonance Imaging*, *3*(4), 587–596.

Stone, C. J. (1977), Consistent nonparametric regression, *The annals of statistics*, pp. 595–620.

Telionis, D., Y. Yang, and O. Rediniotis (2009), Recent developments in multi-hole probe (mhp) technology, in *20th International Congress of Mechanical Engineering*, vol. 21.

Treaster, A., and A. Yocum (1979), Calibration and application of 5-hole probes, *ISA transactions*, *18*(3), 23–34.

Wackernagel, H. (2013), *Multivariate geostatistics: an introduction with applications*, Springer Science & Business Media.

Wand, M. P., and M. C. Jones (1994), *Kernel smoothing*, Crc Press.

Wendland, H. (2001), Local polynomial reproduction and moving least squares approximation, *IMA Journal of Numerical Analysis*, *21*(1), 285–300.

Wendland, H. (2004), *Scattered data approximation*, vol. 17, Cambridge university press.

Zilliac, G. (1993), Modelling, calibration, and error analysis of seven-hole pressure probes, *Experiments in Fluids*, *14*(1), 104–120.