Implementierung einzelner Analysemethoden in eine Auswerte-Toolbox für experimentell gemessene Geschwindigkeitsfelder

BACHELORARBEIT

für die Prüfung zum

Bachelor of Engineering

des Studienganges Informationstechnik

an der Dualen Hochschule Baden-Württemberg Mannheim

von

Fabio Seel

Bearbeitungszeitraum:	24.06. - 28.06.2019; 08.07. - 23.09.2019 (12 Wochen)
Matrikelnummer, Kurs:	6529562, TINF16ITIN
Ausbildungsfirma:	Deutsches Zentrum für Luft- und Raumfahrt e.V. in der Helmholtz-Gemeinschaft
Betreuer der Ausbildungsfirma:	Dr. Stefan Koch Christina Voß
Gutachter der Dualen Hochschule:	Prof. Dr. Rainer Colgen

Erklärung

Ich versichere hiermit, dass ich meine Bachelorarbeit mit dem Thema „*Implementierung einzelner Analysemethoden in eine Auswerte-Toolbox für experimentell gemessene Geschwindigkeitsfelder*“ selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.
Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

_____

Göttingen, den 20. September 2019

Kurzfassung

Name: Fabio Seel **Kurs:** TINF16ITIN

Implementierung einzelner Analysemethoden in eine Auswerte-Toolbox für experimentell gemessene Geschwindigkeitsfelder

Strömungsvorgänge in Fluiden spielen in Naturwissenschaft und Technik eine bedeutende Rolle. Um sie besser verstehen zu können, werden verschiedene Messverfahren eingesetzt und numerische Simulationen durchgeführt. Mit Hilfe der gewonnenen Daten können eine Vielzahl von Analysen betrieben werden. In der vorliegenden Arbeit geht es um die Implementierung zwei dieser Auswertemethoden, der Empirical Mode Decomposition (EMD) und der Clauser Chart Methode. Insbesondere die EMD stellt dabei in der Strömungsanalyse ein relativ neues Verfahren dar. Mit der Multidimensional Ensemble EMD (MEEMD) und der Fast and Adaptive Multivariate and Multidimensional EMD (FA-MVEMD) werden zwei Variationen der EMD betrachtet und hinsichtlich ihrer Anwendbarkeit in der Strömungsanalyse untersucht. Das zweite implementierte Verfahren, die Clauser Chart Methode, kann hingegen als Standardverfahren betrachtet werden und dazu genutzt werden, Messergebnisse sowie die Ergebnisse der EMD zu verifizieren.
Sowohl EMD, als auch die Clauser Chart Methode werden erfolgreich in eine Toolbox eingebunden und getestet. Im Hinblick auf die EMD zeigen sowohl die MEEMD, als auch die FA-MVEMD keine voll zufriedenstellende Lösung. Für einen der größten Nachteile der MEEMD, die hohe Berechnungszeit, wird daher eine Möglichkeit vorgestellt, den Rechenaufwand zu reduzieren.

Abstract

Name: Fabio Seel **Kurs:** TINF16ITIN

Implementation of individual analytical methods in an evaluation toolbox for experimentally measured velocity fields

Flow processes in fluids play an important role in science and technology. To gain a better understanding, different measurement methods and numerical simulations are performed. Various analytical methods can be applied to the collected data. This thesis deals with the implementation of two of those methods, the Empirical Mode Decomposition (EMD) and the Clauser Chart Method. Especially the EMD is a relatively new approach to be used in the analysis of flow data. Hence, with the Multidimensional Ensemble EMD (MEEMD) and the Fast and Adaptive Multivariate and Multidimensional EMD (FA-MVEMD) two variations of EMD are considered and examined regarding their applicability in the analysis of flow data. The second method to be implemented, the Clauser Chart Method, is a standard procedure and can be used to verify measurement results and the results of the EMD.
Both methods are succesfully integrated into a toolbox and tested. When considering EMD, neither of the MEEMD and FA-MVEMD can show completely satisfying results. Therefore, one of the biggest disadvantages of the MEEMD is alleviated by presenting a possibility to reduce the computing cost.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

2D	zweidimensional
3D	dreidimensional
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BEMD	Bidimensional Empirical Mode Decomposition
CEEMDAN	Complete Ensemble Empirical Mode Decomposition with Added Noise
DLR	Deutsches Zentrum für Luft- und Raumfahrt
EEMD	Ensemble Empirical Mode Decomposition
EMD	Empirical Mode Decomposition
FA-MVEMD	Fast and Adaptive Multivariate and Multidimensional Empirical Mode Decomposition
GUI	Graphical User Interface
HGK	Hochgeschwindigkeitskonfigurationen
IMF	Intrinsic Mode Function
MEEMD	Multidimensional Ensemble Empirical Mode Decomposition
Min	Minimum
Max	Maximum
NetCDF	Network Common Data Format
PIV	Particle Image Velocimetry
SDK	Software Development Kit
STB	Shake-The-Box

Lateinische Symbole

$\vec{a}$, $\vec{b}$	Koeffizienten einer beliebigen Funktion
A	Fläche (m²)
B	additive Konstante bei Clauser Charts
e_{min}, e_{max}	Einhüllende eines Signals, Interpolation durch Minima und Maxima
F	Kraft (N)
$F_{\|\|}$	Scherkraft (N)
i	Index-Variable
h	beliebiges Signal
m	Mittelwert / gemittelte Funktion
M	Ensemble Größe
S	S-Nummer, Abbruchkriterium
u, U	Geschwindigkeit in X-Richtung (m/s)
u^+	normierte Strömungsgeschwindigkeit
u_τ	Wandschubspannungsgeschwindigkeit (m/s)
v	Geschwindigkeit in Y-Richtung (m/s)
w	Geschwindigkeit in Z-Richtung (m/s)
X	Position in Raumrichtung (mm)
Y	Position in Raumrichtung (mm)
y^+	normierter Wandabstand
y_{Wand}	Wandposition (mm)
Z	Position in Raumrichtung (mm)

Griechische Symbole

κ von Karman Konstante
μ Toleranzwert (Abbruchkriterium)
ν kinematische Viskosität (m^2/s)
ρ Dichte des Fluids (kg/m^3)
τ Wandschubspannung (N/m^2)
φ Richtungsvektor

1 Einleitung

Ein zentraler Bestandteil der Forschungsarbeit auf dem Gebiet der Strömungsmechanik ist die Analyse von experimentell gewonnen Daten. Ziel ist es, die Strömungsvorgänge besser zu verstehen und hierzu quantitative Aussagen treffen zu können. Dabei werden unterschiedlichste Messverfahren eingesetzt, um die zur Beschreibung der Strömung notwendigen Größen messen zu können. Auch für die auf eine Messung folgende Analyse stehen eine Reihe von Methoden zur Verfügung. Sowohl auf der Seite der Messverfahren, als auch auf der Seite der Analysemethoden gibt es immer wieder neue Ansätze und eine stetige Weiterentwicklung der verwendeten Verfahren.

1.1 Motivation

Die vorliegende Arbeit entstand am Deutschen Zentrum für Luft- und Raumfahrt (DLR) in der Abteilung Hochgeschwindigkeitskonfigurationen (HGK) des Instituts für Aerodynamik und Strömungstechnik. In dieser Abteilung werden vor allem Strömungen im trans- und hypersonischen Bereich untersucht. Solche Strömungen zeichnen sich dadurch aus, dass Schallgeschwindigkeit oder ein Vielfaches dieser erreicht wird. Dabei werden unter anderem die Messverfahren *Particle Image Velocimetry* (PIV) und *Shake-The-Box* (STB) eingesetzt. Mit beiden kann ein Geschwindigkeitsfeld gemessen werden - also eine Momentaufnahme der Strömung, bei der Orten im Raum Geschwindigkeiten zugeordnet sind. Um die Analyse der experimentell gewonnenen Daten zu vereinfachen und zu beschleunigen, entstand die Idee der Entwicklung einer Auswerte-Toolbox. Die Motivation hierfür begründet sich darin, dass die Auswertungen dieser Messungen bisher durch eine Vielzahl verschiedener Programme und eigener Implementierung der Wissenschaftler getätigt wird. Eines der neuesten Verfahren, das bei der Analyse solcher Geschwindigkeitsfelder angewandt werden kann, ist die *Empirical Mode Decomposition* (EMD). Dieses an sich noch sehr junge Verfahren findet in letzter Zeit immer mehr in unterschiedlichsten Bereichen Anwendung - von Bildverarbeitung über Wirtschaftstrends bis hin zu Geowissenschaften. Im Bereich der Strömungsanalyse kann durch die EMD

ein Geschwindigkeitsfeld in mehrere additive Komponenten aufgeteilt werden. Jede dieser Komponenten, genannt Moden, enthält in etwa gleichgroße Strukturen - also Gebiete, in denen die Geschwindigkeit von dem Durchschnitt abweicht. Nachdem die Moden von dem Geschwindigkeitsfeld abgezogen wurden, bleibt das sogenannte Residuum über. Ein Beispiel dafür ist in Abbildung 1.1 gezeigt.

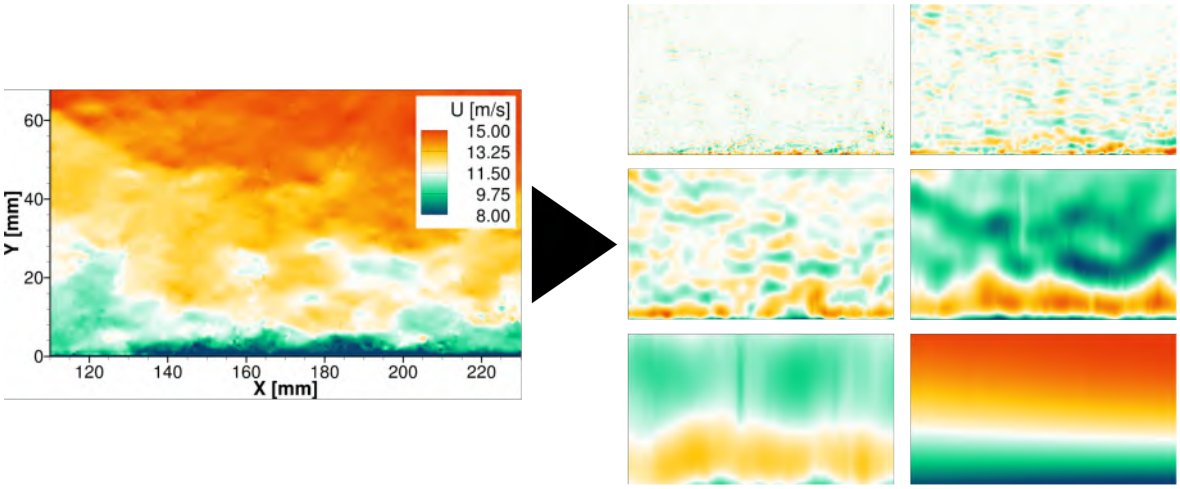


Abbildung 1.1: Schematische EMD-Zerlegung eines Geschwindigkeitsfeldes (links) in verschiedene Moden und Residuum (rechts)

Unter dem Gesichtspunkt der Aerodynamik ist es nun interessant zu betrachten, inwiefern große Strukturen, die verstärkt in größerem Abstand zu einem umströmten Objekt auftreten, die kleinen Strukturen beeinflussen, die in unmittelbarere Nähe des Objektes vorzufinden sind. Um dies sinnvoll auswerten zu können, muss die Position der Oberfläche des Objektes möglichst genau bekannt sein. Die *Clauser Chart Methode* stellt eine Möglichkeit zur Verfügung, diese Wandposition aus den gemessenen Geschwindigkeitsdaten zu bestimmen. Die Hauptaufgabe der Methode ist allerdings, die Wandschubspannungsgeschwindigkeit zu bestimmen - einer charakteristischen Geschwindigkeit, die auch zur Normierung eingesetzt wird und so einen Vergleich unterschiedlicher Versuchskonfigurationen ermöglicht. Zusätzlich ermöglicht sie zu untersuchen, wie nah an der Wand Daten aufgenommen werden konnten - und damit, ob die für die EMD-Analyse interessanten Bereiche überhaupt in der Messung enthalten sind.
Basierend auf diesen Punkten stellt die Implementierung der *Empirical Mode Decomposition* in Kombination mit der *Clauser Chart Methode* eine große Bereiche-

rung für die Analysetoolbox dar. Es wird ermöglicht, die Strömung unter neuen Gesichtspunkten zu erforschen und neue Erkenntnisse zu gewinnen.

1.2 Anforderungen und Ziele

Das Ziel dieser Arbeit ist es, in der Toolbox die beiden Verfahren so zu implementieren, dass sie problemlos ergänzend zu bereits bestehenden Auswertungen genutzt werden können. So muss betrachtet werden, wie die Toolbox erweitert werden muss, um dem Nutzer für die Verfahren eine gute Anwendbarkeit bieten zu können. Während die *Clauser Chart Methode* in der Strömungsmechanik als „Standardverfahren" betrachtet werden kann, ist die Nutzung der EMD zur strömungs-physikalischen Untersuchung noch relativ neu. Daher muss für diese Arbeit auch geprüft werden, welche Probleme bei der Anwendung der EMD bei Geschwindigkeitsfeldern auftreten und wie diese gelöst werden können. Dabei ist auch der Zeitbedarf für die Berechnungen ein wichtiger Faktor, da mit der Toolbox eine große Menge an Daten ausgewertet werden soll. Die genauen Anforderungen werden in den jeweiligen Kapiteln erarbeitet.

1.3 Vorgehen

Kern dieser Arbeit ist die Implementierungen der *Empirical Mode Decomposition* (EMD) und der *Clauser Chart Methode* in die Auswertetoolbox. Da das benötigte Hintergrundwissen nur eine geringe Schnittmenge hat, ist der zugehörige Theorieteil jeweils in den betreffenden Kapiteln vorzufinden.
Zunächst wird in Kapitel 2 aber eine kurze Vorstellung des Anwendungskontextes gegeben, darauffolgend wird in Kapitel 3 der Stand der Toolbox zu Beginn der Arbeit dargestellt.
Anschließend werden in Kapitel 4 die Anpassungen und Erweiterungen der Toolbox vorgestellt, die sich aus den Anforderungen der Verfahren ergeben.
In Kapitel 5 geht es dann ausführlich um die *Empirical Mode Decomposition*. Dazu wird zunächst notwendiges Hintergrundwissen erarbeitet. Ein besonderes Augenmerk liegt dabei auf den verschiedenen Möglichkeiten zur Anwendung der EMD bei der Untersuchung mehrdimensionaler Daten. Nachdem mit der *Multidimensional Ensemble Empirical Mode Decomposition* (MEEMD) eine erste Implementierung

eines solchen Verfahrens vorgestellt und hinsichtlich der Anwendbarkeit untersucht wurde (Abschnitt 5.3), wird daraus motiviert in Abschnitt 5.4 eine Methode zur Beschleunigung des Verfahrens vorgeschlagen. Anschließend wird in Abschnitt 5.5 die Implementierung der *Fast and Adaptive Multivariate and Multidimensional Empirical Mode Decomposition* (FA-MVEMD) als ein zweites Verfahren vorgestellt und im Anwendungskontext bewertet. Das Kapitel wird mit einem Vergleich der beiden Ansätze abgeschlossen.
Daran anschließend wird in Kapitel 6 die *Clauser Chart Methode* erarbeitet und die Implementierung vorgestellt. Auch hier folgt der Aufbau wiederum dem Schema Erarbeitung des Hintergrundwissens - Implementierung - Ergebnisanalyse.
Abschließend werden in Kapitel 7 die Ergebnisse der Arbeit zusammengefasst und ein Ausblick über mögliche anschließende Untersuchungen und Vorhaben im Kontext der Toolbox, sowie im Speziellen der betrachteten Methoden, gegeben.

2 Anwendungskontext

Nachfolgend werden zunächst wichtige Begriffe aus dem Anwendungskontext erklärt. Darauffolgend werden die Messverfahren zum besseren Verständnis der Arbeit kurz dargestellt, mit denen die Daten ermittelt werden, die in der Toolbox analysiert werden sollen. Anschließend werden die verwendete Programmiersprache und die relevanten Dateiformate präsentiert.

2.1 Begriffe aus dem Bereich der Strömungsmechanik

In der Abteilung HGK - und damit von den Nutzern der Toolbox - werden insbesondere Geschwindigkeitsfelder von Grenzschichten betrachtet. Auch Geschwindigkeitsprofile spielen bei der Analyse eine große Rolle.

Grenzschicht Wird ein Objekt von einem Fluid umströmt, bildet sich an der Oberfläche des Objektes eine so genannte Grenzschicht aus, in der die ungestörte Außengeschwindigkeit in Richtung des Objektes typischerweise abfällt und schließlich direkt an der Wand gleich Null ist. Die Grenzschicht ist als der Teil einer Strömung definiert, der durch die Viskosität (Zähigkeit) des Fluides beeinflusst wird [36, S. 24].

Geschwindigkeitsfeld Unter einem Geschwindigkeitsfeld versteht man eine Momentaufnahme eines Teiles der Strömung, für die an mehreren Punkten im Raum die Strömungsgeschwindigkeit bekannt ist. Ein solches Geschwindigkeitsfeld kann sich über zwei oder drei Raumdimensionen erstrecken, je nach Messverfahren. Die Geschwindigkeit wird meist mit den drei Komponenten u, v und w dargestellt. Jede dieser Komponenten entspricht der Geschwindigkeit in die entsprechende Raumrichtung: u ist die Geschwindigkeit in Strömungsrichtung (X), v die Geschwindigkeit in wandnormaler Richtung (Y) und w entsprechend quer zur Strömung (Z-Richtung). Aus einem Geschwindigkeitsfeld oder einer Reihe von Geschwindigkeitsfeldern lassen sich auch andere Größen wie Beschleunigungen ableiten.

Geschwindigkeitsprofil Aus einem Geschwindigkeitsfeld lässt sich ebenfalls ein sogenanntes Geschwindigkeitsprofil bestimmen. Dieses erhält man, wenn man die Geschwindigkeit bei gleichem Abstand zur Oberfläche des untersuchten Objektes

mittelt. Das Ergebnis wird in Abhängigkeit von der Entfernung zur Oberfläche des Objektes aufgetragen.

2.2 Verfahren der Strömungsmessung

Zu den besonderen Herausforderungen bei der experimentellen Strömungsuntersuchung zählt, dass die Strömung durch die Messung nicht beeinflusst werden darf. Gleichzeitig möchte man eine möglichst hohe räumliche Auflösung der Messpunkte erzielen, an denen die Strömungseigenschaften bekannt sind. Um diese konträren Anforderungen erfüllen zu können, wurden optische Messverfahren entwickelt, zu denen auch *Particle Image Velocimetry* (PIV) und *Shake-The-Box* (STB) gehören. Beide arbeiten damit, dass in die Strömung kleine Partikel eingebracht werden. Durch die Belichtung eines Messbereichs mit einem Laser werden die Partikel sichtbar gemacht und das an ihnen gestreute Licht wird mit Kameras aufgenommen. Anschließend können mittels mathematischer Auswerteverfahren Partikelbewegungen bestimmt und daraus Geschwindigkeiten berechnet werden.

2.2.1 Particle Image Velocimetry (PIV)

Bei der *Particle Image Velocimetry* (PIV) wird das Messfeld zweimal kurz nacheinander belichtet und fotografiert. Die Geschwindigkeitsfelder werden anschließend aus dem Versatz der Partikel zwischen den beiden Pulsen berechnet. Dazu wird angenommen, dass sich benachbarte Partikel (annähernd) homogen bewegen. Mithilfe von statistischen Methoden (zum Beispiel einer Kreuz-Korrelation) wird für Teilbereiche der Bilder ein durchschnittlicher Verschiebungsvektor ermittelt. Diese Verschiebungsvektoren können dann unter Berücksichtigung der Größe des Messvolumens in ein Geschwindigkeitsfeld umgerechnet werden. Bei einer klassischen PIV-Messung wird nur ein „Schnitt“ durch die Messstrecke mit Hilfe einer Kamera betrachtet (siehe Abbildung 2.1). Das resultierende Geschwindigkeitsfeld ist daher zweidimensional, und auch die Teilchenbewegung ist nur innerhalb des Schnittes bekannt. Es werden also nur zwei Geschwindigkeitskomponenten erfasst. Um die Geschwindigkeit in die dritte Raumrichtung berechnen zu können, wurde *Stereoskopisches-PIV* entwickelt. Dazu werden mindestens zwei Kameras genutzt und die Verschiebungen innerhalb der Bilder beider Kameras des selben Zeitstempels miteinander verglichen. Durch die (wenn auch geringe) Tiefe des Messfeldes

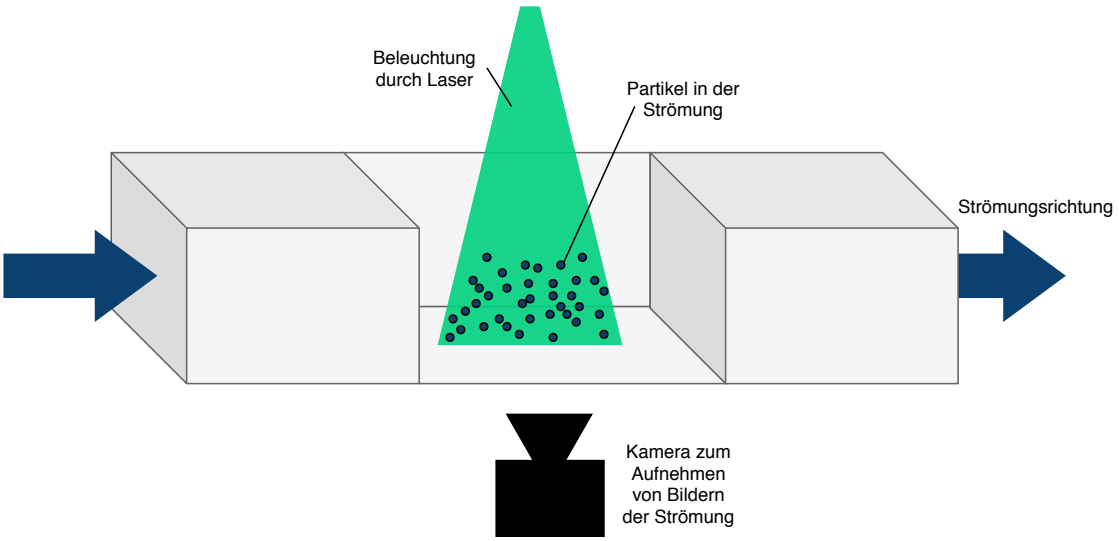


Abbildung 2.1: Schematische Darstellung einer PIV-Messung

können bei dem Vergleich Bewegungen in die dritte Raumrichtung festgestellt werden. Ausführliche Beschreibungen zum Messverfahren finden sich in *Einführung in die Strömungsmeßtechnik* [11, S. 181-194] und insbesondere in *Particle Image Velocimetry - A Practical Guide* [28].

2.2.2 Shake-The-Box (STB)

Im Gegensatz zu der statistischen Auswertung bei PIV, ermöglicht das im DLR entwickelte Messverfahren *Shake-The-Box* (STB, [35]) das Nachverfolgen einzelner Partikel („Tracking“) über eine Serie von Bildern. Mit dem Verfahren ist man nun außerdem in der Lage, ein Messvolumen statt nur eines Schnittes untersuchen zu können. In Abbildung 2.2 sind die einzelnen „Tracks“ der Partikel einer STB-Messung zu erkennen. Der grundsätzliche Messaufbau ist dem PIV-Aufbau ähnlich, allerdings werden mehrere Kamerasysteme eingesetzt. Ein Kamerasystem besteht aus mehreren Kameras, die auf Laserpulse abgestimmt Bilder aufnehmen. Dadurch kann die Zeit zwischen zwei Pulsen verringert werden - was insbesondere bei hohen Strömungsgeschwindigkeiten für den Algorithmus notwendig ist. In den aufgenommenen Bildern wird durch den STB-Algorithmus nach Partikeln gesucht, die in mehreren oder allen Bildern auftauchen, um deren Position zu bestimmen und daraus eine Partikelbahn zu berechnen. Ausgehend von zwei aufeinanderfolgenden Pulsen wird basierend auf einem Prediktor-Geschwindigkeitsfeld die mutmaßliche

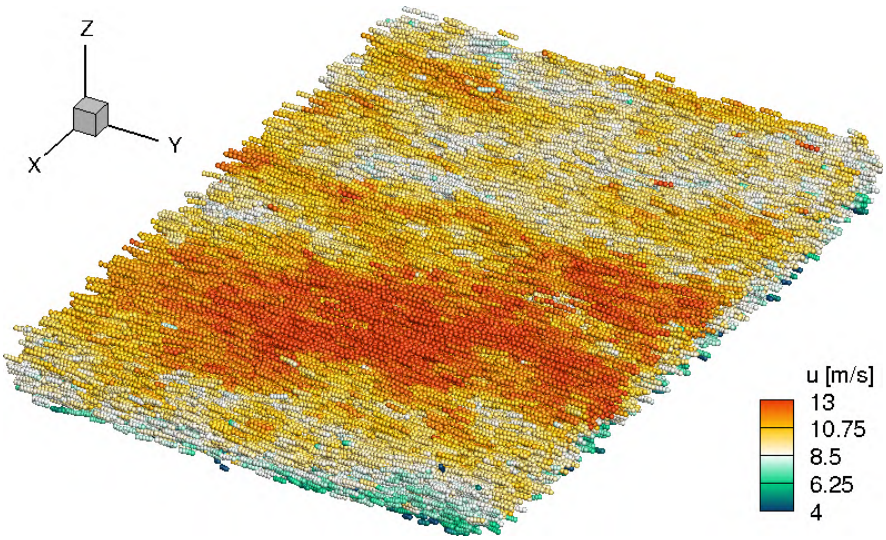


Abbildung 2.2: Ergebnis einer STB-Messung: Einfärbung nach Geschwindigkeit u in Strömungsrichtung X

Position eines Partikels von Puls zu Puls bestimmt. Werden an den prognostizierten Positionen hinreichend passende Partikel gefunden, wird diese Partikelbahn gespeichert. Zusätzlich werden die gefundenen Partikelbahnen aus den Ausgangsbildern entfernt, um dort die Partikeldichte zu verringern. Bei einer iterativen Anwendung des Verfahrens können durch die geringere Partikeldichte neue Partikelbahnen gefunden werden.

2.3 Programmiersprache und GUI

In dem Projekt werden primär die Programmiersprache C++ mit Qt [27] als Framework für die graphische Oberfläche eingesetzt. Die Programmiersprache C++ verspricht durch kompilierten Code sowie Hardware-Nähe eine hohe Verarbeitungsgeschwindigkeit. Außerdem gibt es eine Vielzahl wissenschaftlicher Bibliotheken, die C++-kompatibel sind. Bei Qt handelt es sich ebenfalls um eine C++ Bibliothek, die sich zum einen durch Plattformunabhängigkeit auszeichnet und zum anderen eine Vielzahl an Komponenten (Widgets) und Fenstern vordefiniert, wie beispielsweise Tabellendarstellungen, Dialoge oder Menüs. Außerdem können eigene Widgets eingebunden werden, sodass es eine große Erweiterungsmöglichkeit gibt.

2.4 Verwendete Dateiformate

Eines der am häufigsten verwendeten Dateiformate im Kontext dieser Arbeit ist das Tecplot-Dateiformat. Tecplot ist ein wissenschaftliches Programm zum Darstellen und Aufbereiten von Messergebnissen. Es gibt drei Tecplot Dateiformate, ein ASCII-Dateiformat („.dat“), sowie zwei binäre („.plt“ und das neuere „.szplt“). Für das neuere binäre Dateiformat gibt es eine offizielle TecIO Bibliothek, die in C++ genutzt werden kann, sowie eine Dokumentation samt Beschreibung des ASCII-Dateiformats, dem Tecplot Data Format Guide [10]. Ein weiteres, gelegentlich verwendetes Dateiformat ist das Network Common Data Format (NetCDF, „.nc“, „.cdf“), welches explizit für den Austausch wissenschaftlicher Daten entwickelt wurde [31]. Auch hierfür gibt es eine Bibliothek mit C++ Schnittstelle.

3 Stand der Toolbox zu Beginn der Arbeit

Die Grundlage der Toolbox ist in einer vorhergenden Praxisphase bereits implementiert worden [37], wie beispielsweise Lese- und Schreibroutinen und auch eine erste Routine zur Verarbeitung von Messdaten. Daher soll ein kurzer Überblick über die Architektur des Programmes gegeben werden, das bereits implementierte Verfahren zur Überführung von Daten auf ein strukturiertes Gitter vorgestellt sowie die Möglichkeiten zur Darstellung der Ergebnisse betrachtet werden.

3.1 Architektur und Erweiterungspunkte

Das Programm ist software-architektonisch in vier Ebenen gegliedert, sodass Darstellung, Berechnung, Datenhaltung und Im-/Export so gut es geht voneinander getrennt sind (siehe Abbildung 3.1). Die Zugriffe zwischen den Ebenen finden jeweils über Schnittstellen statt.
Die drei nachfolgenden Abschnitte beschreiben Anwendungsfälle des Programms, an dem eine Erweiterung möglich sein sollte. Dies wurde bei der bisherigen Entwicklung des Programmes beachtet: Sowohl für Lese- und Schreibroutinen, wie auch Berechnungsverfahren und Darstellungs-Widgets gibt es eine abstrakte Schnittstelle. Diese sind in Abbildung 3.1 orange hervorgehoben.

3.1.1 Lese- und Schreibroutinen

Die Lese- und Schreibroutinen müssen zwischen der internen Organisation der Daten und dem jeweiligen Ein-/Ausgabeformat übersetzen. Demzufolge ist die Schnittstelle hier denkbar einfach gehalten: Es muss lediglich eine Funktion definiert werden, die einen Datensatz annimmt und an einen vorgegebenen Pfad schreibt, oder einen Pfad annimt und einen Datensatz zurückgibt. Die Lese- und Schreibroutinen sind für die benötigten Dateiformate Tecplot und NetCDF mithilfe der Bibliotheken bereits implementiert. Nachfolgend muss folglich nur noch mit der intern verwendeten Abstraktion der Daten gearbeitet werden.

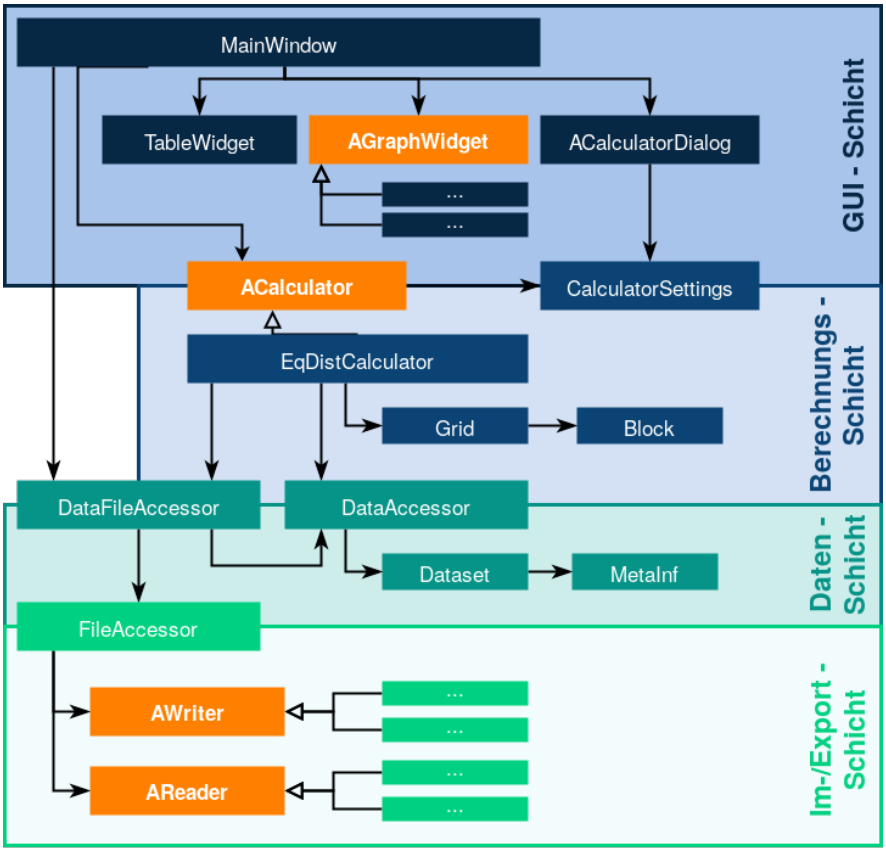


Abbildung 3.1: Interne Struktur des Programms: Ebenenorientierte Gliederung - die Erweiterungspunkte sind orange eingefärbt

3.1.2 Berechnungsroutinen

Jede Berechnung wird über die abstrakte Klasse des *ACalculator* beschrieben. In dem *ACalculator* sind bereits allgemeine Funktionen definiert, wie das Setzen allgemeiner Einstellungen für eine Berechnung (welcher Datensatz genutzt werden soll, wo das Ergebnis abgespeichert werden soll, etc.). Jede Berechnungsroutine muss daher nur noch die eigentliche Berechnung beschreiben. Die Schnittstelle des *ACalculator* fordert von der Funktion als Rückgabewert eine Liste an Pfaden, an denen die Ergebnisse der Berechnung abgelegt wurden. Die abgespeicherten Ergebnisse enthalten alle zur Interpretation notwendigen Informationen.

3.1.3 Graphik-Widgets

Alle Widgets, die zur Darstellung der Daten dienen, müssen ein vollständiges Qt-Widget sein, um eingebunden werden zu können. Außerdem sollten sie über eine Funktion verfügen, um den aktuell angezeigten Datensatz auszutauschen. Da eine Tabellendarstellung darüber hinaus jedoch nur wenig mit möglichen graphischen Darstellungen gemeinsam hat, ist dies bereits die gesamte Beschreibung der Schnittstelle. Für ein graphisches Widget wurde jedoch bereits eine erweiterte Schnittstelle mit den gewünschten Funktionen definiert, die „per Knopfdruck“ von der GUI aus gestartet werden sollen.

3.2 Daten-Überführung auf ein strukturiertes Gitter

Einige Analyseverfahren können nur angewandt werden, wenn die Daten auf einem strukturierten, äquidistanten Gitter vorliegen. Unter einem strukturierten Gitter ist hierbei insbesondere zu verstehen, dass ein Datenpunkt entlang jeder Raumdimension einen Vorgänger und einen Nachfolger hat - mit Ausnahme der Randpunkte des Datensatzes. Bei einer Messung, bei der die Datenpunkte in zwei Raumdimensionen auftreten, bilden also jeweils vier Messpunkte ein Rechteck, bei drei Dimensionen acht einen Quader. Die Ergebnisse einer STB-Messung erfüllen genau das nicht. Die einzelnen Datenpunkte liegen beliebig im Raum verteilt, es gibt keine Struktur in der sie angeordnet sind. Ein erstes, bereits implementiertes Berechnungsverfahren ist daher eine Methode zur Überführung von STB-Messwerten auf ein strukturiertes Gitter. Abbildung 3.2 zeigt das Prinzip dieser Methode.

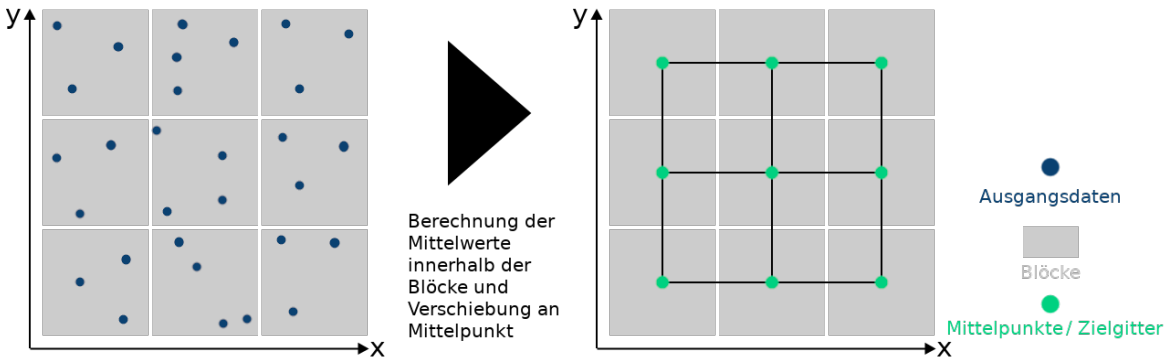
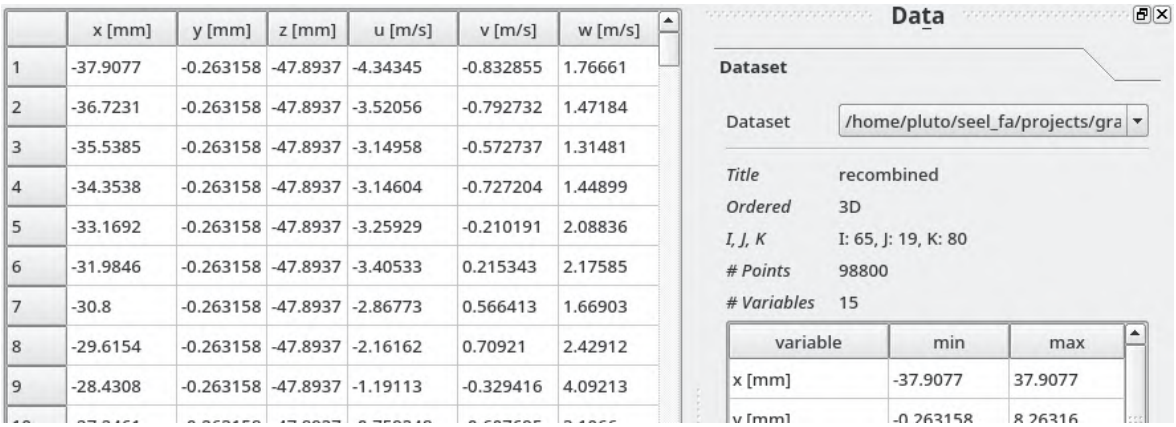


Abbildung 3.2: Prinzip der Überführung auf ein strukturiertes Gitter

Dazu wird der Datensatz in Blöcke unterteilt, innerhalb derer alle enthaltenen Datenpunkte gemittelt werden. Gleichzeitig wird dadurch ein Zielgitter für die Berechnung definiert, das aus den Mittelpunkten der Blöcke besteht. Nebenbei ergibt sich ein weiterer Anwendungsfall für diese Routine: Die Berechnung eines Geschwindigkeitsprofils. Liegt ein dreidimensionales Geschwindigkeitsfeld vor, bei dem die wandnormale Achse die Y-Achse ist, kann die Berechnungsmethode wie nachfolgend beschrieben angewandt werden, um ein Geschwindigkeitsprofil zu bestimmen: Bei der Definition des Zielgitters wird die Anzahl der gewünschten Blöcke für X und Z auf eins gesetzt. Das heißt, der Mittelwert wird spannweitig und in Strömungsrichtung über das gesamte Messvolumen bestimmt. Für die Y-Achse, also in wandnormaler Richtung, kann die gewünschte Anzahl an Datenpunkten angegeben werden. Da die Berechnungsroutine es zulässt, dass auch über mehrere Dateien und somit Datensätze gemittelt wird, können so auch statistisch aussagekräftige Daten aus mehreren (Teil-)Messungen gewonnen werden.

3.3 Möglichkeiten zur Darstellung der Ergebnisse

Für die Benutzbarkeit des Programms ist es wichtig, dass die Ergebnisse so dargestellt werden, dass schnell kontrolliert werden kann, ob mit der jeweiligen Berechnung das gewünschte Ergebnis erzielt wurde. Hierfür ist noch keine zufriedenstellende Implementierung vorhanden. So kann im Wesentlichen nur auf eine Tabellendarstellung der Daten zurückgegriffen werden. Zusätzlich werden auch die Meta-Informationen über den Datensatz dargestellt (vgl. Abbildung 3.3).



	x [mm]	y [mm]	z [mm]	u [m/s]	v [m/s]	w [m/s]
1	-37.9077	-0.263158	-47.8937	-4.34345	-0.832855	1.76661
2	-36.7231	-0.263158	-47.8937	-3.52056	-0.792732	1.47184
3	-35.5385	-0.263158	-47.8937	-3.14958	-0.572737	1.31481
4	-34.3538	-0.263158	-47.8937	-3.14604	-0.727204	1.44899
5	-33.1692	-0.263158	-47.8937	-3.25929	-0.210191	2.08836
6	-31.9846	-0.263158	-47.8937	-3.40533	0.215343	2.17585
7	-30.8	-0.263158	-47.8937	-2.86773	0.566413	1.66903
8	-29.6154	-0.263158	-47.8937	-2.16162	0.70921	2.42912
9	-28.4308	-0.263158	-47.8937	-1.19113	-0.329416	4.09213

Abbildung 3.3: GUI der Toolbox (Ausschnitt): Tabellenansicht der Daten

Für die Überführung auf ein strukturiertes Gitter reicht diese Darstellung aus, da die wichtigste Information ist, ob das resultierende Gitter die gewünschte Anzahl Punkte entlang der jeweiligen Dimensionen hat (was den Meta-Informationen *I, J, K* zu entnehmen ist). Hingegen kann nicht auf einen Blick beurteilt werden, ob die ermittelten Mittelwerte realistisch sind. Hierzu wäre eine graphische Darstellung (2D, 3D) der Daten mit Einfärbung gemäß einer weiteren Größe, beispielsweise einer Geschwindigkeitskomponente, wünschenswert.

4 Notwendige Anpassungen der Toolbox

Mit der bereits existierenden Toolbox ist eine gute Ausgangslage vorhanden, um die im Rahmen dieser Arbeit umzusetzenden Verfahren zu implementieren. Nichtsdestotrotz müssen einige Anpassungen und Verbesserungen vorgenommen werden, um eine gute Benutzbarkeit zu ermöglichen. Dazu wird zunächst betrachtet, welche neuen Anforderungen durch die zu implementierenden Verfahren hinzukommen. Für die EMD ist es für die Bewertung der physikalischen Plausibilität wichtig, dass noch in der Toolbox eine graphische Betrachtung der Ergebnisse möglich ist. Daher sollte ein Weg gefunden werden, um zwei- und dreidimensionale Geschwindigkeitsfelder darstellen zu können. Für das Clauser Chart Verfahren ist es hingegen notwendig, in simplen X/Y-Graphen Funktionen und einzelne Datenpunkte plotten zu können. Die Implementierung der entsprechenden Oberflächen wird nachfolgend vorgestellt (Abschnitt 4.1). Außerdem wird die Implementierug einer Berechnungs-Queue dargelegt, welche die eigentlichen Berechnungen in Hintergrund-Threads verschiebt (Abschnitt 4.2).

4.1 Graphische Darstellung der Daten

Wie bereits erwähnt, sollen Daten auf verschiedene Arten graphisch dargestellt werden. Es wird daher eine Bibliothek gesucht, die das Darstellen von zwei- und dreidimensionalen Geschwindigkeitsfeldern ermöglicht, sowie eine Bibliothek zum Plotten von Funktionen.

4.1.1 Darstellung von Geschwindigkeitsfeldern – QwtPlot3D

Bei der Darstellung von Geschwindigkeitsfeldern sind besondere Randbedingungen und Anforderungen zu beachten. Diese wurden in [37] bereits herausgearbeitet und Bibliotheken hinsichtlich der Einsetzbarkeit untersucht. Aus verschiedenen Gründen konnte jedoch keine erfolgreiche Implementierung durchgeführt werden. Daher werden nachfolgend noch einmal die wichtigsten Anforderungen aufgelistet:

- Unterstützung der Darstellung dreidimensionaler Daten mit frei beweglichem Betrachtungspunkt

- Möglichkeit zur Einfärbung der Datenpunkte
- Integration in die Qt-Oberfläche

Der Hauptgrund, warum die Bibliothek QwtPlot3D [5] nicht genutzt werden konnte, war die fehlende Kompatibilität mit Qt5. Eine Analyse der im Programm verwendeten GUI-Komponenten zeigt jedoch, dass die Toolbox ausschließlich Komponenten nutzt, die bereits in Qt4.8 enthalten sind. Einzige Ausnahme war die QtDataVisualization Erweiterung, welche ursprünglich für die Darstellung der Geschwindigkeitsfelder angedacht war, bei der eine Einfärbung der Daten jedoch nicht wunschgemäß möglich ist. Mit der Umstellung auf Qt4.8 ist es gelungen, die Einbindung von QwtPlot3D vorzunehmen, wie in Abbildung 4.1 zu sehen ist. Dabei stellt die Bibliothek ein Widget bereit, welches programmatisch in ein Qt-Layout eingefügt werden kann. Anschließend können sowohl Daten hinzugefügt, wie auch aktualisiert werden. Die Bibliothek stellt außerdem bereits eine interaktive Oberfläche bereit, sodass die Ansicht mit der Maus gedreht und bewegt werden kann. Alle im linken Teil von Abbildung 4.1 zu sehenden Funktionen repräsentieren die von der Schnittstelle *AGraphWidget* geforderten Funktionen und werden an das Widget weitergegeben.

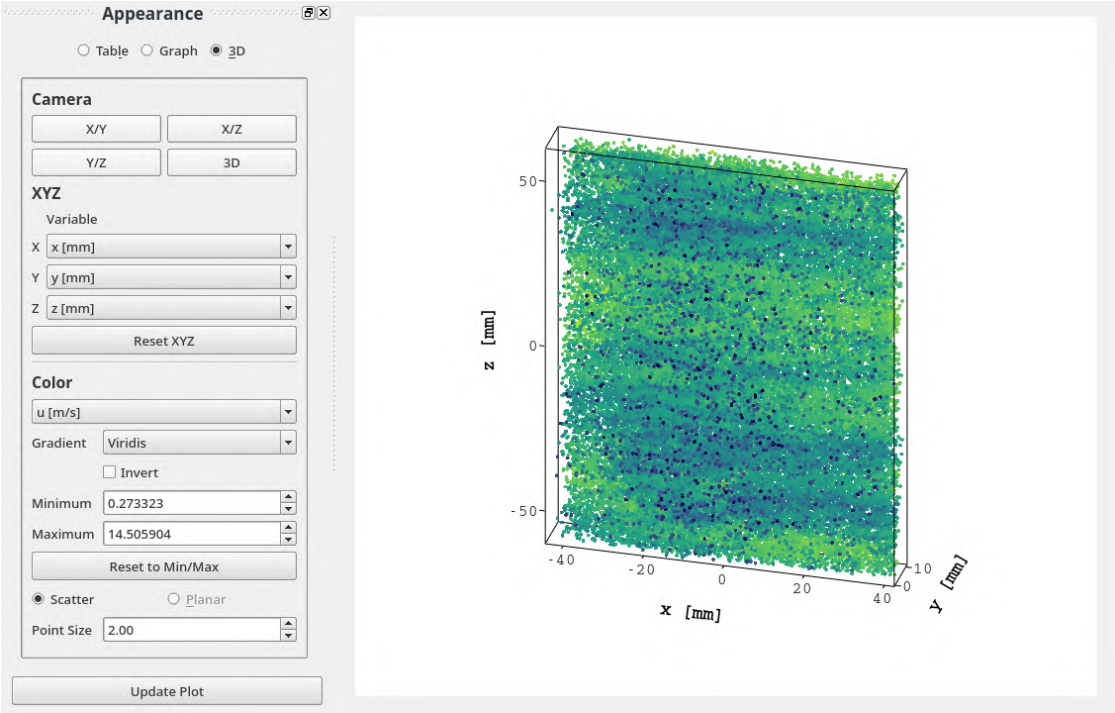


Abbildung 4.1: Geschwindigkeitsfeld, mit QwtPlot3D in der Toolbox dargestellt

4.1.2 Funktionen - QCustomPlot

Für das zweite Plotting-Widget gibt es weniger Anforderungen. So müssen vor allem einzelne Datenpunkte in einer X/Y-Darstellung präsentiert werden können. Außerdem sollte auch die Möglichkeit vorhanden sein, einen Funktionsverlauf darzustellen. Hinzu kommt als wichtige Anforderung noch eine logarithmische Achsendarstellung. Mit der QCustomPlot-Bibliothek [12] können alle diese Anforderungen erfüllt werden. Abbildung 4.2 zeigt dies exemplarisch: Es wird ein Geschwindigkeitsprofil von Messdaten dargestellt. Zusätzlich sind eine lineare und eine logarithmische Funktion eingezeichnet.

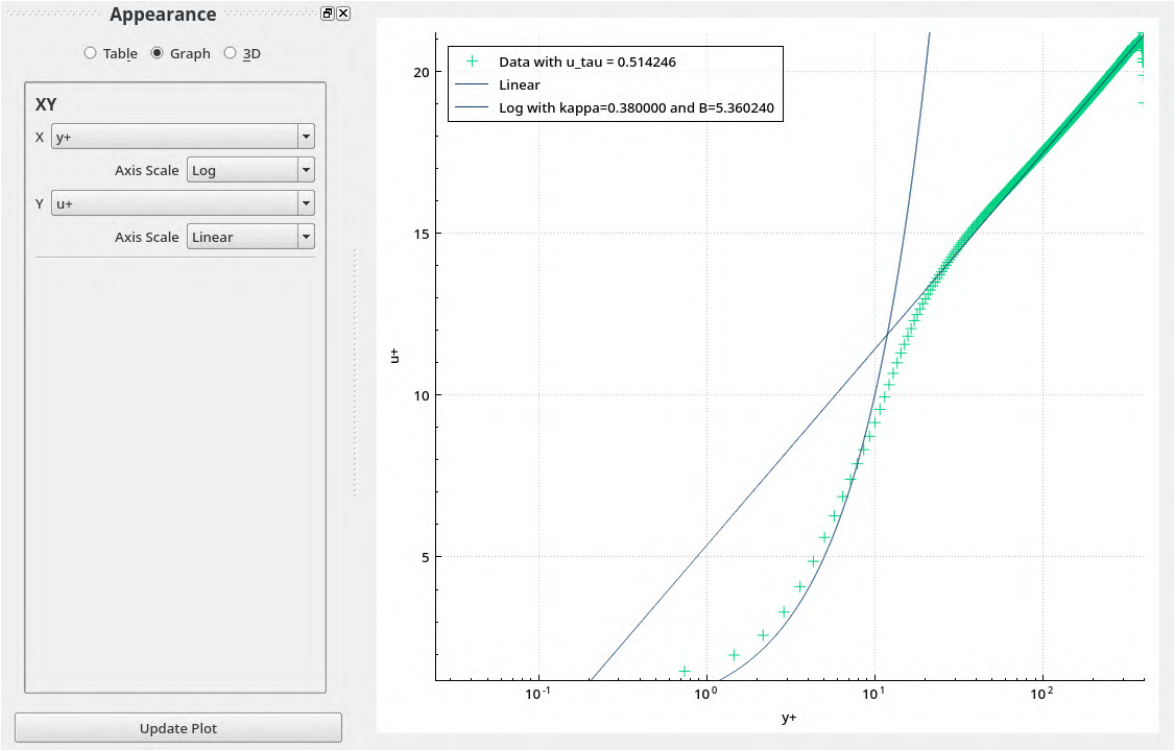


Abbildung 4.2: Geschwindigkeitsprofil, mit QCustomPlot in der Toolbox dargestellt

4.2 Multi-Threading und Berechnungs-Queue

In der zu Beginn vorhandenen Version der Toolbox war das Programm nach Start einer Berechnung solange nicht weiter benutzbar, bis die Berechnung abgeschlossen war. Unterdessen wurde dem Nutzer lediglich ein Dialog mit einem Fortschrittsbalken angezeigt. Verschiedene Gründe sprechen letztlich für die Erneuerung des

Konzeptes an dieser Stelle: Durch die Implementierung der Graphik-Widgets ergibt sich eine sinnvolle Möglichkeit, während der Durchführung einer Berechnung Ergebnisse vorhergehender Berechnungen anzuschauen. Hinzu kommt, dass es durch den hohen Berechnungsaufwand und die damit benötigten Zeiten wünschenswert wäre, mehrere Berechnungen abends starten zu können, damit die Ergebnisse am nächsten Morgen zur Verfügung stehen. Auch das Abbrechen von Berechnungen sollte möglich sein. Zur Implementierung dieser Anforderungen sind zwei Schritte notwendig: Die Verschiebung der Berechnungen in Hintergrundthreads und die Definition einer Queue, in der die zu startenden Berechnungen zwischengespeichert werden können.

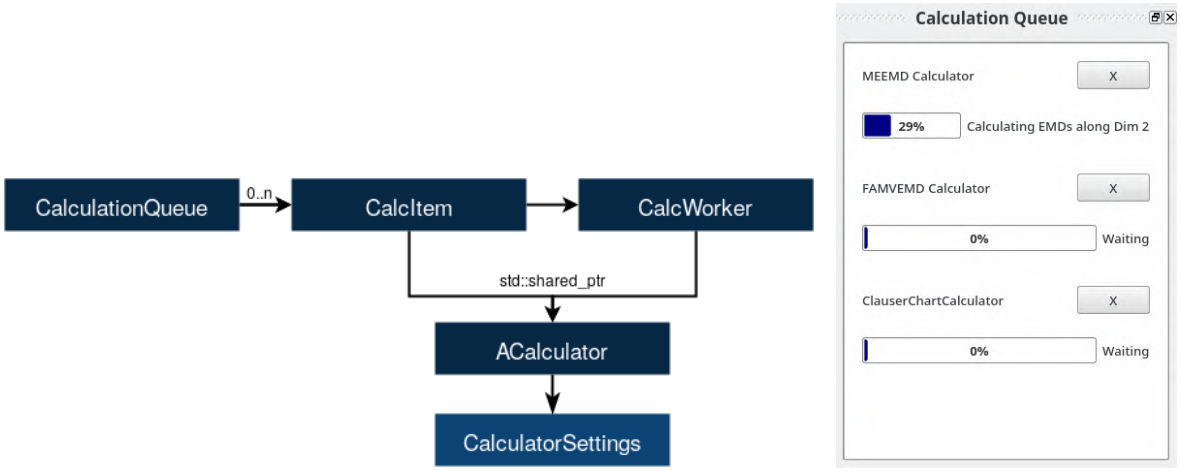
4.2.1 Multi-Threading mit C++ und Qt

In der C++-Standardbibliothek sind Funktionen zur Nutzung von Threads enthalten. So kann mittels eines einfachen Interfaces eine Funktion in einem gesonderten Thread gestartet werden. Beachtet werden muss jedoch, dass ein Thread nicht problemlos von außen beendet werden kann, wenn die auszuführende Funktion noch nicht beendet ist. Der Grund hierfür ist, dass in einem solchen Fall der von dem Thread genutzte Arbeitsspeicher in einem undefinierten Zustand hinterlassen werden würde. Außerdem ist zu beachten, dass die graphische Oberfläche einer Qt-Anwendung nur aus dem Thread aktualisiert werden kann, der diese steuert - und nicht aus einem Hintergrundthread. Dieses Wissen ist wichtig für die Anzeige des Fortschritts der Berechnung. Für beide Fälle (Abbruch einer Berechnung, Anzeige des Fortschritts) muss also ein Mittel zur Kommunikation zwischen den Threads gefunden werden. Mit dem Signal- und Slotsystem von Qt ist dafür eine Möglichkeit vorhanden: Dieses stellt eine Anwendung des Observer-Patterns dar, ermöglicht also, dass ein Objekt auf die Veränderung des internen Zustandes eines zweiten Objektes reagiert. Das Signal-/Slotsystem funktioniert auch thread-übergreifend.

4.2.2 Implementierung der Berechnungs-Queue

Die Implementierung der Berechnungs-Queue wird durch die bereits vorhandene software-architektonische Entkopplung der Berechnungsroutinen begünstigt: Soll eine Berechnung gestartet werden, wird dem Nutzer zunächst ein Dialog angezeigt, um die für die Berechnung gewünschten Einstellungen zu treffen. Anschließend

werden diese an die Berechnungsroutine weitergegeben. Diese muss danach noch explizit gestartet werden. Durch diesen Aufbau, sowie die Abstraktion aller Berechnungsroutinen in eine einheitliche Schnittstelle, ist es möglich, eine Liste der zu startenden Berechnungen zu erstellen. Abbildung 4.3a zeigt den Aufbau der Berechnungs-Queue schematisch als vereinfachtes Klassendiagramm. In die *CalculationQueue* kann eine beliebige Anzahl an *CalcItems* eingefügt werden. Das *CalcItem* kann auf den *CalcWorker* sowie den eigentlichen *ACalculator* zugreifen. Der Verweis auf den *ACalculator* ist dabei durch einen *Shared Pointer* realisiert (auch in der C++-Standardbibliothek enthalten), sodass auch der *CalcWorker* auf das selbe Objekt zugreifen kann.



(a) **(b)**

Abbildung 4.3: Umsetzung der Berechnungs-Queue
(a) Vereinfachtes Klassendiagramm
(b) Graphische Oberfläche

Das Funktionsprinzip ist nun wie folgt: Die *CalculationQueue* bestimmt, welches *CalcItem* als nächstes gestartet wird. Dieses startet dann intern einen Thread, in dem der *CalcWorker* die Berechnungsfunktion des *ACalculator* aufruft. Das Starten und Beenden des Threads obliegt dem *CalcItem*. Über verschiedene Signale wird nun die Kommunikation in beide Richtungen durchgeführt. So kann an den *ACalculator* das Signal gesendet werden, dass die Berechnung abgebrochen werden soll. Hier muss in den Berechnungsfunktionen die Reaktion auf dieses Signal implementiert sein, damit der Thread ordentlich beendet werden kann. Im Hauptthread des Programms wird jedoch nicht auf die erfolgreiche Beendigung des Berechnungsthreads gewar-

tet, da es bei einzelnen Berechnungen sehr lange dauern kann, bis der Abbruch erfolgreich umgesetzt ist. Stattdessen wird der Thread *„detached"*, also abgelöst. Dies stellt kein Problem dar, da im Falle eines Abbruchs keine Rückmeldung des Threads mehr benötigt wird. Von der Berechnungsroutine kann in die andere Richtung per Signal der Fortschritt kommuniziert werden, sodass dieser dem Nutzer präsentiert werden kann, wie in Abbildung 4.3b zu sehen ist. Außerdem kann eine textuelle Information über den aktuellen Stand der Berechnung, sowie die nachfolgenden Berechnungen angezeigt werden. Wird eine Berechnung abgebrochen oder ist fertig, wird das entsprechende Element einfach aus der Liste entfernt und das nächste rückt nach. Sobald ein Element das oberste in der Liste wird, wird für dieses die Berechnung gestartet. Dadurch können Berechnungen im Hintergrund (z.B. nachts) ausgeführt werden, ohne dass ein Kontrolleingriff notwendig ist.

5 Empirical Mode Decomposition

Bei der Empirical Mode Decomposition (EMD) handelt es sich um ein Verfahren, mit dem ein Signal in verschiedene Bestandteile zerlegt werden kann. Dabei werden die Funktionen, in die das Signal zerlegt wird, aus den Daten selbst entwickelt [15]. Sie müssen hierbei nicht bestimmte Funktionseigenschaften besitzen, sollen jedoch überwiegend nur eine einzige Frequenz enthalten. Damit kann die EMD als datengetriebene Analysemethode betrachtet werden.
In Abschnitt 5.1 wird zunächst die Theorie zur EMD erarbeitet, anschließend finden sich in Abschnitt 5.2 die aus dem Anwendungskontext resultierenden Anforderungen. Abschnitt 5.3 thematisiert die Implementierung einer Variante der EMD für mehrdimensionale Daten, der *Multidimensional Ensemble EMD* (MEEMD). Für diese wird darauffolgend in Abschnitt 5.4 die Entwicklung einer hinsichtlich der benötigten Rechenzeit verbesserten Variante dargestellt. Mit der *Fast and Adaptive Multivariate and Multidimensional EMD* (FA-MVEMD) wird in Abschnitt 5.5 eine zweite EMD-Implementierung vorgestellt, bevor in Abschnitt 5.6 ein abschließender Vergleich der beiden Verfahren vorgenommen wird.

5.1 Theoretische Grundlagen und Prinzip

Nachfolgend wird zunächst der grundlegende Algorithmus vorgestellt (Abschnitt 5.1.1). Wie in Abbildung 5.1 dargestellt, existiert eine Vielzahl an Variationen der EMD. Die Abbildung zeigt dabei nur die für diese Arbeit relevanten. Diese werden anschließend kurz vorgestellt. Zusätzlich ist in der Abbildung dargestellt, welche EMD-Varianten einander beeinflusst haben. Die Modifikationen des grundlegenden Verfahrens ohne Ausweitung auf mehrere Variablen oder Dimensionen (blau) finden sich in Abschnitt 5.1.2. Die multivariaten EMD-Varianten (orange) werden in Abschnitt 5.1.3 dargestellt und durch die Erweiterungen für die mehrdimensionale Anwendung in Abschnitt 5.1.4. Die FA-MVEMD kombiniert Verfahren zur multivariaten und multidimensionalen EMD und wird in Abschnitt 5.1.5 vorgestellt.

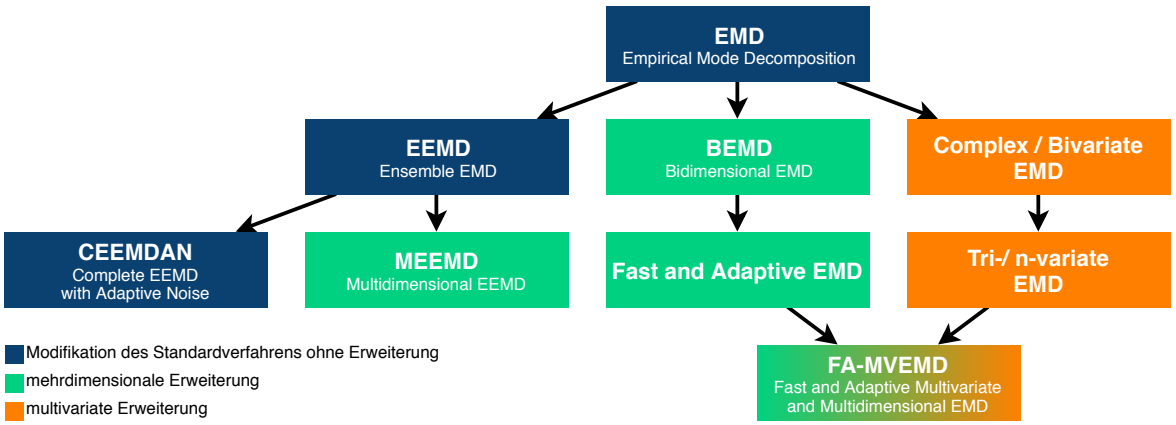


Abbildung 5.1: Überblick über die EMD-Varianten und Erweiterungen

5.1.1 Grundlegender Algorithmus

In Abbildung 5.2 ist das Ergebnis einer EMD eines Signals dargestellt, anhand dessen man gut die Grundidee nachvollziehen kann. Das Signal ist die Überlagerung, also Aufsummierung, zweier Dreiecks- und einer Sinussfunktion. Werden Funktionen addiert, „schwingt“ im Resultat die höherfrequente Ausgangsfunktion um die andere. Zur umgekehrten Zerlegung wird daher angenommen, dass die lokalen Minima und Maxima eines Signals zum höchstfrequenten Teil eines Signals gehören [15, S.917f]. Der lokale Trend, um den dieser hochfrequente Teil dann schwingt, beschreibt gewissermaßen ein lokales Mittel. Die Aufteilung in die beiden Kompo-

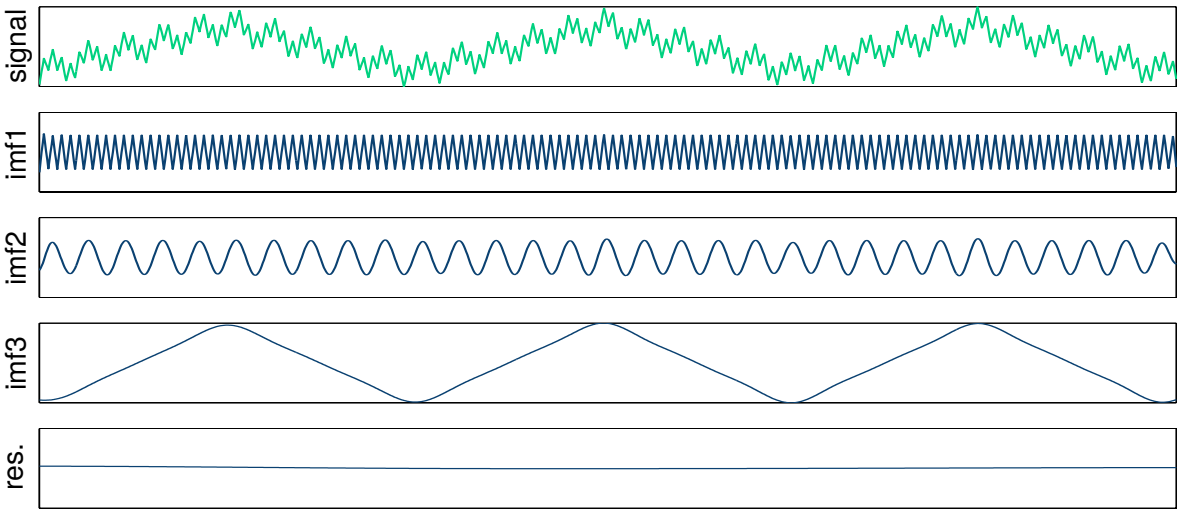
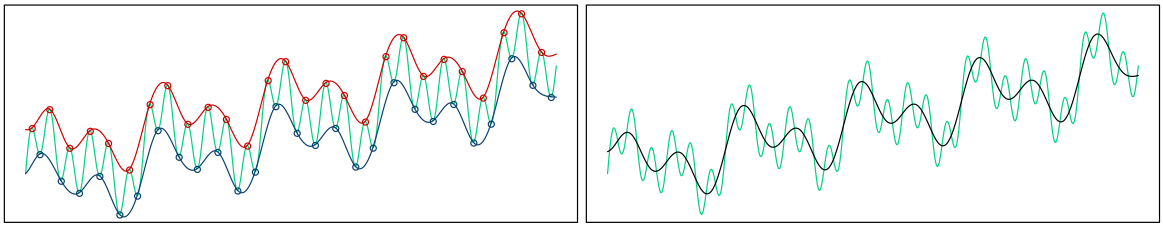
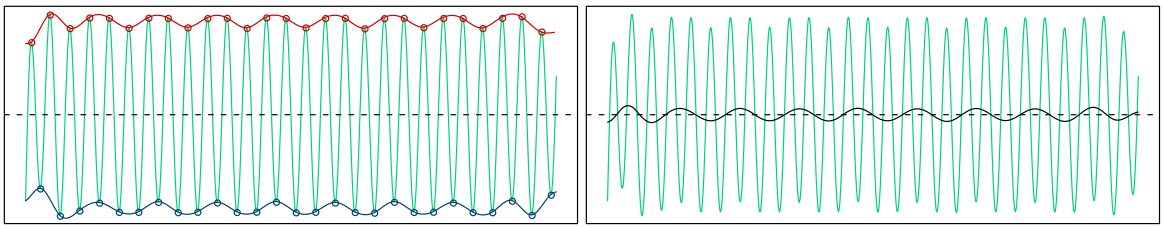


Abbildung 5.2: EMD: Ausgangssignal und Intrinsic Mode Functions (IMFs) sowie Residuum (modifiziert, aus [34])

nenten geschieht bei der EMD durch das sogenannte *Sifting*: Zunächst wird durch die lokalen Maxima und Minima je eine Interpolante bestimmt, üblicherweise mit kubischen Splines (wie auch in Huang; Wu [17]). Diese beiden Funktionen beschreiben zusammen eine „Einhüllende“ des Signals (Abbildung 5.3a). Mittig zwischen den Interpolanten befindet sich gewissermaßen ein „lokales Mittel“ des Signals. Zieht man dieses vom Ausgangssignal ab, erhält man den hochfrequenten Teil. Die vorhergehenden Schritte (Finden der Maxima, Bestimmung der Einhüllenden, Bestimmung der lokalen Mittelwertfunktion, Aufteilung des Signals) werden so oft wiederholt („Sifting“), bis ein Abbruchkriterium erreicht ist und somit eine sogenannte *Intrinsic Mode Function* (IMF) bestimmt wurde (Abbildung 5.3b).



(a) Bestimmung der Einhüllenden und gemittelten Funktion



(b) Extrahierte IMF und Sifting

Abbildung 5.3: EMD: Schritte bei der Dekomposition (modifiziert, aus [21])

Die IMF muss vor allem zwei Kriterien genügen [15, S. 915]:

- Die Anzahl lokaler Extrempunkte und der Nulldurchgänge darf sich um maximal Eins unterscheiden
- Die Funktion soll über den gesamten Bereich im Mittel annähernd den Wert Null annehmen.

Zur (annähernden) Erfüllung der Kriterien wurden verschiedene Abbruchkriterien vorgeschlagen. Ein relativ simpler Ansatz ist das schlichte Definieren einer maximalen Anzahl an Sifting-Iterationen [48]. Ein weiteres Konzept ist das S-Nummer-Kriterium [14]: Der Sifting-Prozess wird abgebrochen, wenn die Anzahl an Nulldurchgängen und Extrempunkten sich um maximal 1 unterscheidet, und deren Anzahl

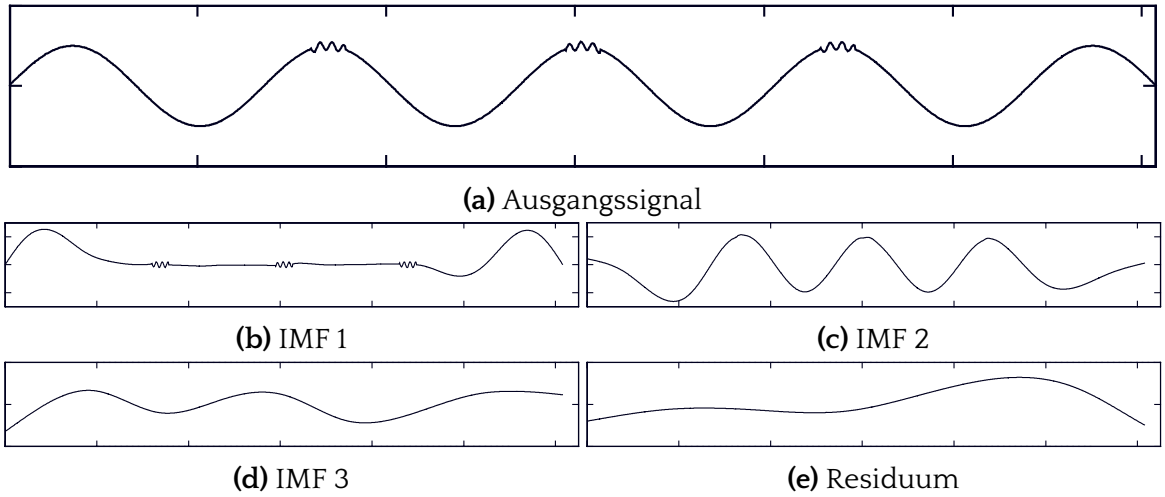
für S Iterationen gleichbleibt. Der optimale Wert für S befindet sich zwischen 3 und 8 [16]. Diese beiden Kriterien, S-Nummer-Kriterium und maximale Anzahl an Sifting-Iterationen, können auch in Kombination zur Anwendung kommen. Eine andere Möglichkeit besteht in der Definition eines Cauchy-ähnlichen Kriteriums, was in der ursprünglichen Veröffentlichung der EMD angedacht war [15]. Hierbei wird der Sifting-Prozess abgebrochen, wenn das lokale Mittel der extrahierten IMF eine hinreichend geringe Abweichung von Null vorweist (dazu wird ein Toleranzwert definiert). Das Problem dieser Methode ist, dass sie dazu tendiert, keine variierenden Amplituden in den IMFs zu zulassen.
Unabhängig davon, welches Kriterium genutzt wird, können anschließend iterativ weitere IMFs bestimmt werden, die zunehmend größere Frequenzen darstellen. Dazu wird das Signal abzüglich des hochfrequenten Anteils als neues Ausgangssignal betrachtet und der auch in Algorithmus 1 beschriebene Prozess erneut angewandt. Die Bestimmung von IMFs kann entweder mit demselben Kriterium abgebrochen werden, oder bei Erreichen einer festgesetzten Maximal-Anzahl an IMFs.

Algorithmus 1 Grundlegender Algorithmus zur Extraktion einer IMF

1: Suche alle Extrempunkte des Signals $h(k)$.
2: Interpoliere zwischen allen Minima (respektive Maxima) um die untere (obere) Einhüllende des Signals zu erhalten $(e_{min}(k), e_{max}(k))$.
3: Berechne die Funktion des lokalen Mittels als Durchschnitt der oberen und unteren Einhüllenden $m(k) = [e_{min}(k) + e_{max}(k)]/2$.
4: Ziehe die Funktion des lokalen Mittels vom Ausgangssignal ab, um die „oszillierende Mode“ zu erhalten $s(k) = h(k) - m(k)$.
5: Wenn $s(k)$ das Abbruchkriterium erfüllt, betrachte $s(k)$ als IMF. Ansonsten setze $h(k) = s(k)$ und wiederhole den Prozess ab Schritt 1.

Problem des Mode Mixing Bei der Anwendung des Standard-EMD Algorithmus tritt das so genannte *Mode Mixing* auf. Unter *Mode Mixing* versteht man, dass entweder in einer einzelnen IMF Signalbestandteile sehr unterschiedlicher Skalen (also Größenordnungen) enthalten sind, oder dass Signalbestandteile vergleichbarer Skalen auf unterschiedliche IMFs aufgeteilt sind. In Abbildung 5.4 ist dies exemplarisch dargestellt: Das Ausgangssignal enthält ein leichtes Rauschen, was dazu führt dass die Sinusfunktion in die IMF 1 und IMF 2 aufgeteilt ist. Der Ursprung dafür sind meistens Signalstörungen [48]. Das Problem ist, dass *Mode Mixing* die Aussagekraft

der einzelnen IMFs mindert [15], und ein Vergleich unterschiedlich großer Skalen schwierig wird.



(a) Ausgangssignal

(b) IMF 1 **(c)** IMF 2

(d) IMF 3 **(e)** Residuum

Abbildung 5.4: Auftreten von Mode Mixing (modifiziert, aus [48])

5.1.2 Ensemble EMD (EEMD)

Mit der *Ensemble EMD* (EEMD, Wu; Huang [48]) wird das Problem des *Mode Mixing* weitestgehend gelöst: Durch das Aufaddieren von weißem Rauschen mit begrenzter Amplitude vor der Analyse werden Signalstörungen „überdeckt". Zusätzlich wird durch das weiße Rauschen eine einheitliche Referenzskala geschaffen. Wird dieses Verfahren nun wiederholt ausgeführt und die resultierenden IMFs gemittelt, so löscht sich der Fehler, der durch das Hinzufügen des Rauschens entsteht, weitestgehend heraus. Die Anzahl an Wiederholungen wird Ensemble-Größe genannt. Laut Wu; Huang [48] reicht bereits eine Ensemble-Größe im unteren dreistelligen Bereich, um in den Ergebnissen weniger als 1 % Fehler zu enthalten, der durch das Rauschen hinzugekommen ist. Dies gilt, solange die Amplitude des hinzugefügten Rauschens (die zweite wichtige Größe bei diesem Prozess), geringer als die Standard-Abweichung der Original-Daten ist. Für die Rauschamplitude wird ein Wert von 20 % der Standard-Abweichung der Daten empfohlen.

Complete EEMD with Adaptive Noise (CEEMDAN) Bei der EEMD kann das Ausgangssignal aus den extrahierten IMFs nicht vollständig wiederhergestellt werden [44]. Im ursprünglichen EMD-Algorithmus ist die korrekte Wiederherstellung dadurch

sichergestellt, dass nach dem Extrahieren einer IMF das Residuum als Ausgangssignal genutzt wird. Bei der EEMD wird das Signal mehrmals mit unterschiedlichem aufaddiertem Rauschen getrennt voneinander analysiert und anschließend gemittelt. Dadurch gibt es auch mehrere Residuen und eine korrekte Weiterverarbeitung kann nicht mehr garantiert werden. Durch Torres; Colominas; Schlotthauer; Flandrin [44] wird daher eine Abwandlung der EEMD vorgeschlagen, die den iterativen Extraktionscharakter der ursprünglichen EMD wieder stärker aufnimmt. So wird für jede einzelne IMF das Rauschen generiert und addiert, gemittelt und das Residuum bestimmt. Mit diesem Residuum wird der Prozess für die nächste IMF wiederholt. Dieses Verfahren erlaubt es außerdem, für jede Mode einzeln das Signal-zu-Rausch-Verhältnis zu definieren. Zusätzlich wird verhindert, dass durch das Rauschen zusätzliche Moden entstehen. Problematisch an diesem Verfahren ist, dass in den ersten Moden ein hoher Rauschanteil enthalten ist und zudem die vergleichbare Information erst in späteren Moden enthalten ist als bei der EEMD. Colominas; Schlotthauer; Torres [9] schlagen allerdings Anpassungen vor, wie diesen Problemen entgegengewirkt werden kann.

5.1.3 Multivariate EMD

Oft besteht eine Messung aus mehreren Variablen, die als zusammengehörig betrachtet werden können und daher auch gemeinsam analysiert werden sollten. Dazu wird die sogenannte multivariate EMD benötigt. Ein beispielhaftes Ergebnis für diese ist in Abbildung 5.5 dargestellt. Die Anwendung der in diesem Fall bi-

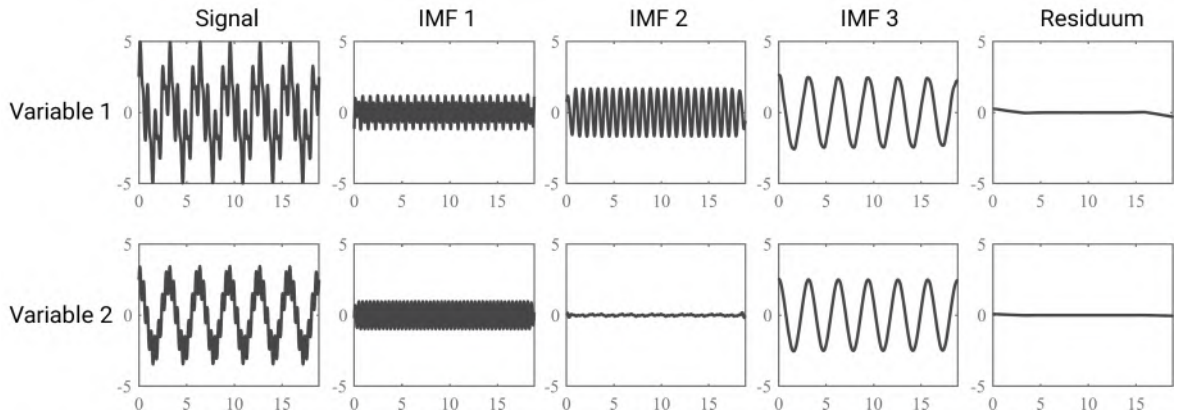
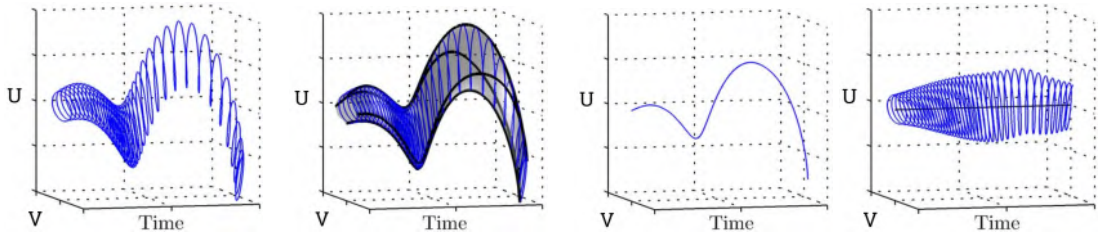


Abbildung 5.5: Beispiel für eine bivariate EMD-Anwendung (modifiziert, aus [43])

variaten EMD sorgt für eine bessere Vergleichbarkeit zwischen den Variablen in den extrahierten IMFs. Damit ist gemeint, dass bei einem zeitlichen Signal, wie in Abbildung 5.5, die Frequenzen in einer IMF über alle Variablen identisch sind. Damit kann über die Amplitude direkt verglichen werden, wie stark diese Frequenz in den Variablen enthalten ist. In diesem Fall ist beispielsweise bei der zweiten IMF zu erkennen, dass die zugehörige Frequenz ausschließlich bei Variable eins auftritt. Die Signalbestandteile aus den IMFs eins und drei sind hingegen bei beiden Variablen in etwa gleich vertreten.

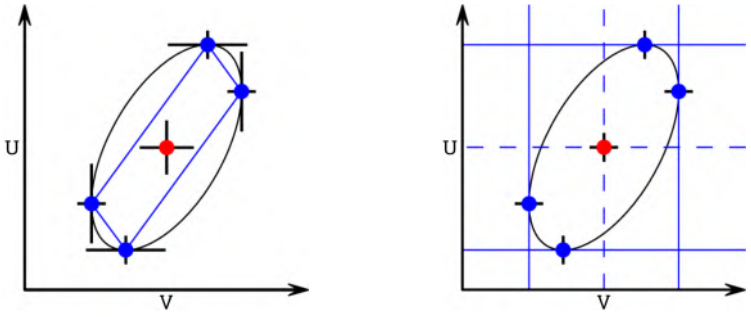
Bivariate (komplexe) EMD Zunächst soll betrachtet werden, wie eine solche EMD für ein Signal mit zwei Variablen („bivariat“) funktioniert. Das Verfahren dazu wurde durch Rilling; Flandrin; Gonçalves; Lilly [33] beschrieben. Etwa zeitgleich gab es erste Veröffentlichungen für die Applikation der EMD im komplexen Raum [38, 1], was - wie nachfolgend deutlich wird - auch als eine Form der bivariaten EMD aufgefasst werden kann. Für die bivariate EMD werden die beiden Variablen so interpretiert, dass sie einem Punkt in der komplexen Ebene zugeordnet werden können. Bei der EMD ist ein entscheidender Schritt die Bestimmung der Einhüllenden, und dazu werden die Extrempunkte benötigt. Bei einem sich im Zeitverlauf ändernden, bivariaten Signal wie in Abbildung 5.6a, können keine Extrempunkte im eigentlichen Sinne definiert werden. Daher wird an dieser Stelle stattdessen mit dem Begriff der Rotation gearbeitet: Ein bivariates Signal setzt sich aus einer Überlagerung von mehreren Rotationen zusammen. Dies kann sehr anschaulich als Analogon zu der „Überlagerung von Schwingungen“ der klassischen EMD gesehen werden, wie auch in Abbildung 5.6 zu sehen ist. Das Signal lässt sich durch Summation eines schnell und eines langsamer rotierenden Anteils beschreiben.



(a) Ausgangssignal **(b)** Einhüllende **(c)** langsamer rotierender Anteil **(d)** schnell rotierender Anteil

Abbildung 5.6: Zerlegung eines bivariaten Signals in rotierende Anteile [33]

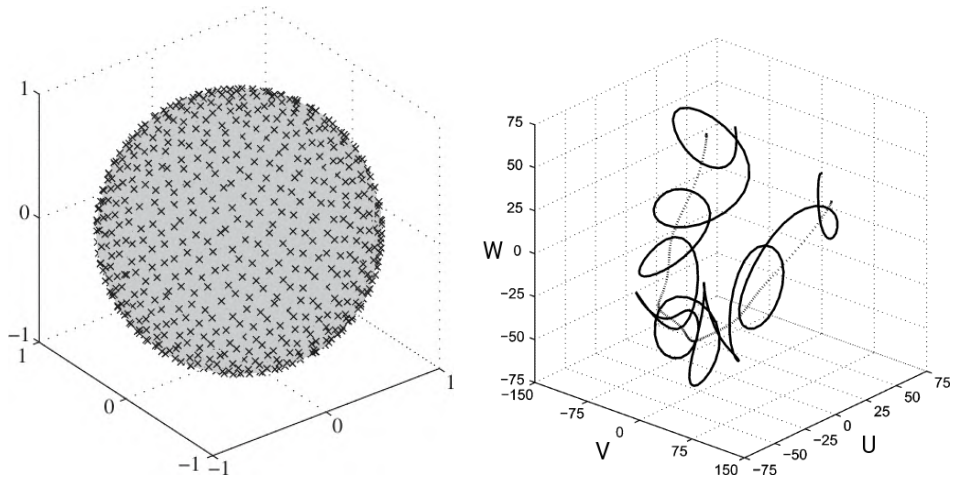
Dazu wird als Einhüllende ein „Schlauch“ um die Funktion gelegt (Abbildung 5.6b). Der langsamer rotierende Anteil (Abbildung 5.6c) entspricht dann dem Mittel der Einhüllenden, der schnell rotierende (Abbildung 5.6d) dem Ausgangssignal abzüglich dem langsam rotierenden Anteil. Die Bestimmung der Einhüllenden geschieht letztenendes ähnlich wie bei der klassischen EMD. Dazu werden Projektionen des Signals in n Richtungen vorgenommen. Für die Auswahl der Richtungsvektoren φ_i ist eine möglichst gleichmäßige Verteilung gewünscht, um alle Variablen des Signals in gleichen Teilen zu berücksichtigen. Daher werden äquidistante Punkte auf einem Einheitskreis ausgewählt, um die Richtungsvektoren zu definieren. Entlang jeder dieser Richtungen werden die Maxima bestimmt und eine kubische Spline-Interpolation berechnet. Dadurch ist für jeden beliebigen Zeitpunkt die Einhüllende zumindest an n Punkten bekannt. Aus diesen Punkten kann nun auf verschiedenen Wegen ein Mittelwert bestimmt werden (siehe Abbildung 5.7). Eine Möglichkeit besteht darin, über alle Abtastpunkte das Baryzentrum des aufgespannten Polygons zu bestimmen. Rilling; Flandrin; Gonçalves; Lilly [33] argumentieren, dass die in Abbildung 5.7b gezeigte Methode weniger anfällig für Abtastfehler ist und daher verwendet werden sollte. Für diese Methode werden horizontal wie vertikal die Tangenten der Einhüllenden im Abtastpunkt bestimmt. Mittig zwischen den horizontalen und vertikalen Tangenten wird je eine mittlere Gerade definiert (in der Abbildung gestrichtelt). Der Schnittpunkt dieser Geraden wird als Mittelpunkt genutzt. Das Verfahren kann mit dem S-Nummern-Kriterium als Abbruchkriterium analog zum Sifting Prozess der EMD eingesetzt werden.



(a) (b)

Abbildung 5.7: Methoden zur Bestimmung des Mittels der Einhüllenden eines bivariaten Signals [33]
(a) Baryzentrum gleichgewichteter Punkte auf der Einhüllenden
(b) Schnittpunkt der Geraden mittig zwischen den horizontalen/vertikalen Tangenten

Tri- und n-Variate EMD Die Verallgemeinerung der Idee der bivariaten EMD zur trivariaten EMD [29] führte fast unmittelbar zur multivariaten EMD mit beliebig vielen Variablen [30]. Aus diesem Grund werden diese beiden Weiterentwicklungen nicht getrennt voneinander vorgestellt. Der Kernpunkt der Überlegungen ist einmal mehr die Definition einer Einhüllenden um das Signal, die eine Bestimmung des lokalen Mittels ermöglicht. Bei der bivariaten EMD ist dieses Problem durch die Projektion in verschiedene Richtungen in die komplexe Ebene gelöst worden. Für diese können die Extrempunkte bestimmt und somit auch eine Einhüllende angenähert werden. Dieser Gedanke wird für die trivariate EMD fortgeführt [29]: Statt einem Einheitskreis wird zur Definition der Richtungsvektoren nun aber eine Einheitskugel verwendet, wie in Abbildung 5.8a gezeigt wird. Die n-variate EMD nutzt in logischer Fortführung Hypersphären, respektive $n-1$-Sphären. Für eine möglichst gleichmäßige Punktverteilung auf der Sphäre wird ein spezielles Verfahren angewandt, dieses wird in Rehman; Mandic [30] beschrieben. Wie bei der bivariaten EMD kann über die projizierten Signale ein Teil der Einhüllenden bestimmt werden, indem durch die Maxima interpoliert wird. Das lokale Mittel der n-variaten Funktion wird dann als Durchschnitt der durch Projektion und Interpolation bekannten Punkte auf der Einhüllenden bestimmt. Ein Ergebnis dazu ist beispielsweise in Abbildung 5.8b dargestellt, eine allgemeine algorithmische Beschreibung findet sich in Algorithmus 2.



(a) (b)

Abbildung 5.8: Bestimmung des lokalen Mittels eines trivariaten Signals
(a) Wahl der Richtungsvektoren für die Projektionen [30]
(b) Beispiel eines trivariaten Signals (durchgezogen) und lokales Mittel (gestrichelt) [29]

Algorithmus 2 EMD-Algorithmus für n-variate Signale

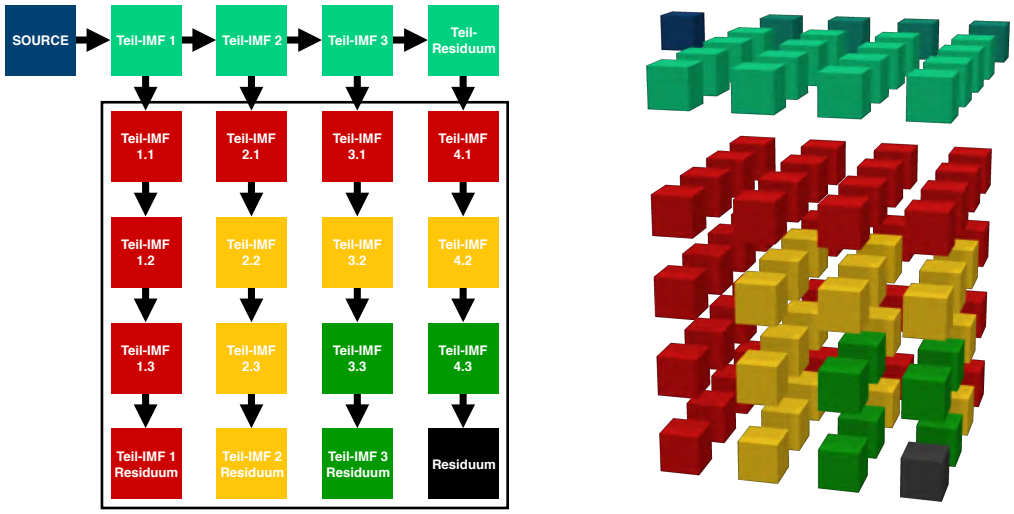
1: Wähle p Abtastpunkte und damit Projektionsrichtungen auf einer $(n-1)$-Sphäre
2: Berechne für jeden Richtungsvektor die Projektion des n-variaten Eingangssignals $h(k)$
3: Suche die Stellen, für welche die projizierten Signale maximal werden
4: Interpoliere durch die Maxima, um die n-variate Einhüllende zu bestimmen
5: Berechne das lokale Mittel $m(k)$ des Signals als Durchschnitt der p durch Projektion und Interpolation bekannten Punkte auf der Einhüllenden.
6: Berechne $s(k) = h(k) - m(k)$.
7: Ist das Abbruchkriterium erfüllt, betrachte $s(k)$ als IMF, ansonsten setze $h(k) = s(k)$ und wiederhole den Prozess.

5.1.4 Mehrdimensionale EMD-Erweiterungen

Insbesondere im Hinblick auf die Rechenintensität stellt die Erweiterung auf mehrere Dimensionen ein Problem dar. Aus diesem Grund wurden verschiedene Ansätze und Abwandlungen vorgeschlagen und diskutiert, die sich vor allem in der Bestimmung der Einhüllenden - und damit der lokal gemittelten Funktion - unterscheiden. Für diesen Schritt ist wiederum die Interpolation zwischen den Extrempunkten entscheidend. Auch deren Bestimmung ist im mehrdimensionalen Raum komplexer. Für die mehrdimensionale Interpolation gibt es verschiedene Möglichkeiten, wie die Nutzung von Radialen Basis Funktionen oder Thin Plate Splines. Beispielsweise wird für den zweidimensionalen Fall bei der Bidimensional Empirical Mode Decomposition (BEMD) [25] eine Fläche statt (eindimensional) einer Kurve interpoliert [25]. Dies stellt eine direkte Verallgemeinerung der EMD dar, ein Überblick über solche Methoden für den zweidimensionalen Fall findet sich in Bhuiyan; Attoh-Okine; Barner; Ayenu-Prah; Adhami [4]. Eines der Hauptprobleme dieser Ansätze ist der hohe Berechnungsaufwand für die mehrdimensionale Interpolation.

Multidimensional EEMD (MEEMD) Die MEEMD [49], beschreibt eine pseudo-n-dimensionale Erweiterung der EMD. Der Ansatz ist dabei sehr simpel: Statt tatsächlich eine mehrdimensionale EMD zu berechnen, wird der Datensatz entlang der Achsen „aufgeschnitten“, bis nur Datenvektoren übrigbleiben, auf welche die eindimensionale EMD angewandt werden kann. Diese wird dann für den gesamten Datensatz zunächst entlang der ersten Achse (in Abbildung 5.9 horizontal) ausgeführt. Das heißt für den zweidimensionalen Fall, dass für jede Zeile einzeln die

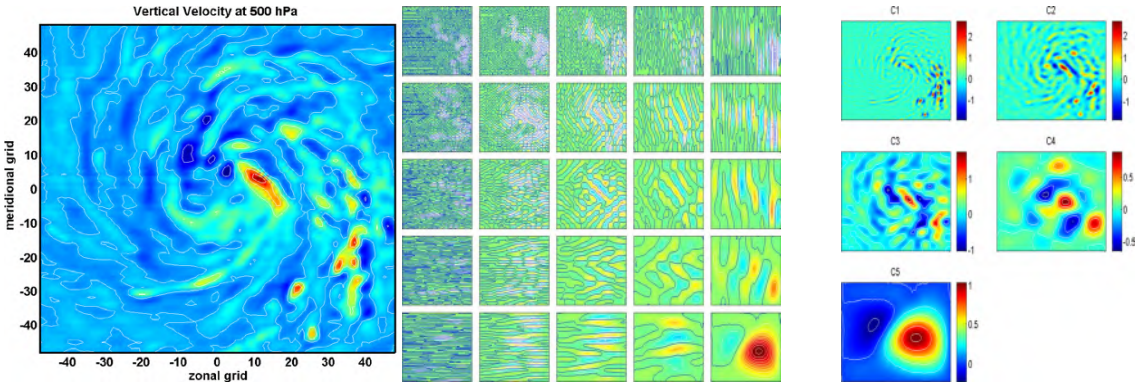
EMD angewandt wird. Die Ergebnisdatensätze, die so berechnet werden, werden anschließend nach demselben Vorgehen entlang der zweiten Achse analysiert (in der Abbildung vertikal). Im zweidimensionalen Fall wird jetzt also für jede Spalte die EMD angewandt. Von den so gewonnenen „Teil-IMFs“ müssen jetzt noch diejenigen kombiniert werden, die etwa gleichskalige Strukturen enthalten. Wie Wu; Huang; Chen [49] zeigen, ist dafür keine weitere Analyse notwendig. Stattdessen können die Teil-IMFs einfach wie in Abbildung 5.9 farblich dargestellt zusammengefasst werden. Alle rot eingezeichneten Teil-IMFs gehören zu der Ergebnis-IMF 1, alle gelben zur Ergebnis IMF 2 und so weiter. Durch das Kombinationsschema ist bereits implizit sichergestellt, dass dies berücksichtigt wird.



(a) 2D **(b)** 3D

Abbildung 5.9: MEEMD: Zerlegung und Kombination für IMFs

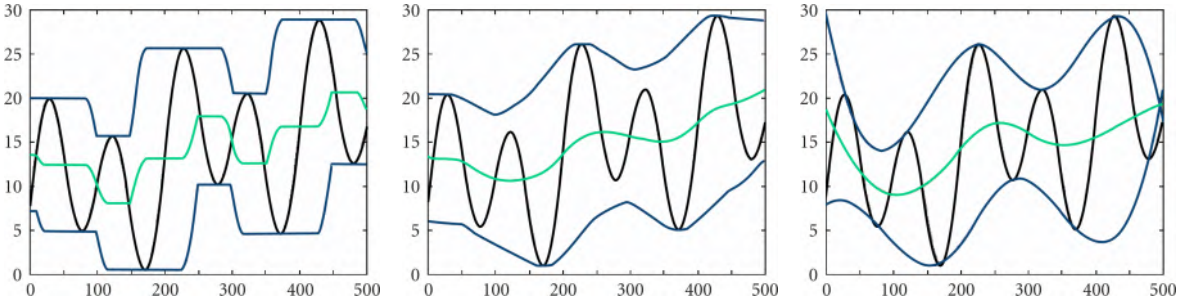
In Abbildung 5.10 ist ein Beispiel der MEEMD für den zweidimensionalen Fall in der Anwendung gezeigt. Der Datensatz (Abbildung 5.10a) wird zunächst gemäß des Algorithmus in die „Teil-IMFs“ aufgeteilt (Abbildung 5.10b) und dann zu den Ergebnis-IMFs zusammengefasst (Abbildung 5.10c). In Abbildung 5.10b ist auch zu erkennen, dass die Teil-IMFs, die zusammengefasst werden (vgl. Abbildung 5.9) über etwa gleichgroße Strukturen verfügen. Für dieses Verfahren wurde auch ein Ansatz zur Parallelisierung vorgestellt, der im Wesentlichen den mehrdimensionalen Datenzugriff optimiert [6].



(a) Ausgangsdaten **(b)** „Teil-IMFs“ **(c)** Ergebnis-IMFs

Abbildung 5.10: MEEMD am Beispiel von Wetterdaten [49]

Fast and Adaptive EMD Eine sehr vielversprechende Methode zur Beschleunigung des Verfahrens ist die Vermeidung des rechenintensivsten Schrittes - der Interpolation. Die Einhüllende der Funktion wird stattdessen näherungsweise durch gleitende Minima / Maxima Funktionen über das Signal bestimmt und anschließend geglättet (siehe Abbildung 5.11). Da gleitende Minima / Maxima auch für mehrere Dimensionen mit vertretbaren Aufwand bestimmt werden können, beschreibt dieser Ansatz eine „echte“ multidimensionale Erweiterungen der EMD und reduziert das Problem nicht temporär auf den eindimensionalen Fall. Für die Bestimmung der Einhüllenden muss allerdings zunächst die passende Fenstergröße der gleitenden Min-/Max-Funktion bestimmt werden [3]. Diese kann aus dem maximalen und minimalen Abstand zwischen den Extrempunkten bestimmt werden. Die Anwendung dieses Verfahrens liefert eine hinreichende Näherung für die Einhüllende [3].



(a) Min-/Max-Filter **(b)** Geglättet **(c)** Spline-Interpolation

Abbildung 5.11: Bestimmung der Einhüllenden durch Min/Max-Filter und Glättung (modifiziert, aus [3])

Außerdem konnte gezeigt werden, dass diese Modifikation eine deutliche Beschleunigung der Berechnungsdauer darstellt [3]. Für die Bestimmung der minimalen und maximalen Abstände zwischen den Extrempunkten kann die Delaunay-Triangulierung genutzt werden [7]. Bei der Delaunay Triangulierung werden immer drei Punkte zu einem Dreieck verbunden, wenn in dem Umkreis des resultierenden Dreiecks kein weiterer Punkt liegt. Die logische Fortführung dieses Verfahrens für den dreidimensionalen Fall wurde durch Riffi; Mahraz; Abbad; Tairi [32] vorgestellt. Die Delaunay-Triangulierung wird durch eine Tetraedrisierung ersetzt - statt des Umkreises wird eine Kugel verwendet. Kürzlich wurden weitere Arbeiten zur Beschleunigung dieses Verfahrens veröffentlicht [13]. Dabei wird ausgenutzt, dass die Minima- und Maxima-Filter auch iterativ immer entlang einer Dimension angewandt werden können.

5.1.5 Fast and Adaptive Multivariate and Multidimensional EMD (FA-MVEMD)

Diese Variante vereint Konzepte der multivariaten EMD (Abschnitt 5.1.3) und der Fast und Adaptive Multidimensional EMD (Abschnitt 5.1.4) [43]. Dazu sind einige Anpassungen notwendig. Bei der multivariaten EMD wird zur Bestimmung der Einhüllenden der Funktion das zusammengesetzte Signal in verschiedene Richtungen projiziert und durch die projizierten Punkte interpoliert. Bei der *Fast and Adaptive EMD* wird für die Einhüllende hingegen statt der Interpolation ein gleitender Min-/Max-Filter genutzt. Die Filtergröße wird aus dem Abstand der Extrempunkte des Signals bestimmt. Bei einem multivariaten Signal genügt zur Bestimmung der Extrempunkte eine Projektion in eine Richtung, insofern in dieser Richtung alle Variablen gleichmäßig berücksichtigt werden. Daher wird für die Projektion der Richtungsvektor $\begin{pmatrix} 1/\sqrt{n} \\ \cdots \\ 1/\sqrt{n} \end{pmatrix}$ vorgeschlagen [43]. Mithilfe der Extrempunkte der Projektion des Signals in diese Richtung kann anschließend die Fenstergröße bestimmt werden. Danach wird für jede Variable einzeln der gleitende Mittelwert bestimmt, allerdings mit der selben Filtergröße. Dadurch wird implizit erreicht, dass in den extrahierten Moden die Strukturen variablenübergreifend gleichgroß sind.

Definition der Filtergröße für den gleitenden Mittelwert Die Entscheidung über die Fenstergröße zur Berechnung des gleitenden Mittelwertes basiert, wie bereits geschrieben, auf den Distanzen zwischen den Extrempunkten der Funktion. Unter-

schiedliche *Fenstertypen* definieren, wie aus diesen Distanzen die Filtergröße bestimmt werden soll:

Typ 1: aus dem kleineren Wert von
- der kleinsten Distanz zwischen den Minima der Funktion und
- der kleinsten Distanz zwischen den Maxima der Funktion

Typ 2: aus dem größeren Wert von
- der kleinsten Distanz zwischen den Minima der Funktion und
- der kleinsten Distanz zwischen den Maxima der Funktion

Typ 3: aus dem kleineren Wert von
- der größten Distanz zwischen den Minima der Funktion und
- der größten Distanz zwischen den Maxima der Funktion

Typ 4: aus dem größeren Wert von
- der größten Distanz zwischen den Minima der Funktion und
- der größten Distanz zwischen den Maxima der Funktion

Typ 5: aus dem Durchschnitt der Varianten 1-4

Typ 6: aus dem Median der Distanzen zwischen den Extrempunkten

Typ 7: aus der Mode, also dem häufigsten Wert unter den Distanzen zwischen den Extrempunkten

Die Fenstertypen 1-4 stammen von Bhuiyan; Adhami; Khan [3], die Typen 5-7 wurden hingegen von Thirumalaisamy [42] zusätzlich definiert. Die Wahl des Fenstertyps bestimmt dabei maßgeblich, wie groß die Strukturen sind, die extrahiert werden. Fenstertyp 1 resultiert in den kleinsten und Fenstertyp 4 in den größten Strukturen.

Abbruchkriterium Eine weitere Anpassung ist die Definition des Abbruchkriteriums. Diese ähnelt am ehesten dem ursprünglich für die EMD vorgeschlagenen, Cauchy-ähnlichen Kriterium. Es wird ein Toleranzwert μ vorgegeben, der unterschritten werden muss, um zum Abbruch zu führen. Dieser beschreibt, wie stark ein betrachtetes Signal von dem zugehörigen gleitenden Mittelwert abweicht. Die Berechnung erfolgt über das Verhältnis der Quadrate des gleitenden Mittelwertes m_i der Variable i zum analysierten Signal h_i über alle Datenpunkte, und muss für jede Variable erfüllt sein:

$$\frac{\sum m_i^2}{\sum h_i^2} < \mu \tag{5.1}$$

5.2 Rahmenbedingungen für die EMD Implementierung

Die Toolbox dient zur Auswertung von Geschwindigkeitsfeldern. Die Anwendung der EMD auf diese stellt ein noch relativ neues Forschungsgebiet dar. Somit muss zunächst untersucht werden, welche Anforderungen an die EMD aus dem Anwendungskontext entstehen. Vorab sollte klargestellt werden, dass es bei der Anwendung der EMD in diesem Fall nicht um zeitliche Signale geht. Stattdessen sollen Strukturen mit räumlicher Ausdehnung in den Geschwindigkeitsfeldern untersucht werden.

Anforderungen Der Anwendungsfall, für den die EMD eingesetzt werden soll, ist der Vergleich von Strukturen unterschiedlicher Größe in Geschwindigkeitsfeldern. So soll beispielsweise untersucht werden, ob die wandfernen großskaligen Strukturen Einfluss auf die wandnahen, kleinskaligen Strukturen nehmen. Wie bereits beschrieben, liegen Geschwindigkeitsfelder in zwei- und dreidimensionaler Ausprägung vor. Auf beide soll das Verfahren angewandt werden können. Bei einer Messung wird außerdem eine Vielzahl an Datensätzen aufgenommen, die verarbeitet werden müssen.
Daraus folgen als Anforderungen an die EMD-Implementierung:

- Umgesetzt werden soll eine der mehrdimensionalen Varianten für zwei und drei Dimensionen.
- Da Mode Mixing die Vergleichbarkeit der IMFs negativ beeinflusst, soll es verhindert werden.
- Der Zeitbedarf muss in einem akzeptablem Rahmen bleiben, um mehrere Datensätze auswerten zu können.
- Die einzelnen Geschwindigkeitskomponenten hängen miteinander zusammen. Eine multivariate EMD - Variante wäre wünschenswert.

Definition Eingabe- und Ausgabedaten Als Eingabe-Daten dienen Ergebnisse von PIV- und STB-Messungen. Dabei ist eine Bedingung für verschiedene Varianten der multidimensionalen EMD, dass die Daten auf einem geordneten Gitter mit gleichmäßigen Abständen vorliegen müssen. Wenn dies nicht der Fall ist, kann dazu das in der Toolbox enthaltene Verfahren zur Überführung der Daten auf ein solches Gitter genutzt werden.

Verfügbare Bibliotheken In Tabelle 5.1 werden verschiedene Bibliotheken aufgeführt, die eine Implementierung der EMD anbieten. Verglichen werden hier die verfügbaren Methoden, Unterstützung für mehrere Variablen, Verfügbarkeit von Routinen für die Verarbeitung von zwei- und dreidimensionalen Daten sowie die Existenz einer C++-Schnittstelle. Hier ist nur eine Auswahl zu sehen, insbesondere für MATLAB gibt es eine Vielzahl weiterer Bibliotheken. Für die klassische, eindimensionale EMD sind in fast jeder Programmiersprache Bibliotheken vorhanden. Für den zweidimensionalen Fall sind EMD-Implementierung häufig auch in Bildverarbeitungsbibliotheken vorzufinden, wie im Falle des unten aufgeführten *R Package EMD*. Bei diesem wurden von den Autoren eigene Modifizierungen der EMD [19] und der BEMD [22] eingebracht. Die *libeemd* ist die einzige Bibliothek, die explizit für die Nutzung in C/C++ geschrieben ist. Die letzten beiden in der Tabelle enthaltenen Bibliotheken sind für die Nutzung in MATLAB ausgelegt. Allerdings ist die vorletzte in C geschrieben und könnte daher in einem C++-Programm adaptiert werden. Da die Erweiterung hinsichtlich mehrerer Dimensionen mit dem Konzept der MEEMD hinreichend einfach scheint, soll die fehlende Unterstützung mehrerer Dimensionen kein Ausschlusskriterium sein. Auch die multivariate EMD-Erweiterung ist nicht zwingend notwendig - wenn, dann sollten jedoch mindestens drei Variablen unterstützt werden. Der Grund dafür ist, dass in den Daten, für welche die Toolbox genutzt werden soll, in der Regel die Geschwindigkeit und/oder Beschleunigung für alle drei Raumrichtungen abgespeichert wird. Da die Toolbox bereits ein bestehendes, in C++ geschriebenes Projekt darstellt, ist diese Unterstützung besonders wichtig. Damit reduziert sich die Auswahl im Wesentlichen auf die *libeemd* und FA-MVEMD, die auch die beiden aktuellsten Bibliotheken sind. Als kompletteste Bibliothek kann die FA-MVEMD angesehen werden, allerdings fehlt die direkte Kompatibilität zu C++. Aus diesem Grund soll zunächst die *libeemd* genutzt und der MEEMD-Ansatz implementiert werden.

	Sprache	Methode	Multi-Var	2D	3D
libeemd [24]	C++	EEMD, CEEMDAN			
R Package EMD [20]	R	EMD, BEMD		✓	
EMD [33]	MATLAB, (C)	EMD, Local, Online	✓(<=2)		
FA-MVEMD [43]	MATLAB	FA-MVEMD	✓(<=3)	✓	✓

Tabelle 5.1: Vergleich verschiedener EMD-Bibliotheken

5.3 Implementierung der Multidimensional Ensemble EMD (MEEMD)

Nachfolgend wird die Implementierung der MEEMD beschrieben. Dazu wird zunächst kurz die genutzte Bibliothek vorgestellt. Da diese nur mit eindimensionalen Daten arbeiten kann, wird darauffolgend die Implementierung beschrieben, mit der mithilfe der Bibliotheksaufrufe das Konzept der MEEMD umgesetzt wird. Anschließend folgt eine Untersuchung der Qualität der Ergebnisse sowie des Zeitbedarfs der Rechnung und eine Bewertung dessen im Hinblick auf die geplante Anwendung.

5.3.1 Vorstellung der libeemd Bibliothek

Die libeemd-Bibliothek stellt für die EMD zwei Methoden zur Verfügung: eemd und ceemdan. Die Parameter sind bei beiden identisch:

- Pointer auf Eingabe-Datenarray
- Anzahl Punkte (n)
- Pointer auf Array für das Ergebnis
- Anzahl der zu extrahierenden IMFs *
- Ensemble-Größe (M) *
- Stärke des Rauschens *
- S-Nummer *
- Maximale Anzahl an Sifting-Iterationen *
- Seed für die Zufallszahlen Generierung für die Rauschgenerierung

Die mit * markierten Punkte können vom Nutzer des Programms in einem Dialog eingestellt werden, ebenso die Entscheidung, ob eine EEMD oder CEEMDAN ausgeführt werden soll. Das Ergebnis-Array muss eine Speicherkapazität von $n * M$ Werten haben. Die ersten n Werte gehören dann zur ersten IMF, die folgenden n zur zweiten und so weiter.
Bei der MEEMD ist ursprünglich die Nutzung der EEMD zur Anwendung auf die eindimensionalen Daten angedacht. Da die Bibliothek mit der CEEMDAN aber auch über eine Weiterentwicklung der EEMD verfügt, werden beide in der Toolbox zur Verfügung gestellt und nachfolgend getestet. Nichstdestotrotz wird einheitlich von der MEEMD gesprochen, die schließlich das Prinzip beschreibt.

5.3.2 Implementierung der Logik für die mehrdimensionale Ausführung

Für den Prozess der MEEMD sind verschiedene Schritte notwendig: Zunächst müssen die Daten in die einzelnen Datenvektoren zerlegt werden, auf welche die EEMD angewandt werden soll. Dabei ist auch eine Umsortierung notwendig. Die durch den Bibliotheksaufruf erhaltenen IMFs für diesen Datenvektor müssen immer in den entsprechenden Zwischenergebnis-Datensatz einsortiert und abgespeichert werden. Nach Abschluss aller Zerlegungen muss das Kombinieren nach dem in Abschnitt 5.1.4 beschriebenen Verfahren vorgenommen werden.

Umsortierung der Daten und Anwenden der EEMD Wie beschrieben, stellt die Bibliothek nur Funktionen für die eindimensionale EEMD mit einer Variable zur Verfügung. Das heißt, als Input wird ein Array erwartet, das für jeden Datenpunkt genau einen Wert enthält. Die Daten müssen daher so aufgearbeitet werden, dass sie der von der Bibliothek definierten Schnittstelle entsprechen. Dazu sind zwei Schritte notwendig: Zum einen die Aufteilung in einzelne Datenvektoren nach der aktuell zu analysierenden Dimension, zum anderen die Aufteilung nach den im Datensatz enthaltenen Variablen. Ein Datenvektor enthält dabei die Werte einer Variablen über eine Reihe von Messpunkten (bei der zweidimensionalen Anwendung Zeilen oder Spalten). Die Schnittstelle für den Zugriff auf den Datensatz sieht nur einen Zugriff pro Messpunkt (die Werte aller Variablen des Punktes) oder pro Variable (die Werte aller Messpunkte für diese Variable) vor. Es ist an dieser Stelle unerheblich, welcher Zugriff gewählt wird, mit beiden ist eine akzeptable Lösung konstruierbar. In der Implementierung wird der Zugriff pro Variable gewählt, da so die Koordinaten des Punktes nicht erst aussortiert werden müssen. Der Rückgabevektor über die gesamte Variable wird dann in die einzelnen Zeilen oder Spalten aufgeteilt und für jede die EEMD ausgeführt. Nach dem Bibliotheksaufruf müssen die Daten gemäß ihrer Zugehörigkeit zu den IMFs aufgeteilt und die zur Anwendung der EEMD notwendige Umsortierung rückgängig gemacht werden. Die Ergebnisse werden also direkt in dafür vorgesehene Ergebnisdatensätze einsortiert.

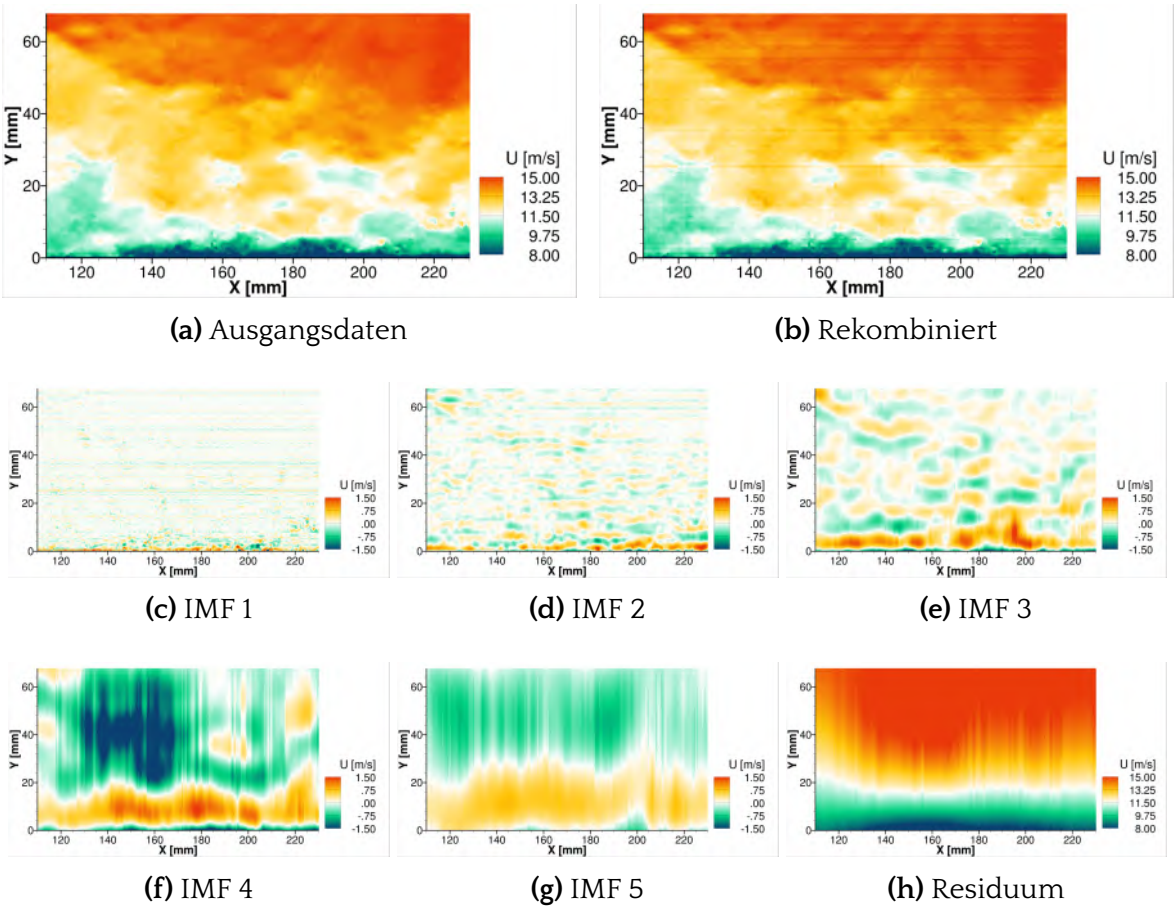
Speichern von Zwischenergebnissen Bei der MEEMD werden große Mengen an Zwischenergebnissen produziert, die nur temporär benötigt werden. Allerdings sind sie zu umfangreich, um sie dauerhaft im Arbeitsspeicher zu halten. Daher wird ein temporärer Ordner angelegt, in den die Zwischenergebnisse mit einem binären Datenformat (Tecplot .szplt) abgelegt werden. Dieses zeichnet sich durch

die höchste Schreib-/Lesegeschwindigkeit aus. Im Arbeitsspeicher wird lediglich der Dateipfad gehalten. Da im letzten Schritt der MEEMD die Datensätze nach dem oben beschriebenen Muster aufaddiert werden, ist es auch an dieser Stelle nicht notwendig, mehr als zwei Datensätze gleichzeitig im Arbeitsspeicher zu halten. So kann die gesamte Berechnung mit geringem Arbeitsspeicher-Aufwand auf jedem Desktop-PC ausgeführt werden - limitierend ist lediglich der Rechenaufwand.

„Combine“ - Schritt Nach der Bestimmung der EEMD für jeden Datenvektor entlang jeder Achse müssen die „Teil-IMFs“ nach dem beschriebenen Verfahren zusammengerechnet werden. Dazu werden bereits bei dem EMD-Durchlauf entlang der letzten Achse die Ergebnisse entsprechend ihrer Zugehörigkeit zu den Ergebnis-IMFs gruppiert. Für das eigentliche Zusammenrechnen wird dann aus jeder Gruppe immer ein Datensatz geladen und die Werte eines jeden Punktes auf den Ergebnisdatensatz aufaddiert. Da die Koordinaten der Punkte zwischen allen Datensätzen übereinstimmen, kann die Zuordnung, auf welchen Punkt aufaddiert werden muss, über die Indizes durchgeführt werden und benötigt somit keinen zusätzlichen Rechenaufwand. Die so erhaltenen, tatsächlichen IMFs werden in den vom Nutzer vorgegebenen Ordner gespeichert.

5.3.3 Qualitätsuntersuchungen

Zur Analyse der Funktionsfähigkeit und Einsetzbarkeit der MEEMD wurden Geschwindigkeitsfelder als Eingangsdaten genutzt. In diesem Fall stammen sie aus einer PIV-Messung und verfügen lediglich über zwei Ortskoordinaten, sodass die MEEMD zweidimensional ausgeführt wird. Dargestellt ist jeweils ein Ausschnitt der X/Y-Ebene mit der Geschwindigkeit in X-Richtung (U) farbig eingezeichnet. Dies wird hier stellvertretend für die Anwendung der MEEMD genutzt und gezeigt. Die genaue Bedeutung der einzelnen Moden und Geschwindigkeitsfelder im physikalischen Kontext ist in diesem Fall nicht relevant, da vor allem das Prinzip der MEEMD untersucht werden soll. Dennoch sollte erwähnt sein, dass bei den Ergebnissen zwei Farbskalen angewandt wurden: Eine für die Ausgangsdaten, das rekombinierte Geschwindigkeitsfeld und das Residuum; eine zweite, mit kleinerem Wertebereich bei den IMFs. Der Grund dafür ist die bessere Erkennbarkeit der Details der Moden. Eine Eigenschaften des in Abbildung 5.12 dargestellten Ergebnisses fällt direkt ins Auge: Insbesondere in den IMFs 4 und 5 sowie dem Residuum sind linienartige



(a) Ausgangsdaten **(b)** Rekombiniert

(c) IMF 1 **(d)** IMF 2 **(e)** IMF 3

(f) IMF 4 **(g)** IMF 5 **(h)** Residuum

Abbildung 5.12: Anwendung der MEEMD: Ausgangssignal und Intrinsic Mode Functions am Beispiel eines Geschwindigkeitsfeldes

Artefakte zu sehen. Die Gründe dafür sind in Wu; Huang; Chen [49] beschrieben: Da jede Zeile und Spalte einzeln analysiert wird, kann bei jeder individuell Mode Mixing auftreten. Bei der Nachbarspalte tritt dies eventuell nicht oder anders auf, was zu den Artefakten führt. Das Problem sollte jedoch durch Anwendung der EEMD gelöst werden - allerdings erst bei sehr großen Ensemble-Größen. In dem hier gezeigten Beispiel wurde die Ensemble-Größe absichtlich sehr klein gewählt (in diesem Fall 20), sodass der Effekt nicht „versteckt“ wird. Eine Analyse der Veränderung bei unterschiedlicher Ensemble-Größe findet sich nachfolgend. Es handelt sich hierbei um einen methodisch bedingten Fehler, der sich auch in anderen Veröffentlichungen findet, in denen die EMD auf diese Art angewandt wurde [2, 23], und wird auch als Problem des MEEMD-Ansatzes genannt [43]. Nachfolgend soll der Grund für

das Auftreten der Artefakte analysiert werden. Anschließend wird eine Möglichkeit untersucht, wie die Artefakte abgeschwächt werden können.

Vorzugsrichtung der auftretenden Artefakte Die Artefakte stellen Diskontinuitäten dar, und treten primär in Richtung der zuletzt analysierten Dimension auf. Um dies zu verifizieren, wurde die Reihenfolge der Achsen, entlang derer die EEMD angewandt wurde, getauscht (vgl. Abbildung 5.13). Bei einem wie hier zweidimensionalen Datensatz bedeutet dies, dass wenn zuerst die X-Achse analysiert wird, die linienartigen Artefakte am Ende in y-Richtung ausgebreitet sein werden (siehe Abbildung 5.13). In diesem Beispiel wurde zur Verstärkung des Effektes wiederum mit 20 eine geringe Ensemble-Größe gewählt. Es ist zu sehen, dass in den niedrigen Moden entlang der zuerst analysierten Achse vergleichbare Artefakte auftreten. Vor dem Hintergrund dieses Wissens lässt sich auf die Ursache für die Artefakte schließen: Die einzelnen Aufrufe der EEMD werden immer nur auf einen Datenvektor - in diesem Fall erst Spalten, dann Zeilen - angewandt. Die Zufälligkeit des aufaddierten Rauschens bedingt nun, dass bei zwei unmittelbar nebeneinander liegenden Datenvektoren geringfügig andere Moden gefunden werden. Sobald der Fehler in einer IMF enthalten ist, tritt er in den nachfolgenden IMFs wieder auf - und kann nach dem selben Zufallsprinzip noch verstärkt werden. Da diese Fehler damit schnell zu lokalen Extrempunkten entlang der jeweils anderen Dimension führen können, wechselt die Richtung von den niedrigen IMFs zu den höheren. Dies kann an einem Beispiel verdeutlicht werden (siehe Abbildung 5.13): Der Datensatz wird zunächst entlang der X-Achse analysiert und die Artefakte treten wie erwartet auf. Wird auf diesen entlang der Y-Achse die Analyse angewandt, so werden für die erste IMF die lokalen Extrema genutzt - und damit auch die Artefakte übernommen. In den nachfolgenden IMFs sind diese dann nur noch geringfügig enthalten. Stattdessen nimmt der Effekt entlang der Y-Achse zu.

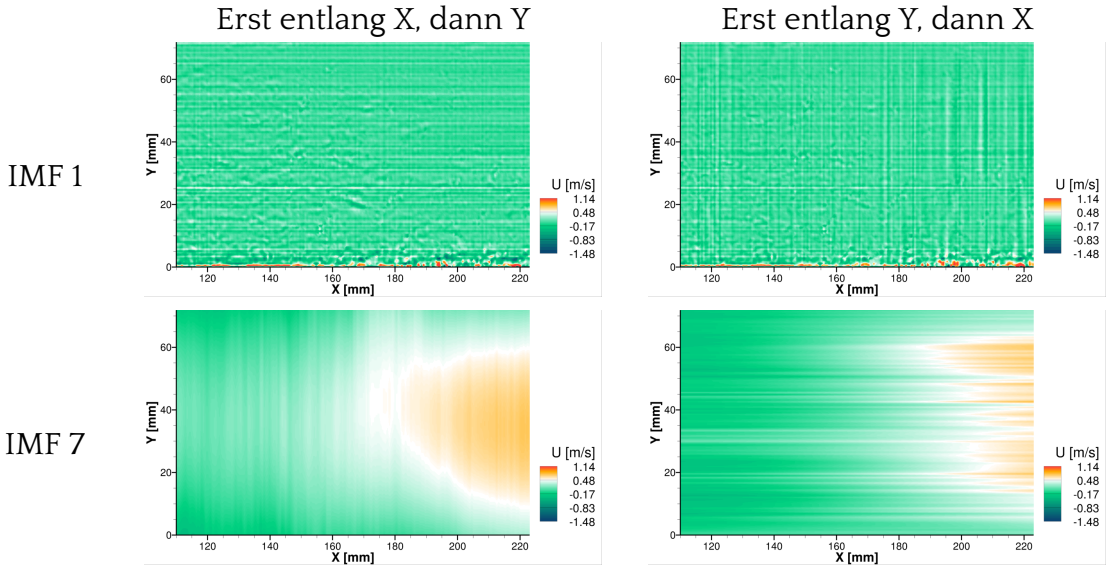


Abbildung 5.13: MEEMD: Vorzugsrichtung der Artefakte

Zusammenhang von der Ensemble-Größe und der Stärke der Artefakte Da die Artefakte durch fehlerhafte Extraktion der „Teil-IMFs“ entstehen, ist zu vermuten, dass der Effekt bei zunehmender Ensemble-Größe abnimmt. Außerdem ist es interessant, die Verwendung der CEEMDAN als EMD-Methode für die einzelnen Datenvektoren zu benutzen, da diese zusätzlich sicherstellt, dass der rekombinierte Datensatz wieder den Ausgangsdaten entspricht. Sowohl die EEMD als auch die CEEMDAN wird bei unterschiedlichen Ensemble-Größen untersucht (siehe Abbildung 5.14). Es ist gut zu sehen, wie bei größeren Ensembles die Artefakte deutlich weniger ausgeprägt sind. Geringfügig sind sie dennoch auch bei einer Ensemble-Größe von 1000 noch vorhanden. Die hohe Anzahl benötigter Ensembles für ein annähernd artefaktfreies Ergebnis ist vor dem Hintergrund des Zeitbedarfs (Abschnitt 5.3.4) problematisch. Eine weitere interessante Beobachtung ist, dass bei Verwendung der CEEMDAN auch horizontale Artefakte auftreten. Allerdings ist in diesem Fall bei einer Ensemble-Größe von 400 bereits ein annähernd artefaktfreies Bild zu erkennen. Das kann allerdings auch daran liegen, dass in der CEEMDAN-IMF deutlich kleinere Strukturen auftreten als in der zugehörigen EEMD-IMF. Dieses Phänomen hat wiederum damit zu tun, dass bei der CEEMDAN die selben Strukturen erst „später“, also in höheren IMFs als bei der EEMD, auftreten [9]. Außerdem bleibt bei der CEEMDAN ein Rest des Rauschens in den extrahierten Moden zurück (insbesondere in den

ersten drei). Alles zusammen scheint die obigen Beobachtungen zu erklären, denn durch die mehrdimensionale Anwendung ist es vorstellbar, dass die Effekte noch weiter verstärkt werden.

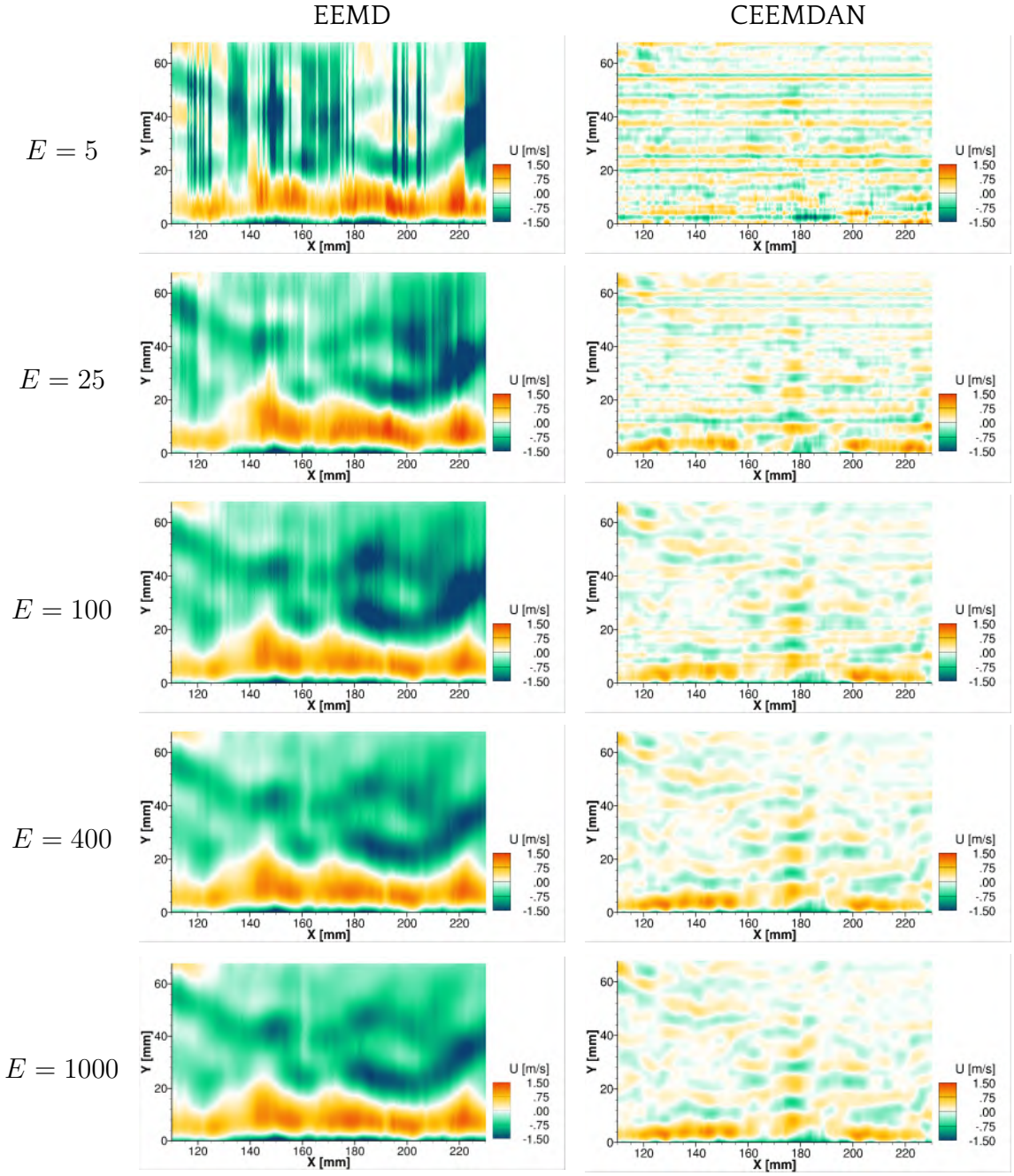
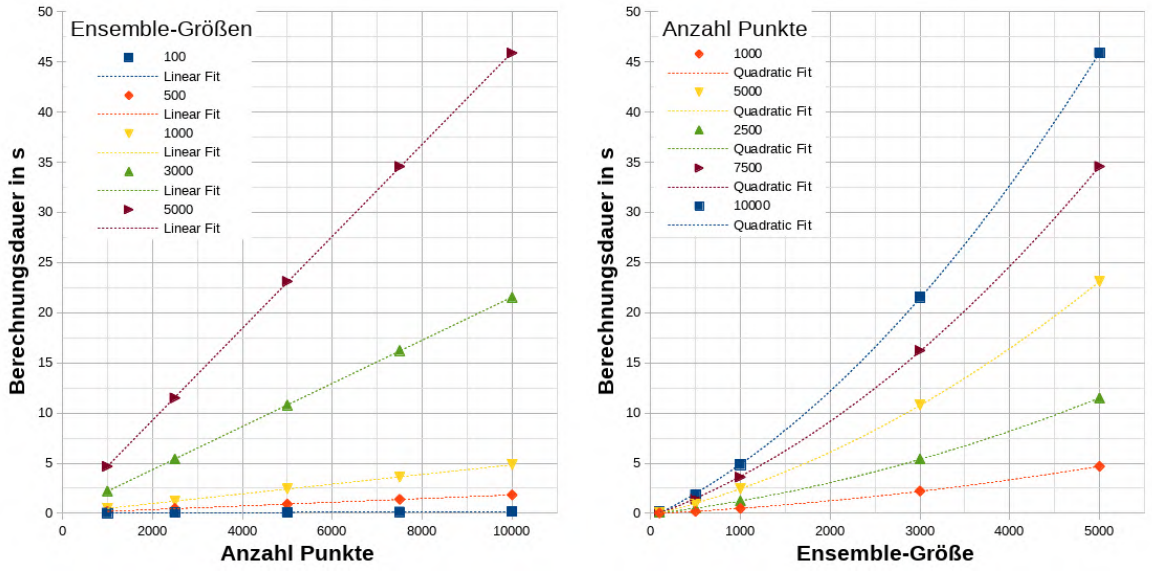


Abbildung 5.14: IMF 4 bei Anwendung nach dem Prinzip der MEEMD, mehrere Ensemble-Größen; EEMD links, CEEMDAN rechts

5.3.4 Zeitbedarf und Rechenaufwand

Da die qualitative Verbesserung dadurch erzielt wird, die Ensemble-Größe zu erhöhen, die EMD aber ohnehin schon ein rechenintensives Verfahren ist, lohnt sich ein Blick auf die Ausführungsdauer der Berechnungen. Zunächst sollen daher die Bibliotheksaufrufe untersucht werden, die den Kern der Berechnungen darstellen. Dies ist in Abbildung 5.15 gezeigt.



(a) verschiedene Anzahl Punkte **(b)** verschiedene Ensemble-Größen

Abbildung 5.15: Zeitbedarf für die Bibliotheksaufrufe der EEMD für verschiedene Ensemble-Größen und Anzahl Punkten

Die Anzahl der berechneten IMFs betrug für alle Ausführungen zehn, auch die Abbruchkriterien für den Sifting-Schritt wurden konstant bei 1 (S-Nummer) und 20 (maximale Anzahl Sifting-Iterationen) gehalten. Das Eingangssignal wurde dabei synthetisch aus mehreren überlagerten Sinus-Funktionen generiert. Während der Zeitbedarf mit der Anzahl Punkte linear steigt (Abbildung 5.15a), ist bei einer Erhöhung der Ensemble-Größe ein quadratischer Mehraufwand zu erkennen (Abbildung 5.15b). Insbesondere vor dem Hintergrund des letzten Kapitels ist dies kritisch zu betrachten, da hohe Ensemble-Größen eine mögliche Lösung zur Verhinderung der Artefakte sind. Auch wenn die gezeigten Daten ausschließlich unter Verwendung der EEMD-Funktion der Bibliothek entstanden sind, konnte stichprobenartig

sichergestellt werden, dass für diese Einstellungen bei Verwendung der CEEMDAN in etwa die selben Berechnungszeiten benötigt werden.

Mehraufwand durch Multidimensionalität und mehrere Variablen Für den realen Fall kommt noch ein erheblicher Zusatzaufwand hinzu. Denn zum einen müssen pro Dimension bereits mehrere EMD-Berechnungen ausgeführt werden (für jeden Datenvektor), anschließend muss für jede berechnete IMF wiederum eine EMD entlang der anderen Dimension ausgeführt werden (und das auch wiederum für jeden Datenvektor). Da die Bibliothek nur die Berechnung von einer einzigen Variable durchführen kann, muss das Prozedere zusätzlich für jede Variable wiederholt werden. Damit kann für die Anzahl der notwendigen Bibliotheksaufrufe folgende Formel aufgestellt werden: Ein Datensatz (beispielsweise ein Geschwindigkeitsfeld) verfügt insgesamt über D Dimensionen. Entlang einer betrachteten Dimension d hat der Datensatz i_d Punkte und es gibt n_{var} Variablen. Soll nun eine Anzahl von n_{IMF} IMFs bestimmt werden, folgt für die Anzahl der Bibliotheksaufrufe $n_{Bib}(d)$ entlang der betrachteten Dimension Gleichung 5.2:

$$n_{Bib}(d) = n_{var} * (n_{IMF})^{d-1} * \frac{\prod_{k=1}^{D} i_k}{i_d} \tag{5.2}$$

Zur Veranschaulichung dessen wird ein Beispiel betrachtet: Ein dreidimensionales Geschwindigkeitsfeld verfügt entlang der X-Achse über 200 ($i_1 = 200$), entlang der Y-Achse über 30 ($i_2 = 30$) und entlang der Z-Achse über 100 ($i_3 = 100$) Messpunkte. An jedem Punkt sind die drei Geschwindigkeitskomponenten als Variablen abgespeichert ($n_{var} = 3$). Es sollen 10 IMFs ($n_{IMF} = 10$) extrahiert werden. Für die EMD entlang der ersten Dimension müssen also für den Datenvektor der Länge $i_1 = 200$ die Anzahl Bibliotheksaufrufe $n_{Bib}(1)$ getätigt werden:

$$\begin{aligned} n_{Bib}(1) &= n_{var} * (n_{IMF})^{0} * \frac{i_1 * i_2 * i_3}{i_1} \\ &= n_{var} * i_2 * i_3 \\ &= 3 * 30 * 100 = 9000 \end{aligned} \tag{5.3}$$

Für die zweite Dimension geht auch die Anzahl IMFs in die Berechnung der Bibliotheksaufrufe ein:

$$\begin{aligned} n_{Bib}(2) &= n_{var} * (n_{IMF})^1 * \frac{i_1 * i_2 * i_3}{i_2} \\ &= n_{var} * n_{IMF} * i_1 * i_3 \\ &= 3 * 10 * 200 * 100 = 600000 \end{aligned} \tag{5.4}$$

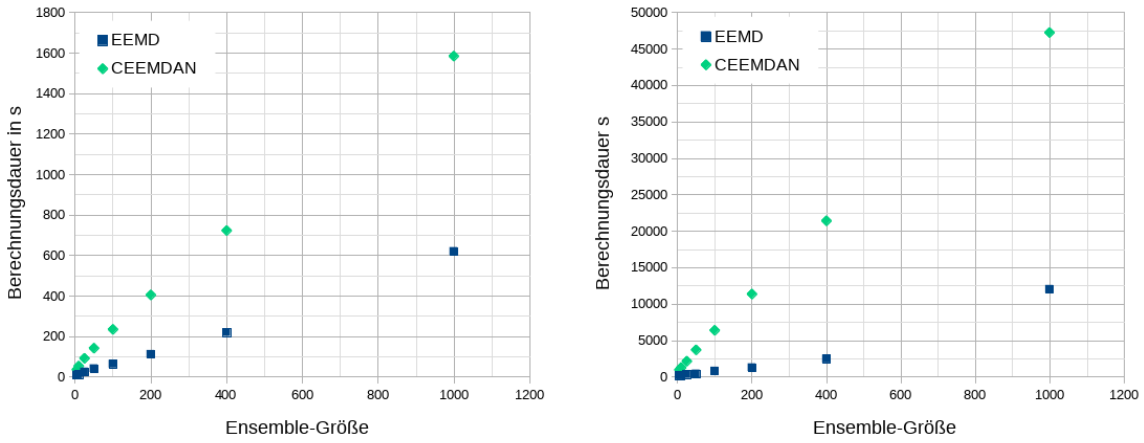
Und für die dritte Dimension ergibt sich analog:

$$\begin{aligned} n_{Bib}(3) &= n_{var} * (n_{IMF})^2 * \frac{i_1 * i_2 * i_3}{i_3} \\ &= n_{var} * (n_{IMF})^2 * i_1 * i_2 \\ &= 3 * 10^2 * 200 * 30 = 1800000 \end{aligned} \tag{5.5}$$

Gleichung 5.2 zeigt, dass der Aufwand mit der Anzahl der zu bestimmenden IMFs gemäß der Dimension potenziell steigt. Die multiplikative Verbindung der Anzahl von Punkten entlang der einzelnen Dimensionen kommt mit einem ähnlichen Effekt hinzu. Das Verfahren soll auch im Dreidimensionalen angewandt werden können. Dabei ist das Ziel, groß- und kleinskalige Strukturen miteinander zu vergleichen - womit auch 8-10 IMFs realistisch sind. Unter Berücksichtigung von Abbildung 5.15b, liegt die Vermutung nahe, dass die Artefakte nicht bei hinnehmbarer Rechenzeit ausreichend minimiert werden können.

Anwendung auf Messdaten Nichstdestotrotz soll dies anhand der Anwendung auf einen echten Datensatzes verifiziert werden. Dazu wurde sowohl bei einem zweidimensionalen, als auch bei einem dreidimensionalen Datensatz die Ensemble-Größe variiert, das Ergebnis ist in Abbildung 5.16 zu sehen. Der 2D Datensatz enthält 96390 Punkte ($765 * 126$) und 3 Variablen, für welche die EMD bestimmt wird. Der 3D Datensatz enthält 98800 Punkte ($65 * 19 * 80$) und 12 Variablen, für welche die EMD bestimmt wird. In beiden Fällen werden 6 IMFs extrahiert, die S-Nummer wird auf 6 gesetzt und eine maximale Anzahl an Sifting Iterationen von 10 gewählt. Es ist zunächst zu erkennen, dass die CEEMDAN hier deutlich langsamer ist. Ein wahrscheinlicher Grund hierfür ist, dass dadurch, dass insgesamt kleinere Strukturen extrahiert werden (Abbildung 5.14), die einzelnen Iterationen aufwändiger sind, da sie mehr Extrempunkte enthalten. Es kann weder eine nennenswerte Verringe-

rung, noch Verstärkung der Zunahme der Berechnungszeiten bei Erhöhung der Ensemble-Größe gesehen werden. Ein wahrscheinlicher Grund dafür ist, dass die größte hier betrachtete Ensemble-Größe mit 1000, bei der dennoch schon sehr gute Ergebnisse erzielt werden, noch klein im Vergleich zu den in Abbildung 5.15b untersuchten ist. Womöglich hängt der zusätzliche Zeitverlust mit der Parallelisierung der Berechnungen zusammen (bibliotheksintern), und tritt daher erst bei höheren Ensemble-Größen auf. Für die Anwendung ist dies erfreulich, da eine Erhöhung der Ensemble-Größe weniger schlimm ist, als es Abbildung 5.15b vermuten lassen würde.



(a) 2D **(b)** 3D

Abbildung 5.16: Zeitbedarf für die komplette Berechnung mit EEMD und CEEMDAN bei verschiedenen Ensemblegrößen bei zwei und drei Dimensionen

5.4 Beschleunigung der MEEMD

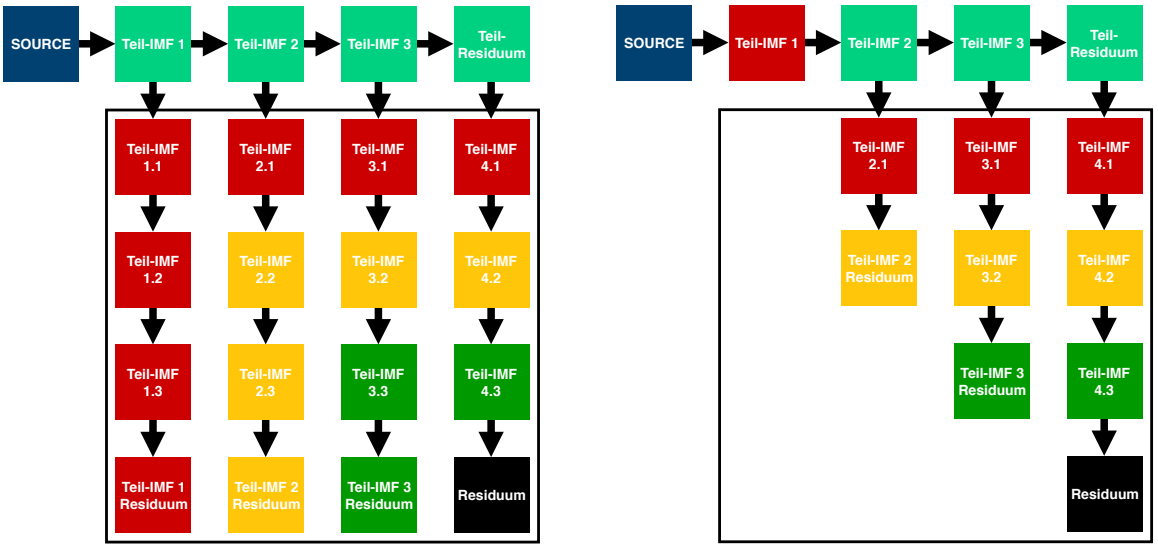
Die hohe Berechnungsdauer der MEEMD motiviert dazu, nach Möglichkeiten zur Beschleunigung der Berechnung zu suchen. Das ist insbesondere vor dem Hintergrund wichtig, dass die Artefaktproblematik nur durch die Erhöhung der Ensemble-Größe gelöst werden kann - also durch einen größeren Rechenaufwand. Nachfolgend wird ein Ansatz beschrieben, mit dem eine deutliche Beschleunigung erzielt werden kann, ohne dass eine qualitative Verschlechterung der Ergebnisse auftritt. Nachdem die grundlegende Idee dafür vorgestellt wurde, wird eine theoretische Abschätzung des Beschleunigungsfaktors vorgenommen und abschließend ein Vergleich des tatsächlichen Zeitbedarfs im Vergleich mit der MEEMD vorgenommen.

5.4.1 Idee für die Optimierung

Es gibt zwei Schlüsselgedanken, welche die Optimierung ermöglichen:

1. Summiert man die IMFs auf, erhält man wieder die Ausgangsdaten
2. Die Zerlegung in die IMFs ist ein iterativer Prozess. Dabei wird immer das Residuum der vorherigen Iteration als Ausgangsdaten für die nächste IMF-Extraktion genutzt.

Bei der MEEMD (ausführlicher beschrieben in Abschnitt 5.1.4) werden außerdem Teil-IMFs bestimmt und aus diesen zum Schluss die Ergebnis-IMF aufsummiert. Genau an dieser Stelle kann die zuvor genannte Eigenschaft ausgenutzt werden, indem untersucht wird, welche Zerlegungen in den tatsächlichen IMFs wieder zusammengerechnet werden - und somit überflüssig sind. In Abbildung 5.17 ist für den zweidimensionalen Fall dargestellt, auf welche Zerlegungen das zutrifft. Die IMFs, die aus der Zerlegung entlang der horizontalen Achse entstehen, werden anschließend entlang der vertikalen Achse zerlegt. Nach Wu; Huang; Chen [49] werden die Teile nach dem in Abbildung 5.17a dargestellten Verfahren gemäß den Farben zusammengerechnet. Für die erste IMF wird die gesamte erste Spalte und Zeile aufaddiert und findet sich im Endergebnis wieder. Allerdings zerlegt die EMD ein Signal in additive Komponenten: Die Summe der roten IMFs in der ersten Spalte entspricht damit der türkisen Teil-IMF 1 (Abbildung 5.17a). Also kann die Zerlegung an dieser Stelle gespart werden. Stattdessen wird einfach die Teil-IMF 1 für das Berechnen der Ergebnis-IMF genutzt (Abbildung 5.17b). Für die zweite IMF ist das Verfahren etwas komplexer, denn Teil-IMF 2.1 (rot) entlang der vertikalen Achse wird für die Ergebnis-IMF 1 benötigt. An dieser Stelle kann jedoch der iterative Charakter der IMF ausgenutzt werden: Wird hier nur eine IMF extrahiert, bleibt ein Residuum. Statt dieses weiter zu zerlegen, kann es als verbliebene Spaltensumme angesehen werden und für die Berechnung der Ergebnis-IMF genutzt werden. Analog dazu müssten für die nachfolgenden IMFs entlang der zweiten Achse je zwei, drei, vier etc. IMFs bestimmt werden. Streng genommen verläuft auch die Bestimmung der ersten IMF nach dem selben Verfahren: Da in diesem Fall Null IMFs bestimmt werden, sind Eingangsdaten (Teil-IMF 1) und das Residuum identisch. So muss letztenendes bei den Zerlegungen entlang der zweiten Achse nur ein einziges Mal die tatsächlich gewünschte Anzahl IMFs bestimmt werden.

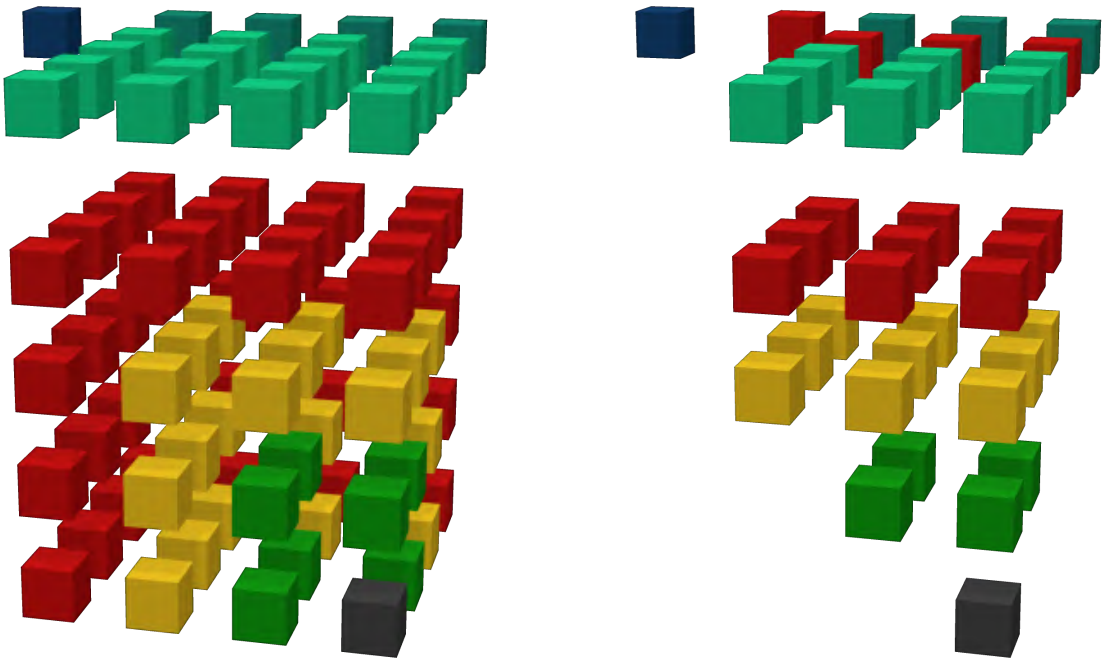


(a) Klassische MEEMD **(b)** Beschleunigte MEEMD

Abbildung 5.17: MEEMD: Klassisches Verfahren und Vorschlag zur Reduzierung des Rechenaufwandes. Rot, Gelb und Grün sind die zu verrechnenden Teil-IMFs für die IMFs 1-3, Schwarz das Residuum. Blau und Türkis sind Ursprungsdatensatz und Zwischenergebnisse, die weiter zerlegt werden.

5.4.2 Verallgemeinerung für *D* Dimensionen

Bei der Verallgemeinerung für D Dimensionen ist zu beachten, dass die Einsparung nicht entlang jeder Dimension vorgenommen werden kann, sondern nur entlang der letzten. Das heißt, für die ersten $D-1$ Dimensionen verläuft der Prozess wie gewohnt - mit einer Ausnahme: Eine Teil-IMF 1 muss nie weiter zerlegt werden. Alle Teile, in die sie zerlegt werden kann, sind sowieso Bestandteil der Ergebnis-IMF 1. Dass dies bei keiner anderen IMF-Zerlegung möglich ist, liegt daran, dass hier Teil-IMFs für mindestens eine weitere IMF bestimmt werden müssen. Dies ist erst möglich, wenn das Ausgangssignal entlang der vorherigen Dimension so weit zerlegt ist wie gewünscht, da die Bestandteile ansonsten in dem (noch) kumulierten Signal „versteckt“ wären. Bei der letzten Dimension kann jedoch, ebenso wie im 2D-Fall, ein großer Teil der Berechnungen gespart werden, wie in Abbildung 5.18 exemplarisch für den dreidimensionalen Fall gezeigt ist.
Das Prinzip ist dabei das gleiche wie im 2D-Fall: Wenn bei der EMD entlang der letzten Dimension die nachfolgenden Teil-IMFs allesamt zur selben Ergebnis-IMF gehören, müssen diese nicht alle einzeln bestimmt werden. Wieviele jeweils be-



(a) Klassische MEEMD **(b)** Beschleunigte MEEMD

Abbildung 5.18: MEEMD: Klassisches Verfahren und Vorschlag zur Reduzierung des Rechenaufwandes. Rot, Gelb und Grün sind die zu verrechnenden Teil-IMFs für die IMFs 1-3, Schwarz das Residuum. Blau und Türkis sind Ursprungsdatensatz und Zwischenergebnisse, die weiter zerlegt werden.

stimmt werden müssen, ergibt sich aus den zuvor getätigten Zerlegungen. Nutzt man während der Zerlegung das Benennungsschema wie in Abbildung 5.17a, erhält jede Teil-IMF eine Indizierung (zB. Teil-IMF 3.2). Der kleinste der Indizes ist dann die Anzahl der zu berechnenden IMFs für diese Teil-IMF entlang der letzten Dimension.

5.4.3 Abschätzung der Beschleunigung und Beurteilung

Da für die letzte Dimension nur etwa $1/D$ der Teil-IMFs bestimmt werden müssen, kann eine deutliche Beschleunigung der Berechnungsdauer erzielt werden. Für den klassischen Ansatz ergibt sich der Rechenaufwand $R_{orig}(n, D)$ für die Bestimmung von n IMFs und D Dimensionen durch Gleichung 5.6:

$$R_{orig}(n, D) \sim \sum_{i=1}^{D} n^i \tag{5.6}$$

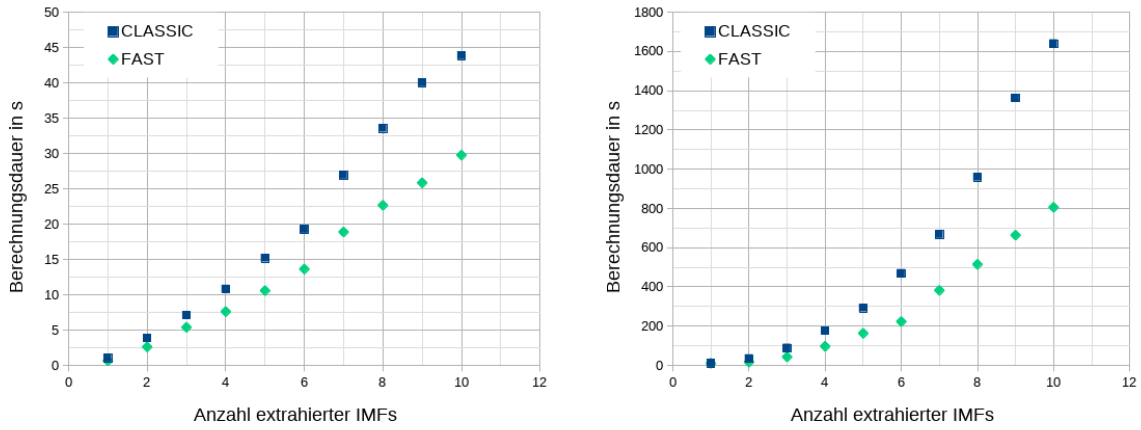
Der Rechenaufwand kann durch den vorgeschlagenen Ansatz auf etwa $1/D$ des ursprünglichen reduziert werden, wie basierend auf Abbildung 5.17 und Abbildung 5.18 abschätzbar ist. Der neue Aufwand ergibt sich somit nach folgender Gleichung:

$$R_{neu}(n, D) \approx \frac{1}{D} R_{orig}(n, D) \sim \frac{1}{D} \sum_{i=1}^{D} n^i \tag{5.7}$$

Eine Erhöhung der Anzahl IMFs n zieht allerdings immer noch eine Vervielfachung des Ergebnisses nach sich, der höchste Exponent in der Summe ist dabei D. Zu beachten ist zusätzlich, dass das Bestimmen der IMFs mit hohen Frequenzen oft länger dauert, da es bei diesen mehr lokale Extrempunkte und somit auch mehr Punkte für die Interpolation bei der Bestimmung der Einhüllenden gibt.
Auch wenn das beschriebene Verfahren somit eine deutliche Verbesserung zum ursprünglichen darstellt, kann es die Probleme der MEEMD nicht vollends lösen. Der Rechenaufwand steigt immer noch stark mit der Dimension und der Anzahl IMFs. Hinsichtlich des Artefakten-Problems wird keine Verbesserung erzielt.

5.4.4 Ergebnisse und Vergleich

Der Optimierungsansatz ändert - wie zuvor beschrieben - fast nichts an den Ergebnissen selbst, sondern bestimmt diese nur schneller. Das erlaubt allerdings wiederum bei gleichem Zeitbedarf mit größerer Ensemble-Größe zu arbeiten und so verbesserte Ergebnisse zu erzielen. Abbildung 5.19 zeigt im direkten Vergleich die benötigten Rechenzeiten für einen zwei- und einen dreidimensionalen Datensatz. Die Beschleunigung ist deutlich zu erkennen, sowohl für den zweidimensionalen wie auch dreidimensionalen Fall steigt die benötigte Zeit bei zunehmender Anzahl IMFs deutlich langsamer an als bei der klassischen MEEMD. Die theoretische Beschleunigung wird hier allerdings nicht erreicht: Für den zweidimensionalen Fall wird die Berechnung in ungefähr $2/3$ der Zeit abgeschlossen, für den dreidimensionalen Fall wird etwa die doppelte Berechnungsgeschwindigkeit erreicht. Der Grund hierfür ist mutmaßlich darin zu sehen, dass nur Teile der Berechnung parallelisiert sind und die nicht parallelisierten Teile stärker ins Gewicht fallen.



(a) 2D **(b)** 3D

Abbildung 5.19: Vergleich der Rechendauer von MEEMD (CLASSIC) und dem hier vorgestellten Verfahren (FAST) für 2D und 3D Daten

5.5 Implementierung der Fast and Adaptive Multivariate and Multidimensional EMD (FA-MVEMD)

Um die auftretenden Artefakte zu vermeiden und der hohen Berechnungsdauer entgegen zu wirken, wurde die MATLAB-Bibliothek mit der Implementierung der *Fast and Adaptive Multivariate and Multidimensional EMD* [42] (FA-MVEMD) in Betracht gezogen. Wie in Abschnitt 5.1.5 beschrieben, werden der rechenintensive Interpolationsschritt gespart und zusätzlich alle Dimensionen gleichzeitig verarbeitet. Ein positiver Nebeneffekt der Nutzung dieser Bibliothek ist, dass es sich um eine multivariate Implementierung handelt und mehrere Variablen gleichzeitig betrachtet werden können. Dies ist bei Geschwindigkeitsfeldern durchaus interessant. Denn die einzelnen Geschwindigkeitskomponenten als Variablen bilden gemeinsam einen Geschwindigkeitsvektor.

5.5.1 Möglichkeiten zur Einbindung von MATLAB-Code in C++

Die FA-MVEMD-Bibliothek ist in MATLAB geschrieben, stellt aber gleichzeitig die einzige frei verfügbare Implementierung eines *Fast and Adaptive*-Ansatzes dar. Daher muss eine Möglichkeit gefunden werden, diesen Code in der in C++ geschriebenen Toolbox zu verwenden. MATLAB selbst stellt verschiedene Möglichkeiten zur Integration von MATLAB-Code in ein C++-Projekt zur Verfügung. Dazu kann die

Funktionalität von MATLAB durch verschiedene Toolboxen erweitert werden. Für den Anwendungsfall der Anbindung von MATLAB-Funktionen in ein C++-Programm kommen die Toolboxen *MATLAB Coder* [40] und *MATLAB Compiler SDK* [41] in Frage. *MATLAB Coder* generiert aus einem MATLAB-Programm C++-Code, die *MATLAB Compiler SDK* ist hingegen dazu da, Closed-Source Exporte vorzunehmen - also die Generierung von ausführbaren Programmen und Bibliotheken. Da letztlich beide Toolboxen eingesetzt wurden, werden nachfolgend die Einschränkungen und Einsatzzwecke kurz dargestellt.

	MATLAB Compiler SDK	**MATLAB Coder**
Output	Shared Libraries (.so)	Portabler C-Quellcode (.c & .h), MEX-Dateien
Unterstützung der MATLAB-Sprache	komplett	Teilmenge
Unterstützung von MATLAB-Toolboxen	fast alle	wenige
Abhängigkeit von Bibliotheken	*MATLAB Runtime*	keine
Zielplattformen	von MATLAB unterstützte (Windows, Mac, Linux)	jede Plattform, die C/C++ unterstützt
Cross-Plattform Kompatibilität	X	✓
Geschwindigkeit	wie der MATLAB-Code	potentiell schneller
Arbeitsspeicherbedarf	erhöht (MATLAB Runtime)	nur durch Daten verursachter

Tabelle 5.2: Vergleich von *MATLAB Compiler SDK* und *MATLAB Coder* im Hinblick auf den Zugriff auf MATLAB-Programme aus einem C- oder C++-Programm, nach [39]

MATLAB Coder Wie bereits geschrieben, übersetzt *MATLAB Coder* ein Programm in C++-Code. Tabelle 5.2 zeigt das größte Problem dabei auf: Die Übersetzung ist nicht für alle MATLAB-Funktionen verfügbar. Zudem ist der generierte Code - wie bei fast jeder automatischen Code-Erstellung - nicht sonderlich übersichtlich. Da zusätzlich auch die logische Struktur des generierten Programmes abweicht, kann dieses Problem auch nicht durch Auskommentieren der betreffenden Funktionen und eigenhändiges Einbinden passender C++-Bibliotheken gelöst werden. Ist eine

Funktion hingegen übersetzbar, so kann die Nutzung von MATLAB-Coder sogar innerhalb eines reinen MATLAB-Projektes empfehlenswert sein, denn es können auch so genannte *MEX* Funktionen generiert werden. Das sind C- oder C++-Funktionen, die aus einem MATLAB-Skript aufgerufen werden können. Da dieser Code im Gegensatz zu MATLAB-Code kompiliert ist, kann eine deutliche Beschleunigung erzielt werden.

MATLAB Compiler SDK Die Nutzung der *MATLAB Compiler SDK* zur Einbindung der Funktionen in C++ sieht grundlegend anders aus. Hier wird lediglich eine Schnittstelle geschaffen, über welche auf den MATLAB-Code zugegriffen werden kann. Im Hintergrund wird die *MATLAB Runtime* gestartet, in der die Funktionen dann ausgeführt werden. Auf dem Rechner, auf dem das Programm genutzt werden soll, muss daher die *MATLAB Runtime* installiert sein. Diese kann immerhin auch ohne MATLAB-Lizenz genutzt werden und kostenfrei auf der MATLAB-Website heruntergeladen werden. Im Gegensatz zu tatsächlich übersetztem Code kann mit dieser Methode kein Geschwindigkeitsgewinn erzielt werden. Dafür können so gut wie alle Funktionen genutzt werden. Da das Starten der *MATLAB Runtime* eine längere Zeit benötigt, wird empfohlen, dies nicht für jeden einzelnen Bibliotheksaufruf durchzuführen. Stattdessen sollte sie gleichzeitig mit dem Programm, in dem sie genutzt wird, gestartet und beendet werden. Es können Bibliotheken mit unterschiedlichen Schnittstellen generiert werden, über die auf die MATLAB-Funktionalität zugegriffen werden kann: Einerseits ist die Nutzung der *mwarray API* möglich, andererseits kann auch die neuere *MATLAB Data API* eingebunden werden. Letztere sollte bevorzugt verwendet werden, denn sie bietet unter anderem Support neuerer C++-Funktionalitäten (C++11 vs C++03) sowie Typ- und Thread-Sicherheit.

5.5.2 Einbindung der Bibliothek

In der FA-MVEMD werden Funktionen genutzt, die für *MATLAB Coder* nicht zur Verfügung stehen. Aus diesem Grund wird die *MATLAB Compiler SDK* genutzt. Die Bibliothek gibt MATLAB-spezifische Datentypen vor, in welchen die Daten als Eingabe für die Bibliotheksaufrufe vorliegen müssen. Die Übersetzung in diese wird durch die Einführung eines Adapters gelöst, durch den die Bibliotheksaufrufe ausgeführt werden. Dadurch kann im Rest des Programms unverändert mit der bereits definierten Datenorganisation gearbeitet werden.

Ausführung der EMD Durch die Bibliothek sind für verschiedene Anwendungsfälle mehrere Funktionen definiert, je nachdem ob der Datensatz zwei- oder dreidimensional ist und ob zwei oder drei Variablen prozessiert werden sollen: EMD2D2V, EMD2D3V, EMD3D2V und EMD3D3V. Folglich ist die Anzahl an Variablen, die gleichzeitig betrachtet werden können auf zwei oder drei limitiert. In Geschwindigkeitsfeldern können die Variablen meist in Gruppen passender Größe eingeteilt werden. Ein Beispiel dafür ist die Definition einer Gruppe mit den Geschwindigkeitskomponenten der drei Raumrichtungen und einer zweiten für die Beschleunigungskomponenten. Daher stellt diese Einschränkung kein schwerwiegendes Problem dar. Allerdings kann das Programm die Entscheidung über die Zusammenstellung der Gruppen nicht automatisch treffen, weil die einzige verfügbare Information dazu der Name der Variable ist. Dieser kann von Datensatz zu Datensatz unterschiedlich sein. Es kann aber zumindest ein „Vorschlag“ über die Zusammensetzung der Gruppen gemacht werden, letztenendes müssen die Gruppen aber von dem Anwender des Programms ausgewählt oder bestätigt werden (vgl. Abbildung 5.20). Der Vorschlag wird basierend auf der erkannten Dimensionalität des Datensatzes (bestimmt durch die IJK Informationen) in Kombination mit der Anzahl der verfügbaren Variablen getroffen. Handelt es sich beispielsweise um einen dreidimensionalen Datensatz mit sechs Variablen ist es wahrscheinlich, dass die Variablen vier bis sechs die Geschwindigkeitskomponenten sind und daher gemeinsam prozessiert werden sollten. Abbildung 5.20 zeigt auch die weiteren Einstellmöglichkeiten für die FA-MVEMD. Dies sind die in Abschnitt 5.1.5 vorgestellten Fenstertypen für den gleitenden Mittelwert, ein Toleranzwert als Abbruchkriterium und natürlich die Anzahl zu extrahierender IMFs.

Aufteilen der IMF-Extraktion in mehrere Bibliotheksaufrufe Die Bibliothek ist so gestaltet, dass die gesamte EMD-Berechnung mit einem einzigen Bibliotheksaufruf ausgeführt werden kann. Dies führt dazu, dass für die Dauer der Berechnung weder der Fortschritt angezeigt, noch die Berechnung abgebrochen werden kann. Um dies zu verhindern, kann die Berechnung in mehrere Teilrechnungen aufgespalten werden, wozu der iterative Charakter der EMD ausgenutzt wird. Das Bestimmen von n IMFs geschieht letztlich dadurch, dass n-mal eine IMF bestimmt wird und das Verfahren immer wieder auf das Residuum angewandt wird. Ob dies nun bibliotheksintern geschieht oder außerhalb, hat keine Auswirkung auf das Endergebnis.

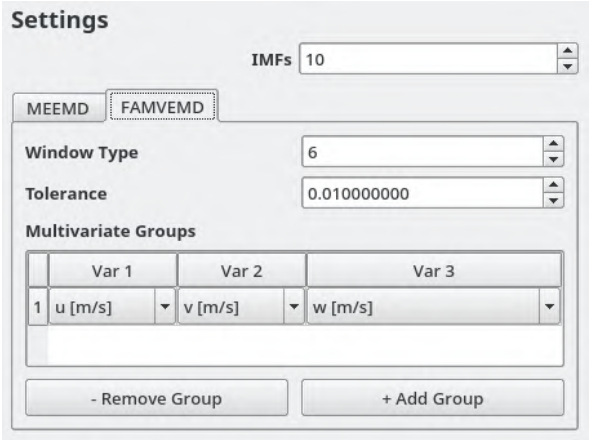


Abbildung 5.20: Ausschnitt des Dialogs zum Starten der FA-MVEMD

Pro Bibliotheksaufruf wird daher immer nur eine IMF angefordert, die Rückgabe ist dann die IMF und das Residuum. Das Residuum wird anschließend als Eingabe für den nächsten Bibliotheksaufruf genutzt. Dadurch ist dieser Teil der EMD-Logik aus der MATLAB-Bibliothek in den C++-Code verschoben worden. Das hat mehrere Vorteile:

1. Nach jedem Bibliotheksaufruf kann der Fortschritt angezeigt und abgebrochen werden
2. Die IMF kann dann direkt persistiert werden, sodass im Falle eines Abbruchs zumindest die Ergebnisse bis zu diesem Punkt erhalten bleiben.
3. Der Arbeitsspeicherbedarf ist geringer, da nicht für alle IMFs gleichzeitig die Daten von der Bibliothek zurückgegeben werden

Analyse der MATLAB-Bibliothek und Beschleunigung Führt man ein Programm in MATLAB aus, kann man direkt verfolgen lassen, welche Zeile des Programms wie oft aufgerufen wurde und wie groß der Zeitbedarf pro Funktion ist. Mithilfe dieser Analyse können sehr schnell zeitkritische Codeausschnitte gefunden werden. In Tabelle 5.3 ist dies für die Berechnung eines dreidimensionalen Datensatzes mit $100 * 100 * 100$ Punkten und drei Variablen bei der Bestimmung einer einzigen IMF gezeigt. Dabei sind - sortiert nach Gesamtzeit - nur die ersten acht Zeilen dargestellt. Es ist zu erkennen (rot markiert), dass der Großteil der Zeit, die für die Berechnung der EMD benötigt wird, durch die Funktion *Ordfilt1* verantwortet wird. Die Funktion wird von *Separable_ordfilt3* aufgerufen, die wiederum von *OSF* aufgerufen wird. *OSF* steht für *Order Statistics Filter*. Ein solcher Filter kann für gleitende Minimum-,

Funktionsname	Aufrufe	Gesamtzeit	Eigenzeit
driver_3d_example	1	95,389 s	0,114 s
EMD3D3V	1	95,269 s	0,086 s
EMD3D3V>OSF	4	93,301 s	0,007 s
EMD3D3V>OSF/Separable_ordfilt3	24	93,295 s	3,241 s
EMD3D3V>OSF/Separable_ordfilt3/Ordfilt1	720.000	90,054 s	88,317 s
squeeze	720.000	1,736 s	1,736 s
EMD3D3V>Pad_smooth	4	1,109 s	0,785 s
MinimaMaxima3D	1	0,533 s	0,356 s
...	...	...	...

Tabelle 5.3: Zeitbedarf und Anzahl der Aufrufe für (Teil-) Funktionen einer 3D MATLAB FA-MVEMD Berechnung

Maximum- oder Medianbestimmung genutzt werden. Dazu wird immer ein „Fenster“ der Größe n, also ein Teil des Datensatzes betrachtet, aufsteigend sortiert und alle Elemente auf den Wert des Elementes an Position i gesetzt. Bei $i = 1$ erhält man so beispielsweise einen Minimumfilter, bei $i = n$ ein Maximumfilter. Für die mehrdimensionale Ausprägung kann ein solcher Filter so umgesetzt werden, dass immer nur eine Dimension betrachtet wird, weshalb er dann die Bezeichnung *separierbar* bekommt. Auch für die *Fast and Adaptive EMD* wurde die Nutzung von separierbaren *OSFs* vorgeschlagen [13], um den gleitenden Mittelwert schneller bestimmen zu können. Das Verfahren erklärt die hohe Zugriffszahl für die Funktion *Ordfilt1* und motiviert dazu, diese genauer zu betrachten (siehe Quellcode 5.1). Die hervorgehobenen Zeilen bieten dabei das größte Optimierungspotenzial:

- An den Rändern werden Elemente gespiegelt und angefügt (in Zeile 8). Dieser aufwändige Schritt kann vermieden werden, da nur Werte gespiegelt werden, die ohnehin im betrachteten Bereich liegen. Es kann also durch das Spiegeln kein neues Minimum oder Maximum auftreten.
- Für die Bestimmung von Minimum und Maximum ist eine Sortierung nicht der effizienteste Weg (in den Zeilen 17-18 und 21-22). Stattdessen können die *min*- und *max*-Funktion von MATLAB genutzt werden.

```
function f_signal = Ordfilt1(signal,order,window_size) % 1-D Rank order filter
    % Pre-processing
    [a,b,c] = size(signal); % Original signal size
    signal = squeeze(signal); % Removing the singleton dimensions
    L = length(signal); % Length of the signal
    signal = reshape(signal, [L,1]); % ensure signal is a column vector
    r = (window_size-1)/2;
    x = [flip(signal(1:r)); signal; flip(signal(end-(r-1):end))]; % Padding
    [M,~] = size(x);
    y = zeros(size(x));
    % Filter
    switch order
        case 'max'
            for m = 1+r:M-r
                temp = x((m-r):(m+r)); % Extract a window (size 2r+1) around m
                w = sort(temp);
                y(m) = w(end); % Select the greatest element
            end
        case 'min'
            for m = 1+r:M-r
                temp = x((m-r):(m+r)); % Extract a window (size 2r+1) around m
                w = sort(temp);
                y(m) = w(1); % Select the smallest element
            end
        otherwise
            error('No such filering operation defined')
    end
    % Post-Processing
    f_signal = y(1+r:end-r);
    f_signal = reshape(f_signal,[a,b,c]); % Restoring Signal size
end
```

Quellcode 5.1: Ursprüngliche Version der *Ordfilt1*-Funktion (aus [42], hinsichtlich der Darstellung abgeändert)

Durch die Überarbeitung der entsprechenden Stellen kann der Code vereinfacht und beschleunigt werden, wie in Quellcode 5.2 zu sehen ist. Die Einfärbung ist dabei analog zu dem vorherigen Codeausschnitt und spiegelt wieder, welche Änderungen zusammengehören. Dennoch beansprucht die Funktion nach wie vor mit Abstand den größten Anteil der Berechnungsdauer, da die benötigte Zeit lediglich halbiert wird (siehe Abbildung 5.21). Daher lohnt es sich, eine weitere Beschleunigungsmaßnahme vorzunehmen. An dieser Stelle kommt der *MATLAB Coder* zum Einsatz.

Dies ist möglich, da alle innerhalb von *Ordfilt1* genutzten Funktionen in C-Code übersetzbar sind. Abbildung 5.21 zeigt, dass durch die Nutzung der generierten *MEX*-Funktion noch einmal eine deutliche Beschleunigung erzielt werden kann. Ohne Frage lässt sich in dem MATLAB-Code noch weiteres Optimierungspotenzial finden, da jedoch der größte Einflussfaktor korrigiert ist und das Verfahren selbst bereits eine schnelle Prozessierung ermöglicht, ist dies nicht von hoher Priorität.

```
function f_signal = Ordfilt1(signal,order,window_size) % 1-D Rank order filter
    L = length(signal); % Length of the signal
    f_signal = zeros(size(signal));
    r = (window_size-1)/2;
    switch order % Filter
        case 'max'
            for m= 1:L
                lower_bound = max(m-r, 1); % Ensure index is in bounds
                upper_bound = min(m+r, L); % Ensure index is in bounds
                f_signal(m) = max(signal(lower_bound:upper_bound));
            end
        case 'min'
            for m= 1:L
                lower_bound = max(m-r, 1); % Ensure index is in bounds
                upper_bound = min(m+r, L); % Ensure index is in bounds
                f_signal(m) = min(signal(lower_bound:upper_bound));
            end
        otherwise
            error('No such filering operation defined')
    end
end
```

Quellcode 5.2: Optimierte *Ordfilt1*-Funktion

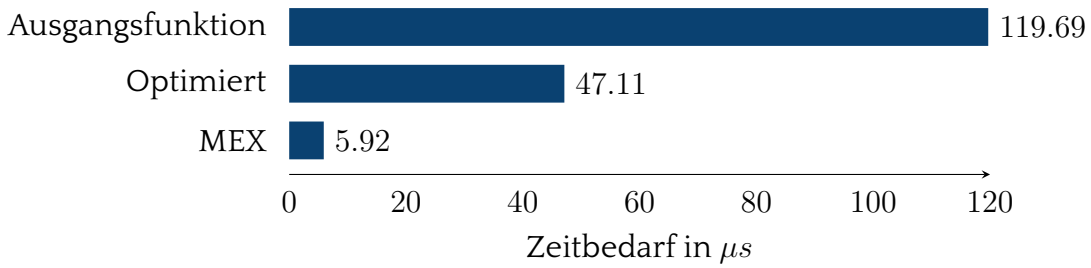
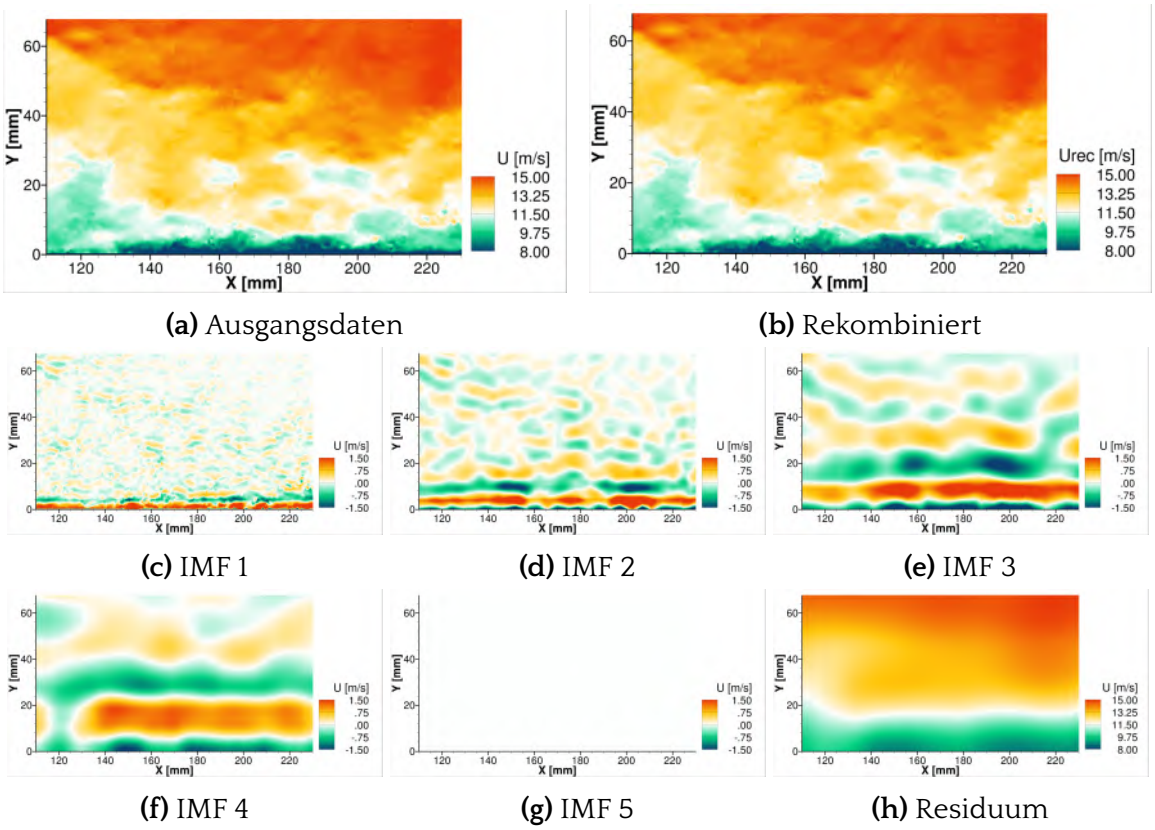


Abbildung 5.21: Durchschnittlicher Zeitbedarf für Varianten der Funktion *Ordfilt1*

5.5.3 Qualitätsuntersuchungen

Nach erfolgreicher Einbindung muss sich nun zeigen, dass sich die an das Verfahren gestellten Erwartungen erfüllen. Zunächst ist in Abbildung 5.22 eine Zerlegung mit der FA-MVEMD dargestellt. Es handelt sich bei dem betrachteten Datensatz um ein 2D-Geschwindigkeitsfeld mit drei Geschwindigkeitskomponenten. Für die IMFs ist erneut ein kleinerer Wertebereich für die Einfärbung genutzt worden als bei dem Ausgangsdatensatz, dem aus den IMFs rekombinierten Datensatz und dem Residuum. Aus Gründen der Übersichtlichkeit werden nicht alle Geschwindigkeitskomponenten gezeigt, sondern nur u. Als Parameter wurden der Fenstertyp 5 und ein Toleranzwert von 0.01 gewählt. Die erzielten Ergebnisse sehen strömungsmechanisch plausibel aus: Es werden in Wandnähe kleine Strukturen mit hoher Amplitude gefunden, größere sowie schwächere Strukturen finden sich vor allem in größerem Abstand zur Wand.



(a) Ausgangsdaten **(b)** Rekombiniert

(c) IMF 1 **(d)** IMF 2 **(e)** IMF 3

(f) IMF 4 **(g)** IMF 5 **(h)** Residuum

Abbildung 5.22: Zerlegung eines Geschwindigkeitsfeldes mit der FA-MVEMD in fünf IMFs und Residuum (Fenstertyp 5, Abbruchkriterium μ = 0.01)

Positiv hervorzuheben ist auch, dass die wiederhergestellten Daten eine hohe Übereinstimmung mit den Ausgangsdaten zeigen. Abbildung 5.23 zeigt, dass die Abweichungen im Bereich von 10^{-7} m/s liegen. Auffällig ist, dass in der fünften IMF bereits keinerlei Information mehr enthalten ist, obwohl in dem Residuum noch Schwankungen zu erkennen sind. Das gilt auch für die nachfolgenden IMFs, die deswegen nicht dargestellt sind. Die Gründe hierfür sollen nachfolgend herausgearbeitet werden. Von den beiden Parametern der FA-MVEMD wird insbesondere der Einfluss des Fenstertyps betrachtet.

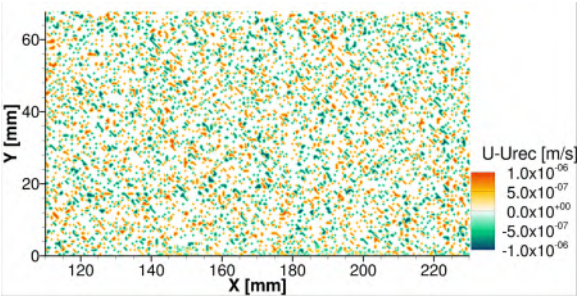


Abbildung 5.23: Differenz von Ausgangsdatensatz (Abbildung 5.22a) und rekombinierten Geschwindigkeitsfeld (Abbildung 5.22b)

Einfluss des Fenstertyps Mit der Wahl der unterschiedlichen Fenstertypen hat der Anwender einen Einfluss auf die Größe der Strukturen, die extrahiert werden. Da die EMD diese Information eigentlich rein datenbasiert erarbeiten soll (wie es bei der MEEMD der Fall ist), muss untersucht werden, inwiefern die Fenstertypen das Ergebnis beeinflussen. Die Ergebnisse bei Anwendung der Fenstertypen 1 und 2 sind fast immer identisch, selbiges gilt für die Fenstertypen 3 und 4. Bei einem Blick auf die Definition der Fenstertypen ist dies wenig verwunderlich: In einem großen Datensatz ist es nicht unwahrscheinlich, das die kleinste Distanz zwischen Minima ähnlich der zwischen Maxima ist - und das selbe gilt für die größte Distanz. Dadurch fallen diese Fenstertypen zu einem einzigen Fall zusammen und werden nachfolgend gemeinsam aufgeführt. Die in Abbildung 5.24 dargestellten Ergebnisse zeigen die starke Abhängigkeit von dem Fenstertyp. Dabei wurden jeweils zehn IMFs bestimmt. Es ist zu erkennen, das für die Fenstertypen 1 und 2 das Residuum noch sehr stark dem Ausgangsdatensatz (Abbildung 5.22a) ähnelt. Bei den Fenstertypen 3 und 4 treten hingegen in der zweiten IMF bereits sehr große Strukturen auf. Das deutet darauf hin, dass kleinere Strukturen bei der Extraktion einfach übergangen

werden und stattdessen in IMFs mit größeren Strukturen enthalten sind. Bei Typ 7 ist das Residuum wiederum noch wenig gleichmäßig, weshalb die Fenstertypen 5 und 6 am geeignetsten erscheinen.

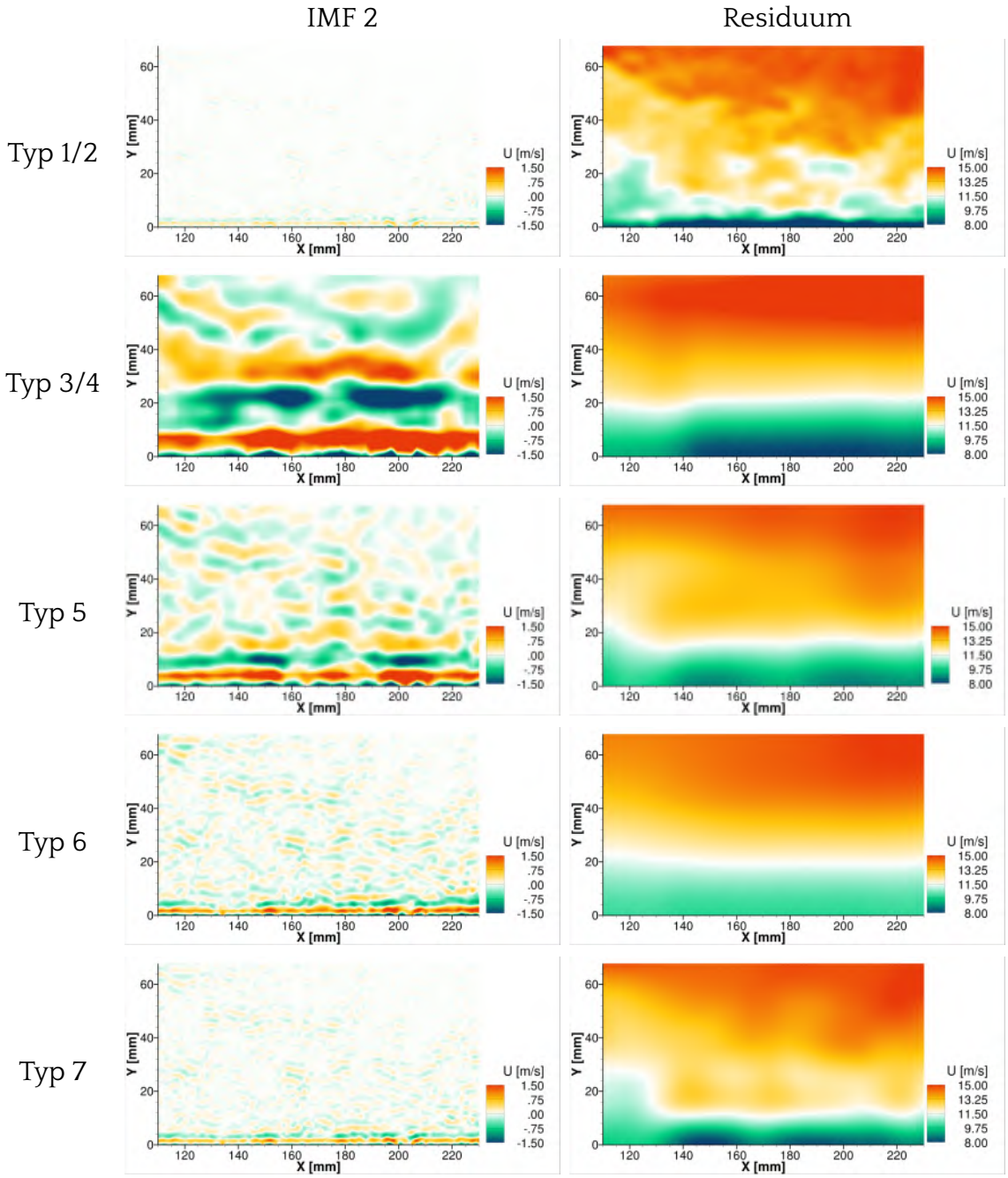


Abbildung 5.24: Untersuchung der FA-MVEMD Fenstertypen: IMF 2 und Residuum nach zehn IMFs

Verarbeitung multivariater Signale Das Ergebnis der Verarbeitung eines multivariaten Signals ist in Abbildung 5.25 dargestellt. Dabei handelt es sich zusätzlich um ein dreidimensionales Geschwindigkeitsfeld, dargestellt sind Schnitte durch dieses an verschiedenen Positionen. Es ist gut zu erkennen, dass die Größe der Strukturen in den jeweiligen IMFs ähnlich ist, die Amplituden jedoch stark unterschiedlich sind. Das Ergebnis erscheint plausibel, da die u-Komponente die Geschwindigkeit

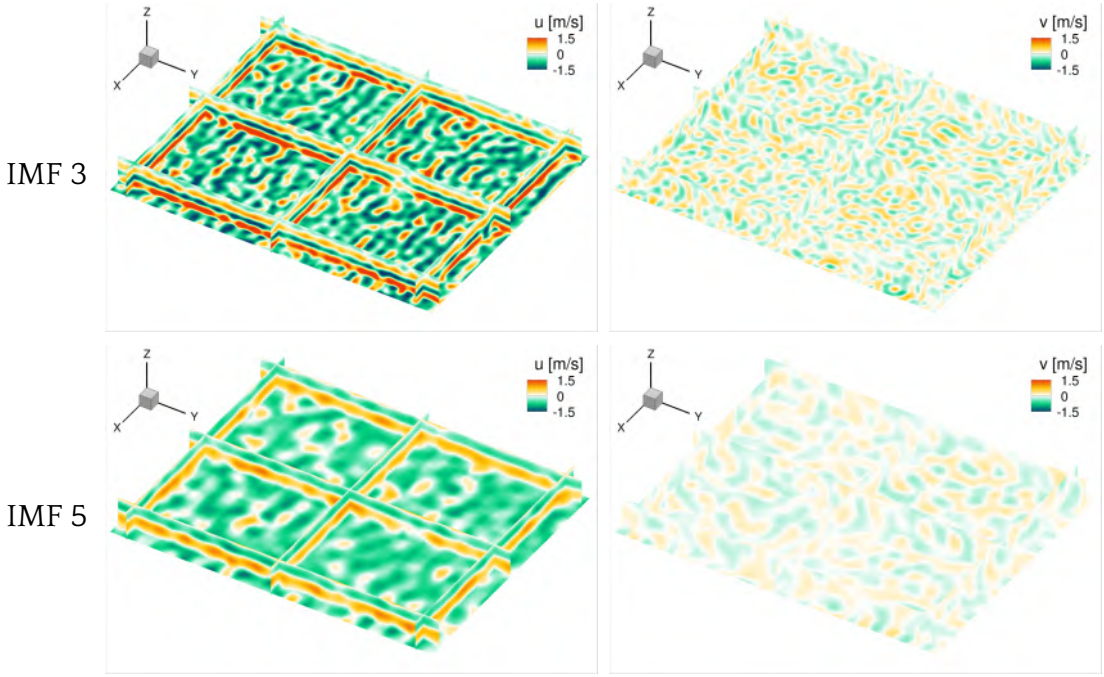


Abbildung 5.25: IMF 3 und 5, Geschwindigkeitskomponenten u und v gemeinsam analysiert

in Strömungsrichtung ist. Damit weist u insgesamt die höchsten Geschwindigkeiten auf, was sich letztlich auch in den Amplituden in den IMFs zeigt.
Wünschenswert wäre es, im Vergleich die Geschwindigkeitskomponenten einzeln zu analysieren. Allerdings kann die Bibliothek nur bi- und trivariate Signale verarbeiten - was mit einem Trick umgangen werden kann: Es wird die bivariate Variante ausgeführt, allerdings wird zweimal die selbe Variable übergeben. Dadurch können zum Vergleich die in Abbildung 5.26 dargestellten IMFs für die (jetzt unabhängig von einander in IMFs zerlegten) Komponenten u und v erzeugt werden. Der Unterschied zu Abbildung 5.25 ist gut zu erkennen: Die IMFs der Komponente v weichen stark ab. So ist die Größe der Strukturen zwischen den beiden Komponenten nicht identisch,

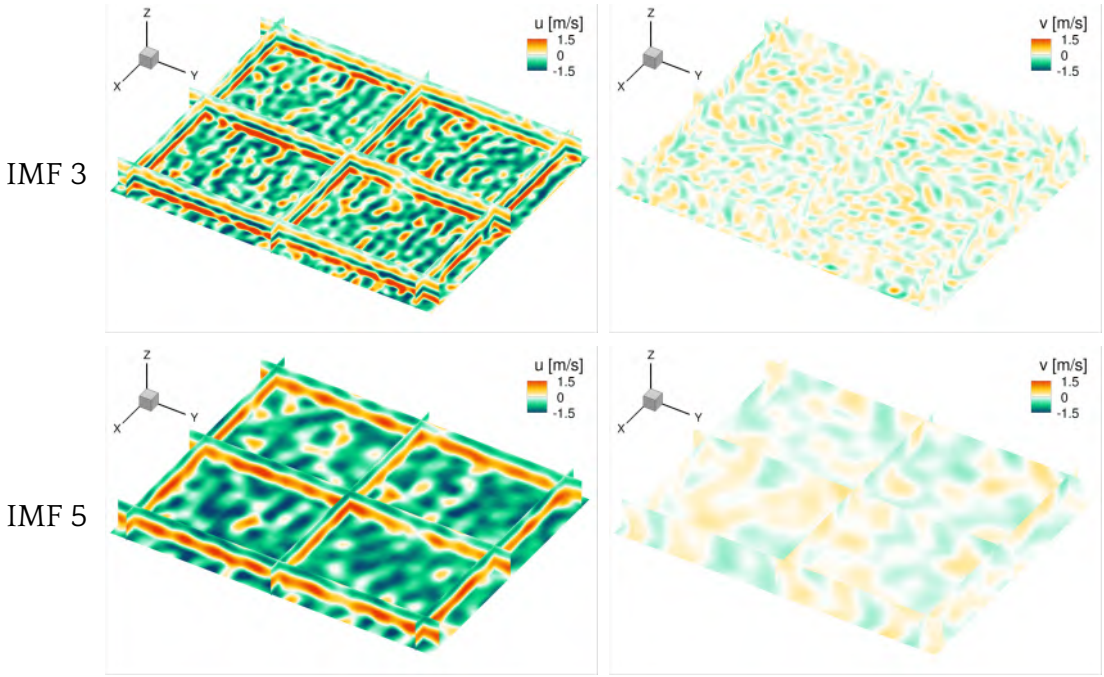
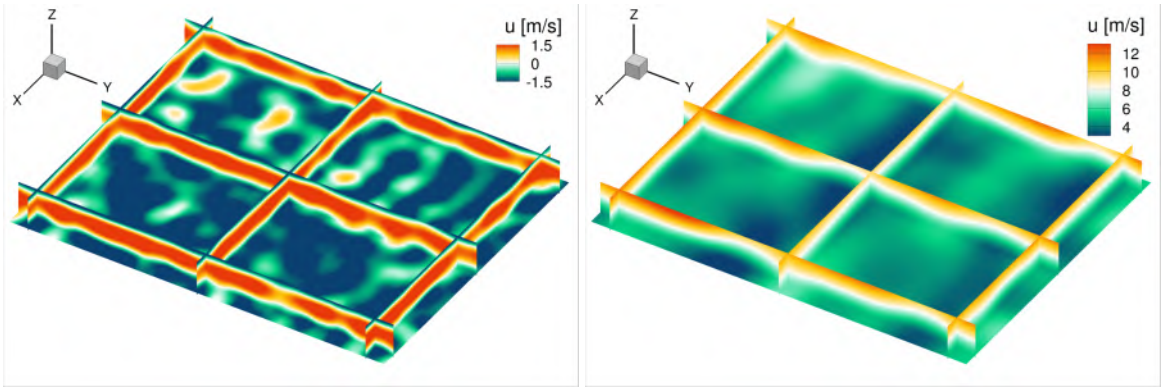


Abbildung 5.26: IMF 3 und 5, Geschwindigkeitskomponenten u und v einzeln analysiert

was insbesondere bei IMF 5 deutlich wird. Da die einzelnen Komponenten jedoch zu einer einzigen Bewegung gehören, sollten sie sich ähneln - ansonsten können in den IMFs physikalisch nicht erklärbare Änderungen der Richtungen der Geschwindigkeitsvektoren auftreten (die einen rein analytischen Ursprung hätten). Das zeigt, dass die multivariate Verarbeitung bei der Analyse der Geschwindigkeitsfelder einen Mehrwert bietet.

Vorzeitiger Abbruch der IMF-Extraktion Bereits in den zuvor gezeigten Daten ist zu erkennen, dass das Residuum noch Strukturen enthält, selbst wenn keine IMFs mehr extrahiert werden können (vgl. Abbildung 5.22). Abbildung 5.27 zeigt dies nun - noch stärker ausgeprägt - bei einem dreidimensionalen Datensatz. In diesem Fall konnten sechs IMFs extrahiert werden, alle nachfolgenden enthalten keine Information mehr. Die naheliegende Vermutung für den vorzeitigen Abbruch der IMF Extraktion ist, dass der Toleranzwert μ zu hoch gewählt wurde. Dies ist aber nicht der alleinige Grund dafür, dass in den Residuen noch Strukturen zu erkennen sind. Die Betrachtung der Ergebnisse legt eine weitere Vermutung nahe: Die maximale Größe der extrahierten Strukturen wird durch die kleinste Dimension des Daten-



(a) IMF 6 **(b)** Residuum

Abbildung 5.27: Anwendung der FA-MVEMD: Letzte extrahierte IMF und Residuum

satzes limitiert. Sobald in einer Dimension nicht mehr genügend Extrempunkte vorhanden sind, wird die Extraktion von IMFs abgebrochen. Dies lässt sich mit dem Minimalbeispiel in Abbildung 5.28 verifizieren: Ein beispielhafter, zweidimensionaler Datensatz mit stark ungleichen Dimensionen enthält die Überlagerung zweier Sinusfunktionen (je in X- und Y-Richtung). Von dem Sinus in X-Richtung ist bei einer Fenstergröße der Datensatzlänge in Y-Richtung (in diesem Fall 100) immer nur maximal ein Extrempunkt sichtbar. Als Folge bleibt dieser Sinus im Residuum erhalten und wird nicht weiter extrahiert.

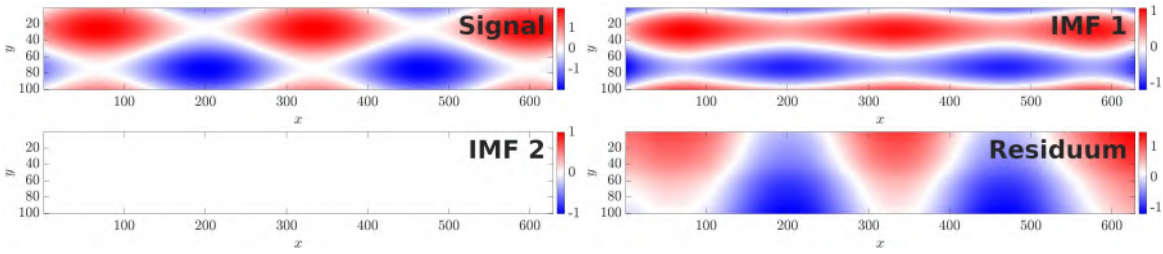


Abbildung 5.28: Anwendung der FA-MVEMD bei ungleichen Dimensionen

Für dieses Problem sind verschiedene Lösungsansätze denkbar, die nachfolgend diskutiert werden sollen. Es besteht die Notwendigkeit einer identischen Fenstergröße für alle Dimensionen, weil durch die Fenstergröße maßgeblich die Größe der extrahierten Strukturen beeinflusst wird. Für alle Dimensionen sollen aber Strukturen gleicher Größe in der jeweiligen IMF enthalten sein. Somit ist es nicht möglich, eine individuelle Fenstergröße pro Dimension zu definieren. Eine weit vielversprechendere Idee ist es hingegen, Fenster zuzulassen, die in ihrer Größe

die kleinste Dimension übertreffen. Damit kein Zugriffsfehler entsteht, muss eine Sonderbehandlung für Dimensionen, die kleiner als die Fenstergröße sind, eingeführt werden. Diese Sonderbehandlung ist identisch mit der Behandlung von Randpunkten und damit ohnehin bereits in der Funktion *Ordfilt1* implementiert. Die gemittelte Funktion hätte entlang dieser Dimension im Endeffekt einen konstanten Wert. Diese Anpassung muss innerhalb der MATLAB-Bibliothek umgesetzt werden und zieht dort umfangreichere Änderungen nach sich, weshalb sie bisher nicht umgesetzt werden konnte.
Ein weiterer Grund für einen vorzeitigen Abbruch der Extraktion ist, dass alle Extrempunkte kolinear sind, also auf einer Linie liegen. Die Delaunay-Triangulation, die ausgeführt wird um die Abstände zwischen den Punkten zu bestimmen, kann dann nicht ausgeführt werden. Für einen solchen Fall muss in die MATLAB-Bibliothek eine Spezialfall-Behandlung eingebaut werden, die dann die Entfernungen zwischen den jeweils benachbarten Extrempunkten ohne Triangulation bestimmt.

Abgleich mit MEEMD Da die MEEMD algorithmisch näher an der „echten" EMD liegt, sollten die mit ihr bestimmten IMFs als Referenz für die Korrektheit der FA-MVEMD genutzt werden. Da das Verfahren jedoch stark abweicht, ist nicht zu erwarten, dass die IMFs identisch sind, jedoch sollten die extrahierten Strukturen ähnlich aussehen. Für die FA-MVEMD kann dies mit dem Fenstertyp 6 in etwa erreicht werden, wie Abbildung 5.29 zeigt. Insbesondere bei den kleinen Strukturen ähneln sich die IMFs. Bei den späteren IMFs werden teilweise Strukturen erst später aufgelöst (der große Bereich mit hoher Geschwindigkeit im unteren Drittel ist bei der FA-MVEMD in der fünften, bei der MEEMD schon in der vierten IMF enthalten). Die anderen Fenstertypen erzielen deutlich weniger brauchbare Ergebnisse: Die Typen 1 und 2 liefern eigentlich immer zu kleine Fenstergrößen. Damit werden bei ihrer Verwendung immer nur Teile der Strukturen extrahiert und die Strukturen dadurch über mehrere IMFs gestreut. Die Nutzung der Typen 3 und 4 bedeutet hingegen fast immer eine zu große Fenstergröße. Daraus resultiert, dass kleinere Strukturen nicht in einzelne IMFs extrahiert werden, sondern in anderen enthalten sind. Bei der Nutzung von Typ 7 kann pro IMF eines der oben genannten Probleme auftauchen - oder gar keines, damit ist Typ 7 nur schwer anwendbar. Einzig bei Typ 5 können auch ähnliche Strukturen extrahiert werden, die Strukturen sind allerdings etwas größer (siehe dazu auch: Abbildung 5.22).

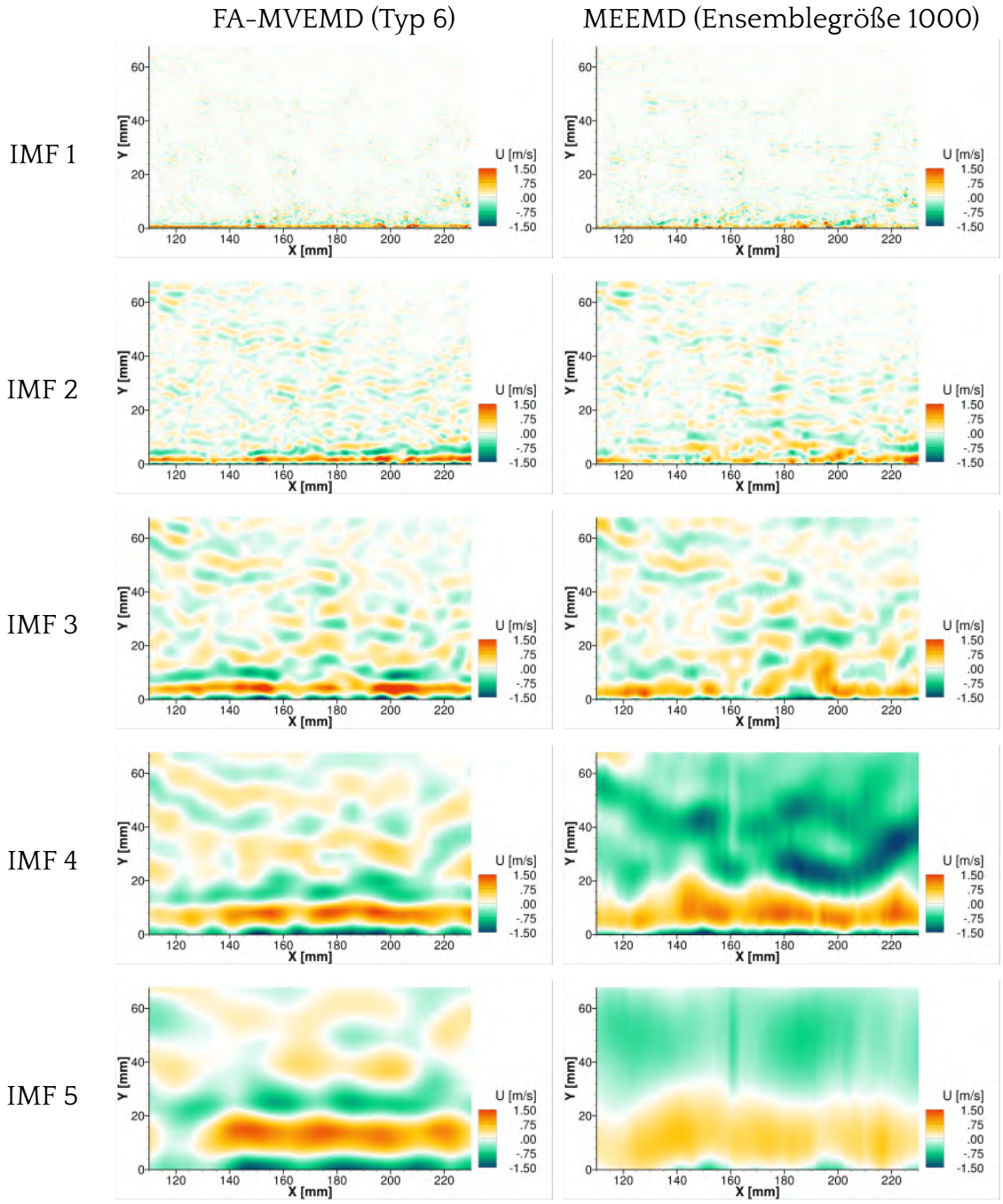


Abbildung 5.29: Vergleich von FA-MVEMD und MEEMD: IMFs 1-5

5.5.4 Zeitbedarf und Rechenaufwand

Ein direkter Vergleich mit der MEEMD hinsichtlich des Zeitbedarfs ist nur bedingt möglich, da die beiden Verfahren mit unterschiedlichen Parametern arbeiten, welche jeweils die Rechenzeit beeinflussen können. Insbesondere die unterschiedlichen Abbruchkriterien erschweren einen Vergleich hinsichtlich des Zeitbedarfs. Um dennoch eine ungefähre Abschätzung zu ermöglichem wird das folgende Vorgehen angewandt: Zunächst wird die FA-MVEMD für ein zwei- und ein dreidimensionales Geschwindigkeitsfeld angewandt. Das Geschwindigkeitsfeld verfügt in beiden Fällen über drei Variablen. Für die FA-MVEMD wird der Fenstertyp 6 und der Toleranzwert $\mu = 0.01$ gewählt, da hier die Ergebnisse als strömungsmechanisch plausibel eingestuft werden können. Anschließend wird betrachtet, wie viele IMFs extrahiert wurden und die selbe Anzahl an IMFs mit der MEEMD bestimmt. Dabei wird für die MEEMD eine Ensemble-Größe von 200 gewählt, da diese brauchbare Ergebnisse liefert und damit eine gewisse Vergleichbarkeit gegeben ist. Die weiteren Parameter für die MEEMD sind gewissermaßen „Standard“ (Stärke des Rauschens 0.2, S-Nummer 3, maximale Anzahl Sifting Iterationen 20). Für beide Fälle wurden letztlich 6 IMFs extrahiert. Der zweidimensionale Datensatz verfügt über $765 * 126$, der dreidimensionale Datensatz über $155 * 19 * 195$ Datenpunkte. Abbildung 5.30 zeigt die Berechnungsdauer für diese Fälle.

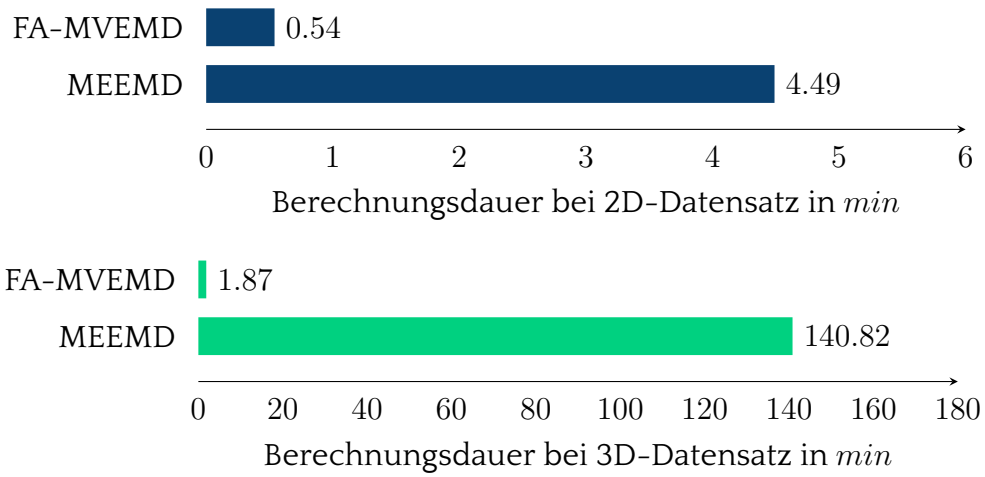


Abbildung 5.30: Vergleich der benötigten Zeit zur Extraktion von 6 IMFs für einen zwei- und dreidimensionalen Datensatz mit FA-MVEMD und MEEMD

Zweifellos ist die FA-MVEMD schneller. Für den zweidimensionalen Fall benötigt sie nur etwa $1/8$ der MEEMD-Zeit, im dreidimensionalen Fall sogar nur ein $1/75$. Hinzu kommt, dass die MEEMD bereits in dem in Abschnitt 5.4 beschriebenen, beschleunigten Verfahren angewandt wird. Zusätzlich ist sie in Teilen parallelisiert, sowie komplett in C++ geschrieben. Es sollte allerdings angemerkt werden, dass die FA-MVEMD bei größeren Abständen zwischen den Extrempunkten langsamer wird, die MEEMD hingegen schneller. Denn aus einem größeren Abstand zwischen den Extrempunkten resultiert für die FA-MVEMD eine größere Fenstergröße für die *Order Statistics Filter*, die hierdurch langsamer werden. Bei der MEEMD heißt dies hingegen weniger Stützstellen, durch welche die Einhüllende interpoliert werden muss. Die maximale Fenstergröße ist in den hier genutzten Beispielfällen durch die ungleichen Dimensionen stark begrenzt, was der Bearbeitungsgeschwindigkeit der FA-MVEMD zusätzlich zugute kommt. Auch wenn die dargestellten Zeiten eine Abschätzung der Einsparung erlauben, können die genauen Faktoren zwischen den benötigten Zeiten in anderen Fällen abweichen. Die Größenordnung wird sich allerdings nicht ändern. Es ist beispielsweise zu erwarten, dass sich die Unterschiede bei kleineren Datensätzen verringern, da dies für die MEEMD eine geringere Anzahl an einzelnen EMD-Ausführungen bedeutet.

5.6 Abschließende Bewertung der implementierten EMD-Varianten

Nach Abschluss der Implementierung der MEEMD und FA-MVEMD soll eine kurze Reflexion der Verfahren durchgeführt werden. Unzweifelhaft sind die Vorteile der FA-MVEMD durch die multivariate Verarbeitung, die Artefaktfreiheit und die hohe Geschwindigkeit gegeben. Nichtsdestotrotz ist anzumerken, dass durch die unterschiedlichen Fenstertypen ein hoher Einfluss auf die extrahierten IMFs ausgeübt werden kann. Dies ist im Hinblick auf die physikalische Interpretierbarkeit problematisch. Zusätzlich ist die Beschränkung der Fenstergröße durch die kleinste Dimension, und damit die fehlende Möglichkeit zur Extrahierung großer Strukturen für eine Analyse hinderlich. Dafür könnte jedoch womöglich Abhilfe geschaffen werden, wenn in der FA-MVEMD die entsprechenden Anpassungen umgesetzt werden.

In Anbetracht dieser Probleme scheint es umso wertvoller, dass für die MEEMD eine deutliche Beschleunigung entwickelt werden konnte. Denn die MEEMD ist hinsichtlich der physikalischen Interpretierbarkeit bereits besser getestet und geprüft als die FA-MVEMD. Das liegt unter anderem auch daran, dass sie algorithmisch näher an der EMD anzusiedeln ist, die bereits in einem breiten Spektrum unterschiedlicher Forschungsgebiete eingesetzt wurde. Das Problem der Artefakte kann durch zusätzlichen Rechenaufwand weitestgehend behoben werden. Durch die Beschleunigung kann in der selben Zeit folglich ein qualitativ besseres Ergebnis erzielt werden als vorher.
Insgesamt lässt sich also sagen, dass die FA-MVEMD bei einem guten Verständnis der Auswirkungen der Fenstertypen und einem Datensatz, der in jede Richtung möglichst gleichviele Punkte enthält, angewendet werden sollte. Ein anderer Anwendungsfall ergibt sich, wenn schnell ein erster Eindruck für mögliche enthaltene Strukturen benötigt wird, oder große Strukturen nicht von Interesse sind. Ersteres ist allerdings auch durch die Nutzung der MEEMD mit einer geringen Ensemble-Größe möglich. Für die extensive Auswertung von Geschwindigkeitsfeldern, wie es hier der Hauptanwendungsfall ist, scheint die Verwendung der beschleunigten MEEMD trotz des immer noch hohen Rechenaufwandes unumgänglich.

6 Clauser Chart Methode

Um für Geschwindigkeitsprofile unterschiedlicher Strömungen eine Vergleichbarkeit zu schaffen, muss eine Normierung erfolgen. Dabei werden sowohl der Abstand y zur Wand (der Oberfläche des untersuchten Objektes), als auch die Geschwindigkeit $u(y)$ normiert. Eine der Größen, mit Hilfe derer die Normierung durchgeführt wird, ist die Wandschubspannnungsgeschwindigkeit u_τ. Diese kann mit der Clauser Chart Methode bestimmt werden. Außerdem können Clauser Charts eingesetzt werden, um die Wandposition abzuschätzen, sollte diese nicht - oder nicht hinreichend - bekannt sein. Desweiteren können Clauser Charts - wie im vorliegenden Fall angedacht - auch eingesetzt werden, um zu analysieren, wie nah an der Wand gemessen wurde. Im Zusammenspiel mit der EMD kann so außerdem untersucht werden, inwieweit die verschiedenen Moden eine Abhängigkeit zum Wandabstand aufweisen. In Abschnitt 6.1 sollen zunächst Begriffe wie Wandschubspannung erklärt, sowie der theoretische Hintergrund erarbeitet werden. Anschließend wird in Abschnitt 6.2 definiert, welche Daten als Eingabe für die Berechnung akzeptiert werden und wie das Ergebnis abgespeichert wird. Nachdem die Implementierung vorgestellt wurde (Abschnitt 6.3), werden schließlich noch Ergebnisse gezeigt und mit einem Referenzfall verglichen (Abschnitt 6.4).

6.1 Theoretische Grundlagen und Prinzip

In diesem Abschnitt werden sowohl die Begriffe erklärt, die für die Clauser Chart Methode wichtig sind (Abschnitt 6.1.1 und Abschnitt 6.1.2), als auch die Methode selbst (Abschnitt 6.1.3). Da für die Clauser Chart Methode eine Funktion durch Messdaten gefittet werden muss, wird abschließend mit der Methode der kleinsten Quadrate die Theorie dazu erarbeitet (Abschnitt 6.1.4).

6.1.1 Wandschubspannung und Wandschubspannungsgeschwindigkeit

Die Wandschubspannung τ ist die durch die Bewegung des Fluids auf die Wand ausgeübte, parallel zur Oberfläche wirkende, Kraft $F_{||}$ pro Flächeneinheit A. Sie folgt aus der Reibung zwischen dem Fluid und der Oberfläche des Objektes und kann

durch Gleichung 6.1 beschrieben werden:

$$\tau = \frac{F_{||}}{A} \tag{6.1}$$

Es handelt sich um eine Scherspannung, da die Kraft parallel zur Wand wirkt. Mithilfe der Wandschubspannnung τ und der Dichte ρ des Fluids kann eine Wandschubspannungsgeschwindigkeit definiert werden:

$$u_\tau = \sqrt{\frac{\tau}{\rho}} \tag{6.2}$$

Diese sollte tatsächlich nicht als eine Geschwindigkeit verstanden werden. Die Bezeichnung resultiert aus der Einheit, die u_τ aufweist (m/s). Die Wandschubspannungsgeschwindigkeit dient in erster Linie der Normierung von u und y. Bei den vorliegenden Messungen mit PIV und STB wurden lediglich Geschwindigkeitsfelder bestimmt, ein direktes Messen der Wandschubspannnung wurde nicht vorgenommen. Folglich wird eine Methode benötigt, aus den Daten die Wandschubspannungsgeschwindigkeit zu bestimmen.

6.1.2 Law of the Wall und universelles Geschwindigkeitsprofil

Das Geschwindigkeitsprofil für eine turbulente Grenzschicht ist in Abbildung 6.1 zu sehen. So unterteilt sich die Grenzschicht in die so genannte viskose Unterschicht mit linearem Verlauf, einen Übergangsbereich und einen logarithmischen Bereich auf. In der Abbildung ist die normierte Geschwindigkeit u^+ gegen den normierten Wandabstand y^+ (mit logarithmischer Achsskalierung) aufgetragen. Die Normierung der Geschwindigkeit und dem Abstand zur Wand wird mit zwei charakteristischen Größen der Strömung, der Wandschubspannungsgeschwindigkeit u_τ und der kinematischen Viskosität ν durchgeführt. Es gilt:

$$u^+ = \frac{u}{u_\tau} \tag{6.3}$$

$$y^+ = \frac{y u_\tau}{\nu} \tag{6.4}$$

Durch diese Normierung werden sowohl die Geschwindigkeits- als auch die Abstandskomponente dimensionslos, was zu der universellen Geschwindigkeitsverteilung führt [36, S. 721].

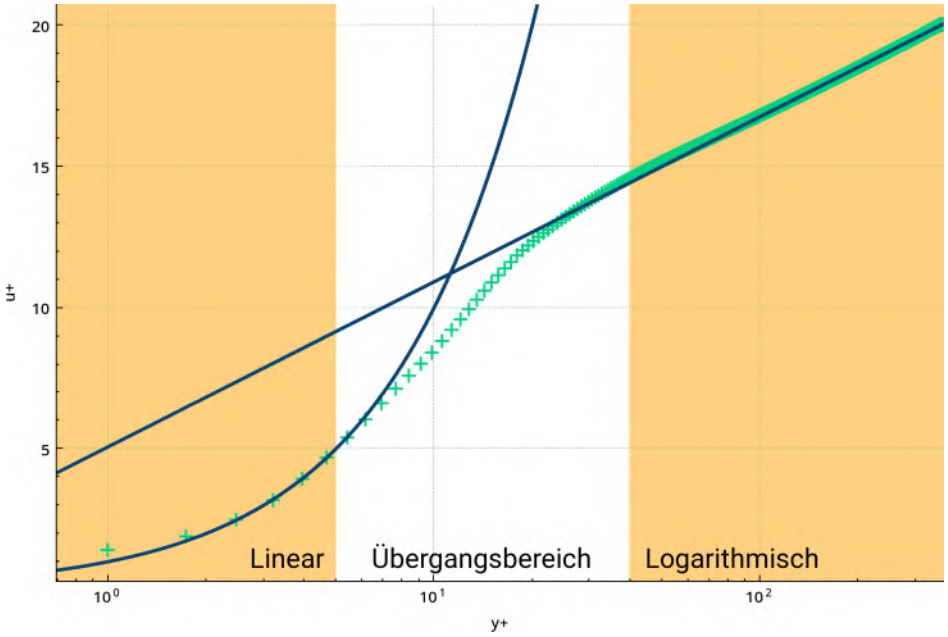


Abbildung 6.1: Normiertes Geschwindigkeitsprofil einer turbulenten Grenzschicht

Für den logarithmischen Teil gilt [45, S. 67]:

$$\frac{u(y)}{u_\tau} = \frac{1}{\kappa} * \ln\left(\frac{yu_\tau}{\nu}\right) + B \tag{6.5}$$

wobei κ die von Karman Konstante ist und B einfach eine additive Komponente. Mit den normierten Größen u^+ und y^+ dargestellt ergibt sich:

$$u^+ = \frac{1}{\kappa} * \ln(y^+) + B \tag{6.6}$$

Dieser Zusammenhang wird auch als *Log Law* oder *Law of the Wall* bezeichnet. Für den linearen Teil des Profils gilt hingegen:

$$\frac{u(y)}{u_\tau} = \frac{yu_\tau}{\nu} \tag{6.7}$$

Beziehungsweise mit den normierten Größen:

$$u^+ = y^+ \tag{6.8}$$

6.1.3 Clauser Chart Methode zur Bestimmung der Wandschubspannungsgeschwindigkeit

Für genau diesen Anwendungsfall - Bestimmung der Wandschubspannnungsgeschwindigkeit - wird die Clauser Chart Methode genutzt, die erstmals durch Clauser [8] beschrieben wurde. Dazu wird Gleichung 6.5 herangezogen, und somit der logarithmische Teil eines Geschwindigkeitsprofils betrachtet. u und y sind experimentelle Messgrößen und κ sowie B Konstanten. Da auch ν durch das Fluid definiert ist, ist u_τ die einzig verbleibende Unbekannte in der Gleichung. Um u_τ zu bestimmen, muss für die gemessenen Daten also zunächst ein über die Zeit gemitteltes Geschwindigkeitsprofil $u(y)$ ermittelt werden. Anschließend wird die Logarithmus-Funktion gesucht, die unter Berücksichtigung der Konstanten die beste Näherung der Messdaten and die Theorie liefert [46]. Dies kann beispielsweise als eine iterative Parameteroptimierung umgesetzt werden [18]. Problematisch bei der Verwendung der Clauser Chart Methode ist, dass die „Konstanten“ κ und B in verschiedenen Veröffentlichungen mit unterschiedlichen Werten beziffert wurden. So wird κ mit Werten zwischen 0.32 und 0.45 angegeben, die Werte für B liegen zwischen 3.5 und 6.3 [51, S. 3080].

6.1.4 Methode der kleinsten Quadrate

Eine Methode, um eine gesuchte Funktion durch eine Folge von Datenpunkten zu „fitten“, ist die Methode der kleinsten Quadrate Wie bei jeder Kurvenanpassung muss dazu zunächst eine sogenannte *Modellfunktion* bekannt sein. Die zugehörigen Parameter werden als Modellparameter bezeichnet. Je nachdem, ob zwischen den einzelnen Modellparametern ein linearer Zusammenhang besteht oder nicht, wird zwischen linearen und nichtlinearen Fits unterschieden. Während lineare Fits direkt analytisch bestimmt werden können, muss bei nichtlinearen Funktionen ein iteratives Näherungsverfahren genutzt werden - die Methode der kleinsten Quadrate lässt beides zu. Hier wird die Modellfunktion so durch die Datenpunkte gelegt, dass die Summe der Fehlerquadrate minmal wird. Unter dem Fehlerquadrat versteht man die quadrierte Abweichung eines - beispielsweise gemessenen - Wertes y_i an der Stelle x_i von dem Modellfunktionswert $f(x_i)$. Die zu minimierende Funktion $R^2(\vec{a})$

mit den Modellparametern $\vec{a} = (a_1, a_2, ...a_m)$ lautet also für n Datenpunkte [47]:

$$R^2(\vec{a}) = \sum_{i=1}^{n} (y_i - f(x_i, \vec{a}))^2 \tag{6.9}$$

Die Quadrierung wird vorgenommen, um starke Abweichungen von der Modellfunktion stärker zu gewichten als geringe. Dadurch wird die Funktion möglichst nah an jeden Punkt „gezogen". Gleichzeitig entsteht jedoch auch eine Anfälligkeit für Ausreißerwerte, die daher nach Möglichkeit aussortiert werden sollten. Für den einfachsten Fall einer linearen Modellfunktion $f(x) = a_0 + a_1 * x$ mit den Parametern a_0 und a_1 lautet die zu minimierende Gleichung:

$$R^2(a_0, a_1) = \sum_{i=1}^{n} (y_i - (a_0 + a_1 * x_i))^2 \tag{6.10}$$

Und zur Bestimmung des Minimums ergeben sich daraus die Gleichungen

$$\frac{\delta R^2(a_0, a_1)}{\delta a_0} = -2 * \sum_{i=1}^{n} (y_i - (a_0 + a_1 * x_i)) = 0 \tag{6.11}$$

$$\frac{\delta R^2(a_0, a_1)}{\delta a_1} = -2 * \sum_{i=1}^{n} (y_i - (a_0 + a_1 * x_i)) * x_i = 0 \tag{6.12}$$

Dies kann allgemein aufgelöst werden [47]:

$$a_1 = \frac{n \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2} \tag{6.13}$$

$$a_0 = \frac{\bar{y} \sum_{i=1}^{n} x_i^2 - \bar{x} \sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2} \tag{6.14}$$

Außerdem kann a_0 dargestellt werden durch

$$a_0 = \bar{y} - a_1 \bar{x} \tag{6.15}$$

Die Lösung für einen linearen Fit kann hiermit direkt analytisch erfolgen. Für einen nichtlinearen Fit müsste ein iteratives Verfahren zur Bestimmung des Minimums von R^2 angewandt werden. Da dies jedoch nicht benötigt wird, wird es nicht weiter betrachtet.

6.2 Definition der Eingabe- und Ausgabedaten

Bevor eine tatsächliche Implementierung des Verfahrens vorgenommen werden kann, muss definiert werden, welche Daten als Eingabedaten akzeptiert werden, und - in diesem Fall noch wichtiger - wie die Ausgabedaten aussehen. Als Eingabedaten kommen, methodisch bedingt, nur Geschwindigkeitsprofile in Frage. Ein solches Geschwindigkeitsprofil verfügt mindestens über eine Ortskoordinate und eine Geschwindigkeitskomponente, und kann somit in einer einfachen Tabelle dargestellt werden. Damit unterscheidet es sich in der rein formalen Beschreibung nicht von den Geschwindigkeitsfeldern, die bereits eingelesen und verarbeitet werden können. Das Ergebnis der Clauser Chart Berechnung ist die Wandschubspannnungsgeschwindigkeit u_τ. Da diese jedoch dazu dient, u und y zu normieren, kann im Rahmen der Berechnung auch die Normierung durchgeführt werden, sodass auch das Ergebnis wiederum ein (dann normiertes) Geschwindigkeitsprofil ist. Dieses kann mit den bereits vorhandenen Methoden zum Schreiben von Dateien gespeichert werden. Nichtsdestotrotz sollte auch u_τ in der Ergebnisdatei direkt abzulesen sein, da es auch für andere Auswertungen von Interesse sein könnte. Dazu wird ein kleiner Trick angewandt: Das ASCII-Tecplot Format „.dat“ verfügt über die Möglichkeit, Kommentare in die Datei zu schreiben. Ein Kommentar beginnt mit einem „#“. Die entsprechenden Lese- und Schreibroutinen wurden nun so angepasst, dass zwischen den Meta-Informationen über den Datensatz und den eigentlichen Daten eine zusätzliche Zeile eingefügt, beziehungsweise ausgelesen werden kann. Nachfolgend finden sich zur Veranschaulichung die ersten Zeilen einer exemplarischen Datei, die hervorgehobene Zeile ist die neu eingefügte:

```
1 TITLE = "prof_ClauserChart"
2 VARIABLES = "y+" "u+" "y/delta" "u_mean/u_inf"
3 #U_TAU=0.540246, NY=1.54838e-05, KAPPA=0.398952, B=4.98078
```

Diese Zeile enthält u_τ und zusätzlich sämtliche im Rahmen der Clauser Chart Berechnungen genutzten Konstanten. So kann die Ergebnisdatei direkt wieder eingelesen und korrekt dargestellt werden. Der Vorteil dabei, diese Information in einem Kommentar zu „verstecken“ besteht darin, dass die Datei trotzdem problemlos in Tecplot geöffnet werden kann. Ohne „#“ führt der Versuch des Einlesens in Tecplot hingegen zu Fehlern, da Tecplot diese Zeile nicht interpretieren kann.

6.3 Implementierung des Verfahrens

Für das Verfahren muss ein Fit durch die gegebenen Messwerte gelegt werden. Dazu müssen zunächst etwaige in Frage kommende Modellfunktionen betrachtet werden, um zu entscheiden, ob ein analytischer oder ein iterativer Ansatz genutzt werden sollte.

6.3.1 Bestimmung von u_τ durch den logarithmischen Bereich

Zunächst wird nur der logarithmische Bereich betrachtet. Durch Multiplikation von Gleichung 6.5 mit u_τ ergibt sich:

$$u(y) = \frac{u_\tau}{\kappa} * \ln\left(\frac{y u_\tau}{\nu}\right) + B u_\tau \tag{6.16}$$

Wird nun der Logarithmus durch die Summe zweier Logarithmen ausgedrückt, lautet die Gleichung:

$$u(y) = \frac{u_\tau}{\kappa} * \left(\ln(y) + \ln\left(\frac{u_\tau}{\nu}\right)\right) + B u_\tau \tag{6.17}$$

$$u(y) = \frac{u_\tau}{\kappa} * \ln(y) + \frac{u_\tau}{\kappa} * \ln\left(\frac{u_\tau}{\nu}\right) + B u_\tau \tag{6.18}$$

Allgemein lässt sich eine Logarithmus-Funktion beschreiben durch

$$f(x) = a_0 + a_1 * \ln(x) \tag{6.19}$$

Die beiden Gleichungen können ineinander überführt werden:

$$f(x) = u(y), \ln(x) = \ln(y) \tag{6.20}$$

$$a_1 = \frac{u_\tau}{\kappa} \tag{6.21}$$

$$a_0 = \frac{u_\tau}{\kappa} * \ln\left(\frac{u_\tau}{\nu}\right) + B u_\tau \tag{6.22}$$

Diese Darstellung erlaubt es, durch die experimentellen Daten mithilfe der Methode der kleinsten Quadrate eine Logarithmus-Funktion zu legen. Es handelt sich hierbei um ein lineares Problem, was auch in Abbildung 6.1 zu erkennen ist: Wird y logarithmiert, ist die resultierende Funktion $f(y_{ln})$ linear und es kann zum Fitten die Methode der kleinsten Quadrate eingesetzt werden. Gleichung 6.20 lässt nicht zu,

dass κ und B frei gewählt werden. Durch die Definition von κ ist die Gleichung bereits hinreichend bestimmt, sodass B aus den bekannten Werten berechnet werden kann. Die Berechnung des Clauser Charts durch die Datenmenge $u(y), y$ findet somit nach Algorithmus 3 statt. Dabei werden nur die Datenpunkte im logarithmischen Teil betrachtet - dieser wird durch den Nutzer in der GUI definiert.

Algorithmus 3 Algorithmus zur Berechnung der Clauser Chart Parameter

1: Logarithmieren jedes y Wertes: $y_{ln} = \ln(y)$
2: Bestimmen der Parameter a_0 und a_1 mittels eines linearen Fits $u(y_{ln}) = a_0 + a_1 y_{ln}$
3: Berechne $u_\tau = a_1 * \kappa$
4: Berechne $B = \frac{a_0}{u_\tau} - \frac{\ln(\frac{u_\tau}{\nu})}{\kappa}$
5: Berechne $u^+ = \frac{u(y)}{u_\tau}$ und $y^+ = \frac{y u_\tau}{\nu}$

6.3.2 Automatische Bestimmung von u_τ, κ, B und der Wandposition mithilfe des linearen und logarithmischen Bereiches

Eine einfachere Methode zur Bestimmung von u_τ besteht darin, den Fit im linearen Teil des Geschwindigkeitsprofils durchzuführen. Dass dies häufig nicht gemacht wird, liegt daran, dass der Messbereich in vielen Versuchen nicht nah genug an die Wand heranreicht [26, S. 366]. Sind jedoch hinreichend viele und genaue Daten aus dem linearen Teil des Profils vorhanden, sollten diese genutzt werden.
Dazu kann wiederum die Methode der kleinsten Quadrate für eine lineare Funktion angewandt werden. Dazu muss Gleichung 6.7 umgestellt werden zu:

$$u(y) = \frac{u_\tau^2}{\nu} y \tag{6.23}$$

Der lineare Fit $u(y) = a_0 + a_1 y$ durch die Messpunkte verfügt allerdings über zwei Parameter, a_0 und a_1. Die sogenannte Haftbedingung aus der Strömungsmechanik fordert aber, dass die Funktion durch den Ursprung läuft, also $u(y) = 0$ für $y = 0$ - und somit $a_0 = 0$ sein müsste. Ist $a_0 \neq 0$, kann damit die Wandposition berechnet

und die Messpunkte entsprechend verschoben werden:

$$u(y_{Wand}) = 0 \tag{6.24}$$

$$\Leftrightarrow a_0 + a_1 y_{Wand} = 0 \tag{6.25}$$

$$\Leftrightarrow y_{Wand} = -\frac{a_0}{a_1} \tag{6.26}$$

Nun können alle y-Werte um die berechnete Wandposition verschoben werden. Der Parameter a_1 des Fits gibt die Steigung der linearen Funktion an und ist in u_τ überführbar:

$$u_\tau = \sqrt{a_1 \nu} \tag{6.27}$$

Anschließend kann wiederum der logarithmische Teil der Funktion gefittet werden - mit dem Unterschied, dass nun nicht u_τ als Unbekannte in der Gleichung auftaucht, sondern κ.
Zusammengefasst werden also die folgenden Schritte durchgeführt:

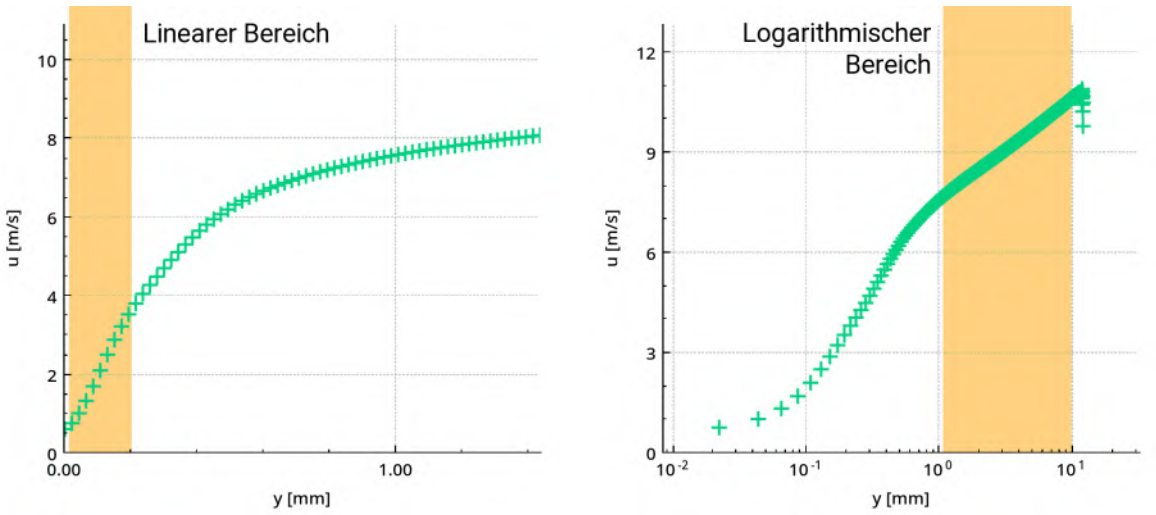
Algorithmus 4 Algorithmus zur Berechnung der Clauser Chart Parameter

1: Bestimmen der Parameter a_0 und a_1 mittels eines linearen Fits $u(y) = a_0 + a_1 y$ im als linear definierten Bereich
2: Bestimmen der Wandposition $y_{Wand} = -\frac{a_0}{a_1}$
3: Verschieben eines jeden Messwertes $y_i = y_i - y_{Wand}$
4: Berechne $u_\tau = \sqrt{a_1 \nu}$
5: Logarithmieren der y Werte: $y_{ln} = \ln(y)$
6: Bestimmen der Parameter b_0 und b_1 mittels eines linearen Fits $u(y_{ln}) = b_0 + b_1 y_{ln}$ im als logarithmisch definierten Bereich
7: Berechne $\kappa = b_1 u_\tau$
8: Berechne $B = \frac{b_0}{u_\tau} - \frac{\ln(\frac{u_\tau}{\nu})}{\kappa}$
9: Berechne $u^+ = \frac{u(y)}{u_\tau}$ und $y^+ = \frac{y u_\tau}{\nu}$

Der Vorteil dieser Methode liegt auf der Hand: Statt der - je nach Veröffentlichung stark variierenden - Konstanten κ und B muss lediglich der Start und das Ende des linearen sowie logarithmischen Bereiches der Funktion abgeschätzt werden. Nachteilig ist anzumerken, dass diese Einschätzung subjektiv ist und somit eine potenzielle Fehlerquelle darstellt.

6.4 Ergebnisse

In Abbildung 6.2 ist ein gemitteltes Geschwindigkeitsprofil aus einer Messung dargestellt. Sowohl der logarithmische, als auch der lineare Teil des Profils sind gut zu erkennen. Für dieses Profil soll nun das ClauserChart bestimmt werden, dabei sollen die zuvor beschriebenen Methoden Anwendung finden.



(a) (b)

Abbildung 6.2: Exemplarisches Geschwindigkeitsprofil
(a) Lineare Skalierung des Wandabstandes (x-Achse), Zoom
(b) Logarithmische Skalierung des Wandabstandes (x-Achse)

Zum Vergleich wird in Abbildung 6.3 zusätzlich das Ergebnis eines Python-Skriptes gezeigt, bei dem sowohl κ, als auch B vorgegeben werden - die Funktion wurde dabei mithilfe des Newton-Raphson-Verfahrens genähert [50].
In Abbildung 6.4 wird der starke Einfluss von κ bei der ersten Methode deutlich: Bei Verwendung des Wertes 0.384, wie in dem Referenz-Chart, weist der berechnete Wert für u_τ einen anderen Wert auf. Bei der Wahl eines κ von 0.41 wird hingegen fast das selbe u_τ berechnet wie in Abbildung 6.3. Auch ist die Abweichung im linearen Teil geringer. Diese Tatsache sollte jedoch an dieser Stelle nicht überinterpretiert werden, da bei der hier angewandten Methode keine Korrektur der Wandposition erfolgt. Im Vergleich mit Abbildung 6.3 ist eine naheliegende Vermutung nach Betrachtung der hier vorgestellten Ergebnisse, dass die Wertekombination von κ und B aus der Referenz für dieses Geschwindigkeitsprofil nicht zulässig ist. Denn bei Vorgabe von κ wird bei der hier vorgestellten Methode B automatisch berechnet

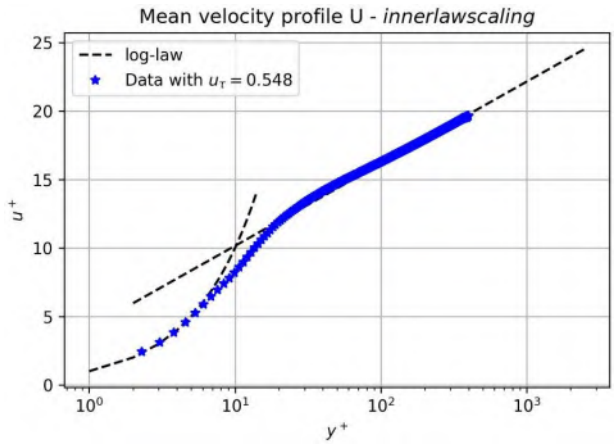


Abbildung 6.3: Referenz Clauser Chart, bestimmt mit $\kappa = 0.384$ und $B = 4.1$

- und dieses weicht mit 5.29 stark von dem Wert 4.1 aus dem Referenz Clauser Chart ab.

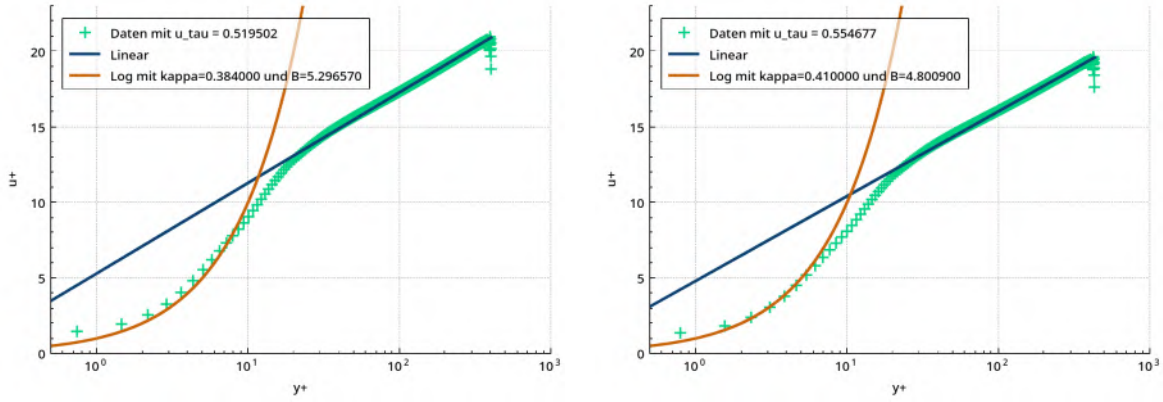
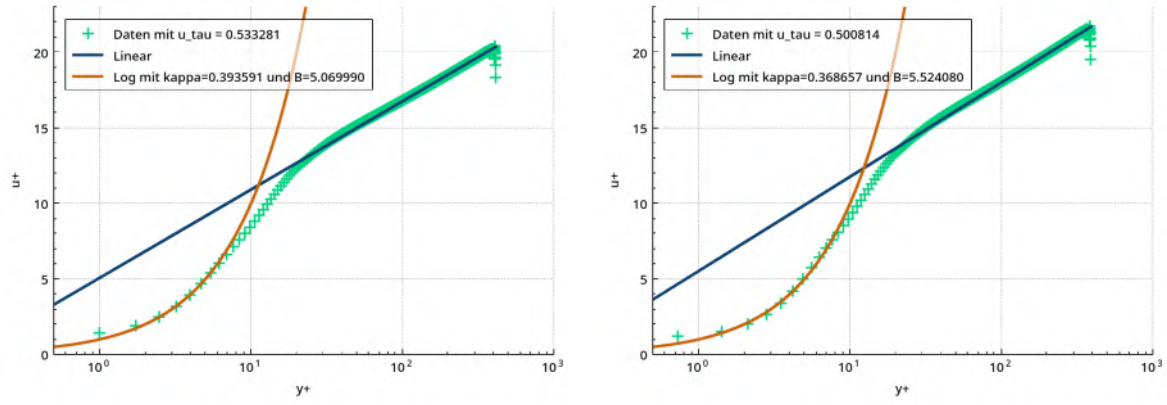


Abbildung 6.4: Clauser Chart unter Verwendung der ersten Methode mit $\kappa = 0.384$ und $\kappa = 0.41$

In Abbildung 6.5 werden zwei Clauser Charts gezeigt, die mithilfe der zweiten vorgestellten Methode erstellt wurden, also der Bestimmung von u_τ im linearen Bereich. Diese sind damit unabhängig von den Konstanten κ und B, die lediglich Ergebnisse der Rechnung sind. Zusätzlich wird die Wandposition verschoben. Der Vergleich zeigt, wie wichtig bei diesem Verfahren die korrekte Wahl der Grenzen für den linearen Bereich ist, da auch hier eine Abweichung in der zweiten Nachkommastelle auftritt. Durch die Verschiebung der Wandposition und der eigenständigen Berechnung der Konstanten passen jedoch beide Plots sehr gut an die normalisierten

Datenpunkte. Laut Theorie sollte der lineare Bereich in etwa bei $y^+ = 5$ enden, was offenbart, dass im Fall zwei der lineare Bereich zu groß gewählt wurde.



(a) **(b)**

Abbildung 6.5: Clauser Chart unter Verwendung der zweiten Methode mit unterschiedlichen Grenzen für den linearen Bereich
(a) $0.05mm < y < 0.15mm$, bzw. mit $u_\tau \approx 0.533$ normiert: $1.72 < y^+ < 5.163$
(b) $0.05mm < y < 0.25mm$, bzw. mit $u_\tau \approx 0.501$ normiert: $1.62 < y^+ < 8.09$

Insgesamt legen die hier vorgestellten Berechnungen nahe, dass bei der Verwendung der Clauser Charts der Einfluss der Konstanten auf das Ergebnis berücksichtigt werden sollten - unabhängig davon, welche Methode zur Bestimmung eingesetzt wird. Insbesondere die für B berechneten Werte liegen allesamt höher als der in der Referenz genutzte, während die mit der zweiten Methode für κ berechneten Werte bei korrekter Wahl des linearen Bereiches dem vorgegebenen nahe kommen. Prinzipiell ist allerdings festzuhalten, das beide implementierte Methoden funktionieren und als Hilfsmittel bei der Validierung der Messdaten eingesetzt werden können. Auch die automatische Bestimmung der Wandposition stellt einen Gewinn dar.

7 Fazit und Ausblick

In der vorliegenden Arbeit wurden mit der *Empirical Mode Decomposition* und der *Clauser Chart Methode* zwei Analysemethoden für eine Toolbox zur Auswertung von Geschwindigkeitsfeldern implementiert. Dazu wurden verschiedene Anpassungen an einer bereits existierenden Toolbox vorgenommen, wie die Implementierung von Darstellungsmöglichkeiten für Geschwindigkeitsfelder sowie Geschwindigkeitsprofile. Außerdem wurde eine Möglichkeit geschaffen, mehrere Berechnungen mittels einer *Queue* nacheinander auszuführen, was bei den zum Teil langen Berechnungsdauern einen großen Vorteil darstellt. So werden mehrere Berechnungen automatisch nacheinander ausgeführt, ohne dass der Nutzer eingreifen muss.
Bei der Implementierung der Verfahren wurde insbesondere die *Empirical Mode Decomposition* (EMD) in ihren verschiedenen Varianten ausführlich untersucht. Für die *Multidimensional Ensemble Empirical Mode Decomposition* (MEEMD) konnte eine entscheidende algorithmische Beschleunigung erzielt werden, welche die Anwendbarkeit dieses Verfahrens bei großen mehrdimensionalen Datensätzen verbessert. Die bei der MEEMD auftretenden Artefakte wurden analysiert und eine Lösungsmöglichkeit wurde dargestellt. Überdies wurde eine weitere Variante der EMD ausführlich untersucht, die *Fast and Adaptive Multidimensional, Multivariate Empirical Mode Decomposition* (FA-MVEMD). Dafür wurde eine MATLAB-Bibliothek in das C++ Programm eingebunden und der MATLAB Code auf verschiedenen Wegen beschleunigt. Für beide Verfahren wurden der Rechenaufwand und die Anwendbarkeit auf Geschwindigkeitsfelder ausführlich betrachtet und geprüft. Als Endergebnis steht die Empfehlung, die MEEMD in den betrachteten Fällen trotz deutlich höherem Rechenaufwand zu nutzen. Der Hauptgrund dafür ist, dass die FA-MVEMD in den betrachteten Fällen nicht in der Lage ist, große Strukturen zu extrahieren.
Die implementierte *Clauser Chart Methode* kann genutzt werden, um wahlweise aus dem logarithmischen oder linearen Bereich eines Geschwindigkeitsprofils die Wandschubspannungsgeschwindigkeit zu bestimmen. Zusätzlich konnte die Bestimmung weiterer für die Darstellung des normalisierten Geschwindigkeitsfeldes notwendige Konstanten automatisiert werden.

Bei einer derart umfangreichen Betrachtung verschiedener Verfahren können nicht sämtliche Ideen umgesetzt werden. Nichtsdestotrotz bieten viele Ansätze großes Potenzial und sollten in Zukunft weiter untersucht werden.
Für die MEEMD wäre wünschenswert, die Artefakte durch weniger rechenaufwändige Methoden zu reduzieren. Dabei ist vorstellbar, dass zunächst für die erste IMF die EMD entlang aller Dimensionen abzuschließen, und erst im Anschluss zur zweiten IMF überzugehen. In ersten Versuchen dazu resultierte dies in zu kleinen Strukturen, weshalb weitere Untersuchungen nötig sind. Sollte dieses Verfahren jedoch erfolgreich umgesetzt werden können, hätte es den positiven Nebeneffekt, dass ein Großteil der „Teil-IMFs“ nicht mehr bestimmt werden müsste. Ein anderer Ansatz ist die „Kopplung“ an benachbarte Zeilen / Spalten, sodass keine starken Diskontinuitäten entstehen können. Wie dies genau umgesetzt werden könnte, müsste allerdings auch genauer untersucht werden. Eine denkbare Möglichkeit wäre die Verwendung eines schwachen Weichzeichners nach jeder IMF-Bestimmung, die Auswirkung eines solchen Vorgehens sind jedoch nicht ohne Weiteres abzuschätzen.
Für die FA-MVEMD hingegen ist eine der dringendsten Anpassungen, dass das Problem des vorzeitigen Abbrechens bei ungleichen Dimensionen gelöst wird. Ein Ansatz dazu wurde bereits in der Arbeit vorgestellt, die Implementierung und Evaluierung steht jedoch noch aus. Auch ist es höchst interessant, andere Fenstertypen zu definieren und auf die hier genutzten Daten anzuwenden, da insbesondere die Fenstertypen 1 bis 4 sowie 7 nur bei synthetischen Daten erfolgsversprechend scheinen. Ein weiteres wichtiges Ziel ist die Überführung der MATLAB-Bibliothek in ein reines C++ Programm. Motivation dazu ist es, die Abhängigkeit von der *MATLAB Runtime* aufzuheben, da diese in der Installation und im Arbeitsspeicherbedarf aufwändig ist.
Aber auch für die Clauser Chart Methode, die im Verhältnis zu den EMD-Varianten einen eher kleinen Teil der Arbeit ausmacht, sind anschließende Untersuchungen denkbar. So könnten die implementierte Vorgehensweise auf Daten aus der Literatur angewandt werden, um diese zu verifizieren. Die Bereitstellung einer Implementierung unter Nutzung eines Näherungsverfahrens (wie beispielsweise dem Newton-Verfahren) bietet die Möglichkeit, nach dem klassischen Ansatz die Konstanten als Inputparameter zu nutzen - wie es an anderer Stelle getan wird.
Abgesehen von diesen interessanten Forschungsaspekten, die sich primär aus den implementierten Verfahren ergeben, sind auch für die Toolbox weitere Anpassun-

gen interessant. In Anbetracht der langen Berechnungsdauer, insbesondere für die MEEMD, wäre es beispielsweise wünschenswert, das Programm auf rechenstärkeren Computern und Berechnungsclustern nutzen zu können. Dafür wären verschiedene Dinge hilfreich: Zunächst ist es notwendig, die Kommandozeilen-Schnittstelle zu erweitern und darüber die Möglichkeit zu schaffen, das Programm ohne GUI zu nutzen. Die Definition von mehreren Berechnungen mit identischen Parametern sollte vereinfacht werden und nicht zuletzt sollten noch mehr Parallelisierungsmöglichkeiten wahrgenommen werden. Zu guter Letzt ist selbstverständlich auch die Implementierung weiterer Verfahren von hohem Interesse, um die Toolbox zu erweitern und tatsächlich als universelles Werkzeug zur Analyse von Geschwindigkeitsfeldern nutzen zu können.

Literaturverzeichnis

1. ALTAF, M Umair Bin; GAUTAMA, Temujin; TANAKA, Toshihisa; MANDIC, Danilo P.
   Rotation invariant complex empirical mode decomposition. In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*. 2007, Bd. 3, S. III–1009.
2. ANSELL, Phillip J; MULLENERS, Karen. Multiscale Vortex Characteristics of Dynamic Stall from Empirical Mode Decomposition. *AIAA Journal*. 2019, S. 1–18.
3. BHUIYAN, Sharif MA; ADHAMI, Reza R; KHAN, Jesmin F.
   Fast and adaptive bidimensional empirical mode decomposition using order-statistics filter based envelope estimation.
   EURASIP Journal on Advances in Signal Processing.
   2008, Jg. 2008, Nr. 1, S. 728356.
4. BHUIYAN, Sharif MA; ATTOH-OKINE, Nii O; BARNER, Kenneth E; AYENU-PRAH, Albert Y; ADHAMI, Reza R. Bidimensional empirical mode decomposition using various interpolation techniques. *Advances in Adaptive Data Analysis*. 2009, Jg. 1, Nr. 02, S. 309–338.
5. BIEBER, Michael. *QwtPlot3D* [online]. 2007 [besucht am 2019-09-02]. Abgerufen unter: http://qwtplot3d.sourceforge.net/.
6. CHANG, Li-Wen; LO, Men-Tzung; ANSSARI, Nasser; HSU, Ke-Hsin; HUANG, Norden E; WEN-MEI, W Hwu. Parallel implementation of multi-dimensional ensemble empirical mode decomposition. In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2011, S. 1621–1624.
7. CHEN, Chin-Yu; GUO, Shu-Mei; CHANG, Wei-Sheng; TSAI, Jason Sheng-Hong; CHENG, Kuo-Sheng. An improved bidimensional empirical mode decomposition: A mean approach for fast decomposition. *Signal Processing*. 2014, Jg. 98, S. 344–358.

8. CLAUSER, Francis H. The turbulent boundary layer. In: *Advances in applied mechanics*. Elsevier, 1956, Bd. 4, S. 1–51.

9. COLOMINAS, Marcelo A; SCHLOTTHAUER, Gaston; TORRES, Maria E. Improved complete ensemble EMD: A suitable tool for biomedical signal processing. *Biomedical Signal Processing and Control*. 2014, Jg. 14, S. 19–29.

10. *Data Format Guide - Tecplot 360 EX 2018 Release 2*. Bellevue, WA, 2018.

11. ECKELMANN, Helmut. *Einführung in die Strömungsmeßtechnik*. Vieweg+Teubner Verlag, 1997-01-01. ISBN 3519023792. Auch verfügbar unter: http://www.ebook.de/de/product/1293442/helmut_eckelmann_einfuehrung_in_die_stroemungsmesstechnik.html [356 S.].

12. EICHHAMMER, Emanuel. *QCustomPlot* [online]. 2018 [besucht am 2019-09-02]. Abgerufen unter: https://www.qcustomplot.com/.

13. HE, Zhi; LI, Jun; LIU, Lin; SHEN, Yi. Three-dimensional empirical mode decomposition (TEMD): A fast approach motivated by separable filters. *Signal Processing*. 2017, Jg. 131, S. 307–319.

14. HUANG, Norden E; SHEN, Zheng; LONG, Steven R. A new view of nonlinear water waves: the Hilbert spectrum. *Annual review of fluid mechanics*. 1999, Jg. 31, Nr. 1, S. 417–457.

15. HUANG, Norden E; SHEN, Zheng; LONG, Steven R; WU, Manli C; SHIH, Hsing H; ZHENG, Quanan; YEN, Nai-Chyuan; TUNG, Chi Chao; LIU, Henry H. The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*. 1998, Jg. 454, Nr. 1971, S. 903–995.

16. HUANG, Norden E; WU, Man-Li C; LONG, Steven R; SHEN, Samuel SP; QU, Wendong; GLOERSEN, Per; FAN, Kuang L. A confidence limit for the empirical mode decomposition and Hilbert spectral analysis. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*. 2003, Jg. 459, Nr. 2037, S. 2317–2345.

17. HUANG, Norden E; WU, Zhaohua. A review on Hilbert-Huang transform: Method and its applications to geophysical studies. *Reviews of geophysics*. 2008, Jg. 46, Nr. 2.

18. KENDALL, Anthony; KOOCHESFAHANI, Manoochehr.
A method for estimating wall friction in turbulent boundary layers. In: *25th AIAA aerodynamic measurement technology and ground testing conference*. 2006, S. 3834.

19. KIM, Donghoh; KIM, Kyungmee O; OH, Hee-Seok.
Extending the Scope of Empirical Mode Decomposition using Smoothing. *EURASIP Journal on Advances in Signal Processing*. 2012, Jg. 2012:168, S. 1–17.

20. KIM, Donghoh; OH, Hee-Seok.
EMD: Empirical Mode Decomposition and Hilbert Spectral Analysis. 2018. R package version 1.5.8.

21. KIM, Donghoh; OH, Hee-Seok.
EMD: a package for empirical mode decomposition and Hilbert spectrum.

22. KIM, Donghoh; PARK, Minjeong; OH, Hee-Seok.
Bidimensional Statistical Empirical Mode Decomposition.
IEEE Signal Processing Letters. 2012, Jg. 19, S. 191–194.

23. KOLL, Matthew David. *Empirical mode decomposition applied to planar and volumetric velocity field measurements of a supersonic separated flow*. 2018. Dissertation.

24. LUUKKO, PJJ; HELSKE, Jouni; RÄSÄNEN, Esa. Introducing libeemd: A program package for performing the ensemble empirical mode decomposition. *Computational Statistics*. 2016, Jg. 31, Nr. 2, S. 545–557.

25. NUNES, Jean Claude; BOUAOUNE, Yasmina; DELECHELLE, Eric; NIANG, Oumar; BUNEL, Ph.
Image analysis by bidimensional empirical mode decomposition. *Image and vision computing*. 2003, Jg. 21, Nr. 12, S. 1019–1026.

26. ÖRLÜ, Ramis; FRANSSON, Jens HM; ALFREDSSON, P Henrik.
On near wall measurements of wall bounded flows—the necessity of an accurate determination of the wall position. *Progress in Aerospace Sciences*. 2010, Jg. 46, Nr. 8, S. 353–387.

27. *Qt | Cross-platform software development for embedded & desktop* [online] [besucht am 2019-09-02]. Abgerufen unter: https://www.qt.io.

28. RAFFEL, Markus; WILLERT, Christian E.; WERELEY, Steve T.; KOMPENHANS, Jürgen. *Particle Image Velocimetry - A Practical Guide*. 2. Aufl. Springer Verlag, 2007-08-11. ISBN 978-3-540-72307-3. Auch verfügbar unter: http://www.ebook.de/de/product/6588753/markus_raffel_chris_willert_steve_wereley_juergen_kompenhans_particle_image_velocimetry.html.

29. REHMAN, Naveed ur; MANDIC, Danilo P. Empirical mode decomposition for trivariate signals. *IEEE Transactions on signal processing*. 2009, Jg. 58, Nr. 3, S. 1059–1068.

30. REHMAN, Naveed; MANDIC, Danilo P. Multivariate empirical mode decomposition. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*. 2009, Jg. 466, Nr. 2117, S. 1291–1302.

31. REW, Russ u. a. *Unidata NetCDF*. UCAR/NCAR - Unidata, 1989. Abgerufen unter DOI: 10.5065/d6h70cw6.

32. RIFFI, Jamal; MAHRAZ, Adnane Mohamed; ABBAD, Abdelghafour; TAIRI, Hamid. 3D extension of the fast and adaptive bidimensional empirical mode decomposition. *Multidimensional Systems and Signal Processing*. 2015, Jg. 26, Nr. 3, S. 823–834.

33. RILLING, Gabriel; FLANDRIN, Patrick; GONÇALVES, Paulo; LILLY, Jonathan M. Bivariate empirical mode decomposition. *IEEE signal processing letters*. 2007, Jg. 14, Nr. 12, S. 936–939.

34. RILLING, Gabriel; FLANDRIN, Patrick; GONCALVES, Paulo u. a. On empirical mode decomposition and its algorithms. In: *IEEE-EURASIP workshop on nonlinear signal and image processing*. 2003, Bd. 3, S. 8–11. Nr. 3.

35. SCHANZ, Daniel; GESEMANN, Sebastian; SCHRÖDER, Andreas. Shake-The-Box: Lagrangian particle tracking at high particle image densities. *Experiments in fluids*. 2016, Jg. 57, Nr. 5, S. 70.

36. SCHLICHTING, H. Boundary-layer theory. In: *Boundary-layer theory*. McGraw-Hill, 1979.

37. SEEL, Fabio. *Entwicklung einer Auswerte-Toolbox für experimentell gemessene Geschwindigkeitsfelder*. 2019. Technischer Bericht. Deutsches Zentrum für Luft- und Raumfahrt e.V. in der Helmholtz-Gemeinschaft.

38. TANAKA, Toshihisa; MANDIC, Danilo P.
Complex empirical mode decomposition. *IEEE Signal Processing Letters*. 2007, Jg. 14, Nr. 2, S. 101–104.

39. TEAM, MATLAB Support. *Should I use MATLAB Compiler SDK, or MATLAB Coder to integrate my MATLAB applications with C/C++? - MATLAB Answers - MATLAB Central* [online] [besucht am 2019-09-06].
Abgerufen unter: https://de.mathworks.com/matlabcentral/answers/223937-should-i-use-matlab-compiler-sdk-or-matlab-coder-to-integrate-my-matlab-applications-with-c-c.

40. THE MATHWORKS, Inc. *MATLAB Coder - MATLAB* [online]
[besucht am 2019-09-06].
Abgerufen unter: https://de.mathworks.com/products/matlab-coder.html.

41. THE MATHWORKS, Inc. *MATLAB Compiler SDK - MATLAB* [online]
[besucht am 2019-09-06]. Abgerufen unter:
https://de.mathworks.com/products/matlab-compiler-sdk.html.

42. THIRUMALAISAMY, Mruthun.
Fast and Adaptive Multivariate and Multidimensional EMD [online].
MATLAB Central File Exchange [besucht am 2019-09-09]. Abgerufen unter:
https://www.mathworks.com/matlabcentral/fileexchange/71270-fast-and-adaptive-multivariate-and-multidimensional-emd.

43. THIRUMALAISAMY, Mruthun R.; ANSELL, Phillip J. Fast and Adaptive Empirical Mode Decomposition for Multidimensional, Multivariate Signals. *IEEE Signal Processing Letters*. 2018-10, Jg. 25, Nr. 10, S. 1550–1554.
Abgerufen unter DOI: 10.1109/lsp.2018.2867335.

44. TORRES, Maria E; COLOMINAS, Marcelo A; SCHLOTTHAUER, Gaston; FLANDRIN, Patrick.
A complete ensemble empirical mode decomposition with adaptive noise. In: *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. 2011, S. 4144–4147.

45. VON KARMAN, Theodore. Mechanische Ähnlichkeit und Turbulenz. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*. 1930, Jg. 1930, S. 58–76.

46. WEI, Tie; SCHMIDT, Rodney; MCMURTRY, Patrick. Comment on the Clauser chart method for determining the friction velocity. *Experiments in fluids*. 2005, Jg. 38, Nr. 5, S. 695–699.

47. WEISSTEIN, Eric W. Least squares fitting. 2002.

48. WU, Zhaohua; HUANG, Norden E. Ensemble empirical mode decomposition: a noise-assisted data analysis method. *Advances in adaptive data analysis*. 2009, Jg. 1, Nr. 01, S. 1–41.

49. WU, Zhaohua; HUANG, Norden E; CHEN, Xianyao. The multi-dimensional ensemble empirical mode decomposition method. *Advances in Adaptive Data Analysis*. 2009, Jg. 1, Nr. 03, S. 339–372.

50. YPMA, Tjalling J. Historical development of the Newton–Raphson method. *SIAM review*. 1995, Jg. 37, Nr. 4, S. 531–551.

51. ZANOUN, E-S; DURST, F; NAGIB, H. Evaluating the law of the wall in two-dimensional fully developed turbulent channel flows. *Physics of Fluids*. 2003, Jg. 15, Nr. 10, S. 3079–3089.