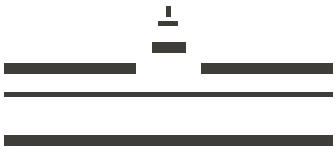
Simulation und Analyse zweidimensionaler Turbulenz auf parallelen Rechenarchitekturen

mit bedingten Mittelwerten
und elliptischen Wirbeln

als Diplomarbeit vorgelegt von

Markus Blank-Burian



Westfälische
Wilhelms-Universität
Münster

institut für
theoretische physik

Institut für
Informatik

Inhaltsverzeichnis

1. Einführung

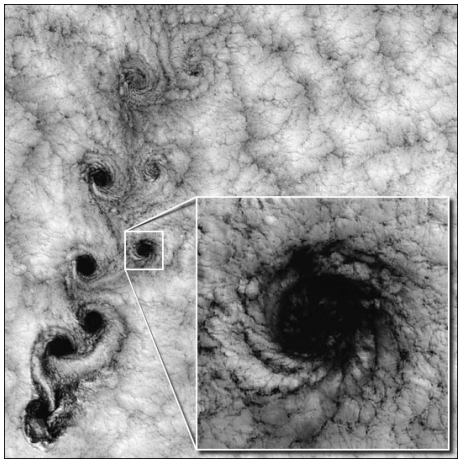
Der Titel dieser Arbeit „Simulation und Analyse zweidimensionaler Turbulenz auf parallelen Rechenarchitekturen“ wirft einige Fragen auf, die zunächst geklärt werden sollen, bevor mit mathematischen, physikalischen und informatischen Grundlagen der Grundstein dafür gelegt wird, um sich anschließend mit Details der Simulation und der Auswertung auseinanderzusetzen.

1.1. Was ist Turbulenz?

Als erstes stellt sich die Frage, was überhaupt Turbulenz ist. Definieren kann man Turbulenz (von lat.: turbare = drehen, beunruhigen, verwirren) als räumlich und zeitlich ungeordneten Fluss. Von großem physikalischen Interesse ist der Fluss von Flüssigkeiten und Gasen. Beide besitzen sowohl die Eigenschaft, dass sie sich nahezu geradlinig gleichförmig mit nur geringen Scherströmungen bewegen können, wie zum Beispiel Wasser in einem Fluss (auch genannt „laminare Strömung“) oder turbulent. Im turbulenten Bereich ist die Bewegung einzelner Flüssigkeits- oder Gasteilchen nahezu chaotisch. Abbildung 1.1 zeigt einige Beispiele für turbulente Strömungen in Luft und in Wasser. Wie man sieht, ist Turbulenz in nahezu jedem Bereich zu finden, in dem sich Gase oder Flüssigkeiten schnell bewegen und deren einheitliche Bewegung gestört wird.

Aufrecht erhalten wird Turbulenz durch eine äußere Kraft, die den normalen Fluss stört und dem System Energie hinzufügt. Ihr Gegenspieler ist die Viskosität (Zähigkeit). Sie sorgt dafür, dass permanent Energie aus dem System dissipiert wird und dadurch die Geschwindigkeit im Mittel abnimmt. Man bezeichnet den Zustand, in dem die Gesamtenergie des Systems nahezu konstant bleibt und sich das Energiespektrum nicht verändert, als stationären Zustand.

(a) Mt. St. Helens beim Ausbruch 1980 (Quelle: USGS/Cascades Volcano Observatory)



(b) Kármánsche Wirbelstraße in der Wolkenschicht hinter den Juan-Fernández-Inseln. (Quelle: NASA Earth Observatory)

(c) Verwirbelung an der Flügelspitze eines Flugzeugs (Quelle: Langley Research Center, NASA)

(d) Strömungsturbulenz an der Seite der USS Los Angeles (Quelle: US Navy)

Abbildung 1.1.: Beispiele für turbulente Strömungen

1.2. Warum müssen wir Turbulenz simulieren?

Mathematisch können Flüssigkeiten und Gase durch die Navier-Stokes-Gleichungen beschrieben werden, die seit der ersten Hälfte des 19. Jahrhunderts bekannt sind [Nav23] [Sto51]. Sie basieren auf einfachen Grundlagen wie der Impuls- oder Massenerhaltung und wurden durch zahlreiche numerische Simulationen bestätigt (das erste Mal im Jahr 1969 in zwei Dimensionen [Lil69] und im Jahr 1972 in drei Dimensionen [OP72]). Ihre Herleitung wird in Kapitel 2.1 erläutert. Obwohl diese Gleichungen schon so lange bekannt sind, ist ihre Lösungsstruktur bisher unverstanden. Deswegen hat das Clay Institute die Lösung dieser Gleichungen als Millenium Problem ausgeschrieben [CJW06]. Der Nobelpreisträger Richard Feynman hat Turbulenz einmal als „wichtigstes ungelöstes Problem der klassischen Physik“ bezeichnet [MK97].

Um die Lösungsstruktur der Navier-Stokes-Gleichungen besser verstehen zu können, trifft man vereinfachte Annahmen über das Verhalten des Gases oder der Flüssigkeit, indem man zum Beispiel annimmt, dass die Turbulenz homogen und isotrop ist. Homogenität bedeutet dabei eine Invarianz unter Verschiebung und Isotropie eine Invarianz unter Rotation. Diese Randbedingungen sind in einem Experiment nur sehr schwer zu erfüllen und deshalb bietet es sich an, auf Simulationen zurückzugreifen. Außerdem lassen sich am Rechner wesentlich einfacher Trajektorien einzelner Teilchen innerhalb der Substanz verfolgen oder einfach nur die Geschwindigkeit an jedem einzelnen Punkt direkt und eindeutig erfassen. Trotz der schwer zu realisierenden Randbedingungen existieren einige Experimente [KG02] z.B. mit Seifenfilmen auf Oberflächen, die zweidimensionale homogene isotrope Turbulenz realtiv gut approximieren.

Für zweidimensionale Turbulenz kann man die Navier-Stokes-Gleichungen zur Wirbeltransportgleichung umformen, wodurch die Rechnung wesentlich vereinfacht wird. Diese Herleitung wird im Kapitel 2.2 kurz dargestellt.

1.3. Warum müssen Turbulenzsimulationen parallelisiert und verteilt werden?

Turbulenzsimulationen gibt es seit den 70er Jahren. Seitdem hat sich die Computer-Hardware stark verbessert. Mit dieser besseren Hardware möchte man aber auch immer detailliertere Auswertungen machen und somit erhöht sich ebenso der Rechenaufwand. Bei dem in dieser Arbeit vorgestellten Algorithmus für die Simulation zweidimensionaler Turbulenz ist die Abhängigkeit der Laufzeit von der Datengröße $\mathcal{O}(N^2 \log(N))$. Daher kann bei Verdopplung der Rechengeschwindigkeit keine Verdopplung der Feldgröße erzielt werden. Dies macht es auch heutzutage noch notwendig, den Algorithmus so zu parallelisieren, dass er von mehreren Recheneinheiten gleichzeitig bearbeitet werden kann. Die größten simulierten Auflösungen liegen

derzeit bei 32768×32768 [BE12] und zur Berechnung von ausreichend Daten zur späteren Analyse in dieser Auflösung werden Supercomputer benötigt. Mit der Parallelisierung geht auch gleichzeitig eine Verteilung der Daten einher, die speziell dann von Nutzen ist, wenn der Speicher einer Recheneinheit zur Berechnung nicht ausreicht. Auf normalen Rechnern ist es inzwischen aus hardware- und kostentechnischen Gründen möglich, genügend Speicher einzubauen, so dass selbst die Berechnung von einem 32768×32768 großen Feld (entspricht 8GB Daten bei `double`-Genauigkeit) ohne Verteilung der Daten möglich ist. Bei Grafikbeschleunigern, die mittlerweile wegen ihrer parallelen Hardwarearchitektur gerne für physikalische Berechnungen und Simulationen eingesetzt werden, ist der Speicher allerdings für eine solch große Datenmenge noch nicht ausreichend. Abhängig von der Auflösung und gewählten Hardware spielt die Verteilung der Daten auf mehrere Recheneinheiten daher eine wichtige Rolle.

Da aus den Simulationsdaten hauptsächlich statistische Ensemblemittel gebildet werden, benötigt man rein theoretisch unendlich viele Simulationsdaten. Allerdings konvergiert das Zeitmittel über die berechneten Daten gegen das Ensemblemittel und deswegen reicht eine endliche Zahl an nicht-korrelierten Simulationsfeldern aus, um vernünftige Ergebnisse zu produzieren. Daher ist es wichtig, solche Felder mit einer hohen Rate zu produzieren (siehe Kapitel 4.3). Die simulierte Zeit pro Sekunde ist also das Maß, nach dem man die Simulationsgeschwindigkeit beurteilen kann.

1.4. Wie sieht die Parallelisierung aus und gibt es weitere Möglichkeiten, die Laufzeit zu reduzieren?

Zunächst muss man wissen, welchen Algorithmus man zur numerischen Integration verwenden möchte. Hierfür stehen verschiedene Verfahren zur Verfügung, die in Kapitel 3.3 beschrieben und miteinander verglichen werden. Die Berechnung des Integrationsschritts soll über ein Pseudospektralverfahren stattfinden, das auch für bisherige Simulationen verwendet wurde. Dieses Verfahren bietet wesentlich bessere Eigenschaften zur Simulation von Turbulenz als Verfahren, die mit finiten Differenzen arbeiten. Das Pseudospektralverfahren wird in Kapitel 3.4 genauer beschrieben. Es verwendet als Hauptbestandteil die Fouriertransformation (Kapitel 3.5).

An diesem Punkt kann nun die Parallelisierung und Verteilung ansetzen. Dazu werden einzelne Datenblöcke eines Datenfeldes auf unterschiedlichen Recheneinheiten gespeichert und bearbeitet. Da die Fouriertransformation eine nicht-lokale Operation ist, müssen dabei Daten nach einem „All-To-All“-Schema ausgetauscht werden. Für normale CPU-Cluster erledigt dies z.B. die FFTW-Bibliothek automatisch [FJ05]. Für Grafikbeschleuniger hingegen gibt es erst seit kurzer Zeit Untersuchungen zur Realisierung der verteilten FFT [CCM10] [NMM12] [NSM12]. Alle diese verteilten Algorithmen sind bisher nur für die dreidimensionale FFT geschrieben und leider

nicht im Internet frei zugänglich. Für die zweidimensionale verteilte FFT muss also ein Algorithmus neu implementiert werden, um auf Grafikbeschleunigern zu arbeiten.

Da die FFT der einzige Teil des Algorithmus ist, der viele Daten zwischen den Recheneinheiten austauschen muss, kann der restliche Teil der Simulation trivial über die lokalen Daten parallelisiert werden. Lediglich das Integrationsverfahren kann eine zusätzliche Synchronisation zum Abgleich des Zeitschrittes benötigen.

Für die gesamte Simulation ist es nun interessant, herauszufinden, welches Integrationsschema das schnellste ist, wie die Integrationsschemata auf unterschiedlicher Hardware skalieren, ob die Fouriertransformation auf Grafikkarten ebenso gut skaliert, wie mit CPUs und welche Auswirkungen die Rechengenauigkeit (`double` bzw. `float`) auf die Simulationsgeschwindigkeit hat. Dies alles wird in Kapitel 3 untersucht.

Außerdem besteht eventuell die Möglichkeit, die Simulationsdaten durch Interpolation vor der Auswertung zu vergrößern, um somit eine größere Auflösung zu erzielen. Da die Auswertung in der Regel nur eine Rechenzeit von $\mathcal{O}(N^2)$ benötigt, könnte dies die Rechenzeit um den Faktor $\mathcal{O}(\log(N))$ verkürzen. Die dazugehörigen Algorithmen, der Speedup durch Parallisierung und Implementation auf der Grafikkarte werden in Kapitel 4.9 genauer analysiert.

Ebenso wichtig, wie die Simulation, ist die spätere Auswertung der Daten. Diese besitzt zumeist eine Laufzeit von $\mathcal{O}(N^2)$. Diese fällt vor allem bei der Berechnung von bedingten Mittelwerten über das Wirbelstärkefeld ins Gewicht. Da diese im Mittelpunkt der weiteren Auswertung liegen, beschäftigt sich Kapitel 4.8.2 speziell mit der Parallisierung dieser bedingten Mittelwerte für Grafikbeschleuniger, wodurch gegenüber der CPU-Implementation ein wesentlicher Geschwindigkeitsvorteil erzielt wird.

1.5. Welche statistischen Eigenschaften kann man „messen“?

Die Analyse von Turbulenz bezieht sich meist auf statistische Eigenschaften, für die Ensemblemittel über ganze Simulationsläufe berechnet werden. Einige, aber noch lange nicht alle Daten sind aus der Theorie vorhersagbar. In Kapitel 2.5 finden sich die Herleitungen einiger wichtiger Kenngrößen, die später in Kapitel 4 benötigt werden. Dort wird analysiert, inwieweit die Simulationsdaten mit den theoretisch erwarteten Größen oder bisherigen Simulationen übereinstimmen. Dies ist natürlich sehr wichtig, um überhaupt die korrekte Funktionsweise der Simulation sicherzustellen. Außerdem soll dort der Einfluss der Auflösung auf die Daten genauer untersucht werden, da von dieser die Laufzeit der Simulation sehr stark abhängt. Natürlich möchte man numerische Ungenauigkeiten ausschließen und möglichst genaue Ergebnisse erzielen, um neue theoretische Modelle zu testen oder parametrisieren.

Wichtige statistische Größen sind Längenskalen und Zeitskalen. Die kleinste Längenskala wird von der Viskosität begrenzt. Diese Längenskala nennt man die Kolmogorov-Länge [Kol41c]. Die größte Länge des Systems ist die der größten Wirbelstruktur, die Integrale Länge [Tay35]. Eine weitere charakteristische Länge ist die der antreibenden Kraft. Sie liegt zwischen diesen beiden Längenskalen. In einer Simulation sind die kleinen Längenskalen durch die Simulationsauflösung begrenzt. Damit diese nicht in Konflikt mit der Simulation geraten, sind äußere Kraft und Viskosität so zu wählen, dass sämtliche Längen gut auflösbar sind.

1.6. Was sind Energie- und Enstropie-Kaskaden?

Startet man eine Simulation mit gegebener Viskosität und Kraft, so wird dem System Energie hinzugefügt bis es den stationären Zustand erreicht. Hierbei verändert sich das Energiespektrum, welches die Energie bestimmter Längenskalen wiederspiegelt. Das Spektrum folgt hierbei in Abhängigkeit von der gewählten Dimension der Simulation Potenzgesetzen, die sich aus der Theorie vorhersagen lassen (z.B. [Kra67] und [Kra71] für 2D). Diese Potenzgesetze bezeichnet man als Kaskaden, da sie dadurch aufgebaut werden, dass Energie von der Kraftlänge zu größeren oder kleineren Längenskalen transportiert wird. In drei Dimensionen wird nur Energie von kleinen zu großen Skalen transportiert. Diese Kaskade nennt man die „direkte“ Kaskade. In zwei Dimensionen existiert eine zusätzliche Kaskade, die Energie von kleinen zu großen Skalen transportiert. Diese nennt man die „inverse“ Kaskade. Beide Kaskaden existieren gleichzeitig, wie auch in [BM10] gezeigt wird.

1.7. Was ist eine Wahrscheinlichkeitsdichte?

Eine Wahrscheinlichkeitsdichte eines turbulenten Geschwindigkeitsfeldes gibt an, mit welcher Wahrscheinlichkeit an einem bestimmten Punkt eine bestimmte Geschwindigkeit vorkommen kann. Bei zweidimensionaler Turbulenz interessiert man sich eher für die Wirbelstärke, da die in diesem Fall eindimensionale Größe das System ebenfalls eindeutig beschreibt. Für homogene Turbulenz hängt diese Wahrscheinlichtkeit nicht mehr vom Ort ab.

Man kann nun auch eine Wahrscheinlichkeitsdichte von Geschwindigkeiten bzw. Wirbelstärken für zwei oder mehr Punkte definieren. Es ist hierbei zu beachten, dass an zwei gleichen Punkten natürlich nur die gleiche Geschwindigkeit/Wirbelstärke vorherrschen kann. Im umgekehrten Fall müssen zwei bliebig weit entfernte Punkte statistisch voneinander unabhängige Geschwindigkeiten/Wirbelstärken besitzen. Im Falle homogener isotroper Turbulenz muss die Wahrscheinlichkeitsdichte invariant unter Verschiebung und Rotation der Punkte sein.

Diese Wahrscheinlichkeitsdichte kann man nun auch als Ensemblemittel über punktweise Verteilungen definieren. Basierend auf diesem Ansatz haben Lundgren [Lun67], Monin [Mon67] und Novikov [Nov68] über die Navier-Stokes-Gleichungen eine zeitliche Evolutionsgleichung für die Wahrscheinlichkeitsverteilung aufgestellt. Dabei trifft man auf das sogenannte Schließungsproblem der Turbulenz. Es ist zum Beispiel nicht möglich, die zeitliche Evolution der Einpunktverteilung ohne die Zweipunktverteilung anzugeben. Eine Möglichkeit, dieses Problem zu beheben, wurde in [BCD02] unter Verwendung des Druckgradienten untersucht.

Eine andere Möglichkeit ist es, die auftauchenden Verteilungen über auf mehrere Punkte bedingte mittelere Geschwindigkeitsfelder oder Wirbelstärkefelder darzustellen, wie es in [Fri+10] gemacht wurde. Über Anwendung der Methode der Charakteristiken gelangt man zu Bewegungsgleichungen, welche die zeitliche Änderung dieser Punkte und deren Wirbelstärken auf diese bedingten Mittelwerte über das Geschwindigkeitsfeld bzw. das Wirbelstärkefeld zurückführen. Diese Rechnungen werden in Kapitel 5 noch einmal für den Spezialfall von zweidimensionaler Turbulenz wiederholt.

1.8. Wie kann man bedingte Mittelwerte theoretisch beschreiben?

Zunächst einmal benötigt man viele Daten, um sich anzuschauen, wie diese bedingten Mittelwerte aufgebaut sind. Diese Daten werden in Kapitel 5.3.1 und 5.3.2 genau dargestellt und analysiert. Analytisch berechnen kann man die auf N Punkte bedingten Mittelwerte über die für N+1 Punkte definierte Wahrscheinlichkeitsverteilung. Bisher wurde angenommen, dass die Verteilung näherungsweise gaußförmig ist [Fri+10]. Dies trifft allerdings in den vorliegenden Simulationen nur für den Spezialfall sehr kleiner Kräfte zu, kann aber recht einfach auf den allgemeineren Fall von stabilen Verteilungen erweitert werden, wie in Kapitel 5.10 gezeigt wird. Selbst diese Erweiterung kann allerdings weder die Formänderung des auf einen Punkt bedingten Mittelwerts in Abhängigkeit der vorgegeben Wirbelstärke beschreiben noch die Verkippung, die bei zwei Punkten beobachtet wird. Würde es gelingen, ein physikalisch motiviertes Modell für die Wahrscheinlichkeitsverteilung zu finden, so könnte man damit die Hierarchie der Evolutionsgleichungen schließen. In Kapitel 5.9 wird zunächst ein Modell untersucht, das mit Hermitepolynomen versucht, die Wahrscheinlichkeitsdichte zu approximieren. Anschließend wird in Kapitel 5.11 eine wesentlich bessere Methode gezeigt, die es nahezu vollständig schafft, den auf einen Punkt bedingten Mittelwert zu beschreiben. Sie beruht auf der Faltung von zwei oder mehreren Wahrscheinlichkeitsverteilungen, was einer Addition zweier Zufallsvariablen entspricht. Eine dieser Zufallsvariablen lässt sich auf Grund ihrer Oszillationen eventuell mit der äußeren Kraft identifizieren.

1.9. Was hat das Ganze mit elliptischen Wirbeln zu tun?

Die Verzerrung der mittleren Wirbelstrukturen des auf zwei Punkte bedingten Mittelwertes ist in erster Näherung nahezu elliptisch. Man könnte daher vermuten, dass eine solche Anordnung zweier elliptischer Wirbel stabil ist. Die Idee in [FF11] war es, aus der Navier-Stokes-Gleichung im Fourierraum eine Gleichung für elliptisch-gaußförmige Wirbel abzuleiten, die dann auf eine Ausdehnungsrichtung reduziert werden und mit einer Multipolentwicklung eines gekoppelten Punktwirbelmodells verglichen werden kann. In Kapitel 6 wird die Herleitung noch einmal auf eine andere Art und Weise wiederholt und anschließend mit vollständig elliptischen Wirbel weitergerechnet. Hieraus lassen sich relativ komplexe Bewegungsgleichungen für den Ort und die Form der Wirbel ableiten.

Eines der Kernresultate der Rechnung wird sein, dass sich elliptische verzogene Wirbel abhängig von ihrer Ausrichtung zueinander anziehen oder abstoßen. Dies ist eine mögliche Erklärung für die Anziehung zweier Wirbel, die letztendlich zur Verschmelzung dieser führt. Das Punktwirbelmodell kann dieses Phänomen nicht erklären. Außerdem wird gezeigt, dass es abhängig von Abstand und Wirbelstärke stabile Winkelstellungen zueinander gibt, die diese permanente Anziehung erst möglich macht. Der dafür nötige Abstand ist in der Rechnung etwas kleiner, als aus entsprechenden Simulationen [MZM88] zu erwarten wäre.

2. Grundlagen

Bevor es nun zur Beschreibung der Simulation und deren Auswertung kommt, müssen erst einmal einige Grundlagen definiert werden. Natürlich muss zunächst geklärt werden, was überhaupt mathematisch simuliert wird. Dies wird die Wirbeltransportgleichung sein, die sich im zweidimensionalen Fall aus den Navier-Stokes-Gleichungen herleiten lässt. Für inkompressible Flüssigkeiten nimmt diese eine noch einfachere Form an, die als Grundlage für die Simulation dienen soll.

Desweiteren stellt die Fouriertransformation einen wesentlichen Baustein der Simulation dar. Deswegen sollen hier noch einmal die wesentlichen Eigenschaften dieser wiederholt werden. Dies ist insbesondere deshalb wichtig, da für diese ein verteilter Algorithmus entworfen werden soll.

Da in Kapitel 5 zur Berechnung der bedingten Mittelwerte ein wenig Wahrscheinlichkeitstheorie benötigt wird, sollen die benötigten Definitionen und Schreibweisen hier ebenfalls kurz wiederholt werden.

Außerdem werden zur Charakterisierung und Überprüfung der Simulation zahlreiche statistische Größen der Turbulenz benötigt. Diese werden hier einmal kurz angegeben oder hergeleitet. Dabei werden sämtliche Formeln so aussehen, dass man durch einfachen Übergang von Integral zu Summe zur numerischen Berechnung gelangt.

Die Simulation und die späteren Auswertungsprogramme sollen aus Performance-Gründen nicht nur für CPU-Cluster, sondern auch für Grafikbeschleuniger entworfen werden. Da diese im Gegensatz zur regulären CPU-Systemen relativ neu sind, soll hier außerdem deren Hardwarearchitektur und Speichermodell vorgestellt werden.

2.1. Navier-Stokes-Gleichungen

Die Navier-Stokes-Gleichungen sind benannt nach Claude-Louis Navier und Sir George Gabriel Stokes. Sie beschreiben die zeitliche Entwicklung des Geschwindigkeitsfeldes einer Flüssigkeit. Im Wesentlichen folgen die Gleichungen aus Impuls- und Massenerhaltung. Weitere Herleitungen für diese finden sich z.B. in [Bat00] und [Pop00]

2.1.1. Reynolds'scher Transportsatz für intensive Größen

Zunächst betrachtet man die allgemeine Erhaltung einer intensiven Größe L einem beliebigen Volumenelement V eines Geschwindigkeitsfeldes $\boldsymbol{u}$. Der erste Term der rechten Seite beschreibt hierbei das Hereinströmen der Größe L in das Volumenelement und q gibt die Quellen oder Senken an

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int_V L\mathrm{d}V &= -\int_{\partial V} L\boldsymbol{u}\cdot\mathrm{d}\boldsymbol{A} + \int_V q\mathrm{d}V \\ &= -\int_V \nabla\cdot(L\boldsymbol{u})\,\mathrm{d}V - \int_V q\mathrm{d}V \\ &= \int_V \left(-\nabla\cdot(L\boldsymbol{u}) + q\right)\mathrm{d}V\end{aligned} \tag{2.1}$$

Im zweiten Schritt wurde dabei der Satz von Gauß angewandt. Vertauscht man nun Integration und Ableitung mit Hilfe der Leibnizschen Regel, so erhält man

$$\int_V \left(\frac{\partial L}{\partial t} + \nabla\cdot(L\boldsymbol{u}) - q\right)\mathrm{d}V = 0 \tag{2.2}$$

Zu beachten ist, dass hier aus der totalen Ableitung eine partielle Ableitung wurde. Da die Gleichung für jedes beliebige Volumen gilt, muss der Integrand stets verschwinden

$$\frac{\partial L}{\partial t} + \nabla\cdot(L\boldsymbol{u}) - q = 0 \tag{2.3}$$

2.1.2. Massenerhaltung

Setzt man in Gleichung (2.3) die Dichte ρ ein, so erhält man

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\boldsymbol{u}) = \frac{\partial \rho}{\partial t} + (\nabla\rho)\cdot\boldsymbol{u} + \rho\,(\nabla\cdot\boldsymbol{u}) = 0 \tag{2.4}$$

Da weder Masse erzeugt noch vernichtet wird, ist in diesem Fall $q = 0$.

2.1.3. Impulserhaltung

Benutzt man Gleichung (2.3) für den Impuls $\rho \boldsymbol{u}$, so folgt

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \boldsymbol{u}) = \boldsymbol{q} \tag{2.5}$$

Nun wendet man die Kettenregel an und sortiert alle Terme

$$\boldsymbol{u} \frac{\partial \rho}{\partial t} + \rho \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \nabla \cdot (\rho \boldsymbol{u}) + \rho \boldsymbol{u} \cdot \nabla \boldsymbol{u} = \boldsymbol{q} \tag{2.6a}$$

$$\boldsymbol{u} \left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) \right) + \rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right) = \boldsymbol{q} \tag{2.6b}$$

Der erste Faktor in Gleichung (2.6b) fällt auf Grund der Massenerhaltung (Gleichung (2.4)) weg und es bleibt übrig

$$\rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right) = \rho \frac{\mathrm{D} \boldsymbol{u}}{\mathrm{D} t} = \boldsymbol{q} \tag{2.7}$$

Die Abkürzung $\frac{\mathrm{D}}{\mathrm{D}t} = \frac{\partial}{\partial t} + \boldsymbol{u} \cdot \nabla$ in Gleichung (2.7) nennt man Substantielle Ableitung und sie ist gleich der totalen Ableitung einer mit dem Geschwindigkeitsfeld mitbewegten Größe.

2.1.4. Der Quellterm

Für den Quellterm q in Gleichung (2.7) lassen sich nun weitere Annahmen treffen. Er lässt sich unterteilen in Spannungskräfte $\nabla \cdot \mathbb{S}$ und äußere Volumenkräfte $\boldsymbol{f}$. $\mathbb{S}$ ist der Spannungstensor.

$$\boldsymbol{q} = \nabla \cdot \mathbb{S} + \boldsymbol{f} = \nabla \cdot (-\pi \mathbb{I} + \mathbb{T}) + \boldsymbol{f} \tag{2.8}$$

Hierbei ist $\pi = -\frac{1}{3} \mathrm{Sp}(\mathbb{S})$ ein zur Spur des Spannungstensors proportionaler Druckterm und $\mathbb{T}$ der reduzierte Spannungstensor. Auf seinen Nicht-Diagonal-Elementen stehen die Scherspannungen τ_{ij}.

2.1.5. Kompressible Newtonische Flüssigkeiten

Für kompressible newtonische Flüssigkeiten gelten folgende Annahmen:

- Der Spannungstensor ist proportional zur Dehnungsrate
- Die Flüssigkeit ist isotrop
- Bewegt sich die Flüssigkeit nicht, so ist $\nabla \cdot \mathbb{T} = 0$
- Für die Scherspannungen gilt $\tau_{ij} = \eta \frac{\partial u_i}{\partial x_j}$ (η ist hier die dynamische Viskosität)

Hieraus folgt, dass Druckterm bzw. Spannungstensor folgende Form haben müssen (siehe [Bat00])

$$\mathbb{T} = \eta \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \delta_{ij} \frac{2}{3} \nabla \cdot \boldsymbol{u} \right) \tag{2.9a}$$

$$\pi = p - \lambda \nabla \cdot \boldsymbol{u} \tag{2.9b}$$

Dabei ist λ eine zweite Viskositätskonstante und p der Druck. Mit Hilfe dieser zwei Gleichungen vereinfacht sich nun $\boldsymbol{q}$ auf (siehe auch [Bat00])

$$\boldsymbol{q} = -\nabla p + \lambda \nabla \left(\nabla \cdot \boldsymbol{u} \right) + \eta \nabla \cdot \left((\nabla \boldsymbol{u}) + (\nabla \boldsymbol{u})^T \right) + \boldsymbol{f} \tag{2.10}$$

Durch Einsetzen in Gleichung (2.7) erhält man nun die Navier-Stokes-Gleichung für kompressible newtonische Flüssigkeiten

$$\rho \frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} = -\nabla p + \lambda \nabla \left(\nabla \cdot \boldsymbol{u} \right) + \eta \nabla \cdot \left((\nabla \boldsymbol{u}) + (\nabla \boldsymbol{u})^T \right) + \boldsymbol{f} \tag{2.11}$$

Im reinen Gravitationsfeld würde die äußere Kraft zum Beispiel durch die Gravitationskraft bestimmt: $f = \rho \boldsymbol{g}$.

2.1.6. Inkompressible Newtonische Flüssigkeiten

Für inkompressible newtonische Flüssigkeiten ist die Substantielle Ableitung $\frac{\mathrm{D}\rho}{\mathrm{D}t} = 0$. Das heißt, dass sich bei der Bewegung eines infinitesimal kleinen Volumenelements entlang $\boldsymbol{u}$ sich dessen Dichte nicht ändert. Setzt man hier Gleichung (2.4) aus der Massenerhaltung ein, so bekommt man

$$\frac{\mathrm{D}\rho}{\mathrm{D}t} = \frac{\partial \rho}{\partial t} + \boldsymbol{u} \cdot \nabla \rho = -\rho \left(\nabla \cdot \boldsymbol{u} \right) = 0 \tag{2.12}$$

Da die Dichte nicht verschwindet, ist diese Bedingung äquivalent zu

$$\nabla \cdot \boldsymbol{u} = 0 \tag{2.13}$$

Üblicherweise wählt man außerdem $\lambda = 0$. Mit diesen Annahmen folgt aus Gleichung (2.11) die Navier-Stokes-Gleichung für inkompressible Newtonische Flüssigkeiten

$$\rho \frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} = \boldsymbol{q} = -\nabla p + \eta \Delta \boldsymbol{u} + \boldsymbol{f} \tag{2.14}$$

2.2. Wirbeltransportgleichung

Die Wirbelstärke ist definiert als die Rotation des Geschwindigkeitsfeldes

$$\boldsymbol{\omega}(\boldsymbol{x}, t) = \nabla \times \boldsymbol{u}(\boldsymbol{x}, t) \tag{2.15}$$

Über diese Definition lässt sich die Wirbeltransportgleichung aus der Navier-Stokes-Gleichung herleiten. Dazu wendet man die Rotation auf Gleichung (2.7) an.

$$\nabla \times \frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} = \nabla \times \left(\frac{\partial u}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right) = \frac{\partial \boldsymbol{\omega}}{\partial t} + \nabla \times (\boldsymbol{u} \cdot \nabla \boldsymbol{u}) = \nabla \times \frac{\boldsymbol{q}}{\rho} \tag{2.16}$$

Nun vereinfacht man die linke Seite mittels folgender Identität

$$\boldsymbol{u} \cdot \nabla \boldsymbol{u} = \frac{1}{2} \nabla (\boldsymbol{u}^2) - \boldsymbol{u} \nabla \times (\nabla \times \boldsymbol{u}) \tag{2.17a}$$

$$\begin{aligned} \nabla \times (\boldsymbol{u} \cdot \nabla \boldsymbol{u}) &= \frac{1}{2} \underbrace{\nabla \times (\nabla \boldsymbol{u}^2)}_{=0} - \nabla \times (\boldsymbol{u} \times (\nabla \times \boldsymbol{u})) \\ &= -\nabla \times (\boldsymbol{u} \times \boldsymbol{\omega}) \\ &= -\boldsymbol{u} \underbrace{(\nabla \cdot \boldsymbol{\omega})}_{=0} + \boldsymbol{\omega} (\nabla \cdot \boldsymbol{u}) - (\boldsymbol{\omega} \cdot \nabla) \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla) \boldsymbol{\omega} \end{aligned} \tag{2.17b}$$

Dabei wurde $\nabla \cdot \boldsymbol{\omega} = \nabla \cdot (\nabla \times \boldsymbol{u}) = 0$ benutzt. Dies ergibt dann folgende zwei äquivalente Gleichungen

$$\frac{\partial \boldsymbol{\omega}}{\partial t} = \nabla \times (\boldsymbol{u} \times \boldsymbol{\omega}) + \nabla \times \frac{\boldsymbol{q}}{\rho} \tag{2.18a}$$

$$\frac{\mathrm{D}\boldsymbol{\omega}}{\mathrm{D}t} = (\boldsymbol{\omega} \cdot \nabla) \boldsymbol{u} - \boldsymbol{\omega} (\nabla \cdot \boldsymbol{u}) + \nabla \times \frac{\boldsymbol{q}}{\rho} \tag{2.18b}$$

2.2.1. Kompressible Newtonische Flüssigkeiten

Für den Kraftterm kompressibler newtonischer Flüssigkeiten (Gleichung (2.10)) ergibt sich

$$\begin{aligned}\nabla \times \frac{\boldsymbol{q}}{\rho} &= \nabla \times \frac{1}{\rho}\left(\nabla \cdot (-\pi \mathbb{I} + \mathbb{T}) + \boldsymbol{f}\right) \\ &= \nabla \times \frac{1}{\rho}\left(-\nabla p + \lambda \nabla(\nabla \boldsymbol{u}) + \eta \nabla\left((\nabla \boldsymbol{u}) + (\nabla \boldsymbol{u})^T\right) + \boldsymbol{f}\right)\end{aligned} \tag{2.19}$$

Zur Vereinfachung benutzt man nun, dass $\lambda \nabla \times (\nabla(\nabla \cdot \boldsymbol{u})) = 0$ ist, sowie

$$\begin{aligned}-\nabla \times (\frac{1}{\rho}\nabla p) &= -\nabla \frac{1}{\rho} \times \nabla p - \frac{1}{\rho} \underbrace{(\nabla \times (\nabla p))}_{=0} \\ &= \frac{1}{\rho^2} \nabla \rho \times \nabla p\end{aligned} \tag{2.20}$$

und erhält damit insgesamt

$$\frac{\mathrm{D}\boldsymbol{\omega}}{\mathrm{D}t} = (\boldsymbol{\omega} \cdot \nabla)\boldsymbol{u} - \boldsymbol{\omega}(\nabla \cdot \boldsymbol{u}) + \frac{1}{\rho^2}(\nabla \rho) \times (\nabla p) + \nabla \times \frac{\nabla \cdot \mathbb{T}}{\rho} + \nabla \times \frac{\boldsymbol{f}}{\rho} \tag{2.21}$$

2.2.2. Inkompressible Newtonische Flüssigkeiten

Für inkompressible newtonische Flüssigkeiten vereinfacht man Gleichung (2.21) wie zuvor in Abschnitt 2.1.6 mit $\nabla \cdot \boldsymbol{u} = 0$ und $\nabla \times (\Delta \boldsymbol{u}) = \Delta(\nabla \times \boldsymbol{u}) = \Delta \boldsymbol{\omega}$

$$\begin{aligned}\frac{\mathrm{D}\boldsymbol{\omega}}{\mathrm{D}t} &= (\boldsymbol{\omega} \cdot \nabla)\boldsymbol{u} + \frac{1}{\rho^2}(\nabla \rho) \times (\nabla p) + \frac{1}{\rho}\eta \Delta \boldsymbol{\omega} - \frac{\eta}{\rho^2}(\nabla \rho) \times \eta \Delta \boldsymbol{u} + \nabla \times \frac{\boldsymbol{f}}{\rho} \\ &= (\boldsymbol{\omega} \cdot \nabla)\boldsymbol{u} + \frac{1}{\rho^2}(\nabla \rho) \times (\nabla p - \eta \Delta \boldsymbol{\omega}) + \frac{1}{\rho}\eta \Delta \boldsymbol{\omega} + \nabla \times \frac{\boldsymbol{f}}{\rho}\end{aligned} \tag{2.22}$$

Für Flüssigkeiten mit konstantem Druck fällt der zweite Term weg, die dynamische Viskosität wird durch die kinematische Viskosität ersetzt ($\nu = \frac{\eta}{\rho}$) und der Kraftterm ebenfalls entsprechend modifiziert ($\tilde{f} = \frac{f}{\rho}$). $\tilde{f}$ ist somit eine Beschleunigung. Die Wirbeltransportgleichung lautet nun

$$\frac{\mathrm{D}\boldsymbol{\omega}}{\mathrm{D}t} = (\boldsymbol{\omega} \cdot \nabla)\boldsymbol{u} + \nu \Delta \boldsymbol{\omega} + \nabla \times \tilde{\boldsymbol{f}} \tag{2.23}$$

2.2.3. Wirbeltransportgleichung in 2D

Ist das Geschwindigkeitsfeld zweidimensional, dann wird die Wirbelstärke eindimensional, denn

$$\boldsymbol{\omega}(\boldsymbol{x},t) = \nabla \times \boldsymbol{u}(\boldsymbol{x},t) = \nabla \times \begin{pmatrix} u_x \\ u_y \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \partial_x u_y - \partial_y u_x \end{pmatrix} = \omega_z(\boldsymbol{x},t)\boldsymbol{e}_z \tag{2.24}$$

Zusätzlich fällt der Wirbelstreckungsterm weg, denn

$$(\boldsymbol{\omega} \cdot \nabla)\boldsymbol{u} = \omega_z \partial_z u_z = 0 \tag{2.25}$$

Somit lautet schließlich die Wirbeltransportgleichung für inkompressible newtonische Flüssigkeiten

$$\frac{\mathrm{D}}{\mathrm{D}t}\omega_z(\boldsymbol{x},t) = \nu\Delta\omega_z(\boldsymbol{x},t) + F(\boldsymbol{x},t) \tag{2.26}$$

Mit $F(\boldsymbol{x},t) = \partial_x \tilde{f} - \partial_y \tilde{f}$ wird nun die eindimensionale Rotation der äußeren Beschleunigung bezeichnet.

2.3. Fouriertransformation

Die Fouriertransformation ist eine Operation, mit deren Hilfe man eine Funktion in einer Funktionsbasis aus komplexen Exponentialfunktionen darstellen kann.

2.3.1. Definition

Für diese Transformation gibt es verschiedene Definitionen. Die hier verwendete ist

$$\mathcal{F}\left[f(x)\right](\omega) = \int f(x)\mathrm{e}^{-i\omega x}\mathrm{d}x \tag{2.27}$$

Die dazugehörige inverse Transformation ist

$$\begin{aligned} \mathcal{F}^{-}1\left[f(\omega)\right](x) &= \frac{1}{2\pi}\int f(x)\mathrm{e}^{i\omega x}\mathrm{d}\omega \\ &= \frac{1}{2\pi}\iint f(x')\mathrm{e}^{i\omega(x'-x)}\mathrm{d}x'\mathrm{d}\omega \\ &= \iint f(x')\delta(x'-x)\mathrm{d}x'\mathrm{d}\omega \\ &= f(x) \end{aligned} \tag{2.28}$$

2.3.2. Ableitungen im Fourierraum

Ableitungen sind im Fourierraum besonders einfach zu berechnen, denn es gilt

$$\mathcal{F}\left[\frac{\partial}{\partial x}f(x)\right] = \int\left(\frac{\partial}{\partial x}f(x)\right)\mathrm{e}^{-ikx}\mathrm{d}x = ik\int f(x)\mathrm{e}^{ikx}\mathrm{d}x = ik\tilde{f}(k) \quad (2.29)$$

Dabei wurde im zweiten Schritt partiell integriert. Der dabei auftauchende Term fällt weg, da die Funktion $f(x)$ an ihren Integrationsgrenzen periodisch fortgesetzt ist. Eine Ableitung $\frac{\partial}{\partial x}$ wird also im Fourierraum durch einen Faktor ik ersetzt. Dies lässt sich problemlos auf höhere Dimensionen übertragen und die Ersetzungsvorschrift lautet dann $\boldsymbol{\nabla} \rightarrow i\boldsymbol{k}$.

2.3.3. Diskrete Fouriertransformation (DFT)

Auf dem Rechner arbeitet man statt mit kontinuierlichen Integralen mit diskreten Summen. Der Übergang zu Summen bestimmt sich über

$$\int_0^L f(x)\mathrm{d}x \Rightarrow \frac{L}{N}\sum_{n=0}^{N-1} f(\frac{n}{N}L) = \frac{L}{N}\sum_{n=0}^{N-1} f_n \quad (2.30)$$

Eine eindimensionale Fouriertransformation entspricht damit

$$\mathcal{F}\left[f(x)\right] = \int_0^L f(x)\mathrm{e}^{-i\omega x}\mathrm{d}x = \int_0^L f(x)\mathrm{e}^{-i2\pi f}\mathrm{d}x \Rightarrow \frac{L}{N}\sum_{n=0}^{N-1} f_n\mathrm{e}^{-i2\pi k\frac{n}{N}L} = \frac{L}{N}f_k \quad (2.31)$$

Hierbei entspricht f_k der allgemeinen Definition einer diskreten Fouriertransformation von f_n.

Ein Algorithmus, um die n-dimensionale diskrete Fouriertransformation mit Auflösung N^n in einer Laufzeit von $O(N^n log(N))$ zu berechnen, wurde 1965 von Jamey W. Cooley und John W. Tukey vorgestellt [CT65]. Dieser Algorithmus wird als schnelle Fouriertransformation (FFT) bezeichnet.

2.3.4. Zweidimensionale Fouriertransformation

Die zweidimensionale FFT berechnet sich wie folgt:

$$\begin{aligned}&\mathcal{F}_{xy}\left[f(x,y)\right]\\&=\int f(x,y)\mathrm{e}^{-iux}\mathrm{e}^{-ivy}\mathrm{d}x\mathrm{d}y=\mathcal{F}_y\left[\mathcal{F}_x\left[f(x,y)\right]\right]=\mathcal{F}_y\left[\tilde{f}(u,y)\right]=\tilde{\tilde{f}}(u,v)\end{aligned}\tag{2.32}$$

Wie man sieht kann man die zweidimensionale FFT als zwei nacheinander durchgeführte eindimensionale Fouriertransformationen betrachten. Die Diskretisierung folgt analog zum eindimensionalen Fall.

2.3.5. Fouriertransformation einer Gaußfunktion

Eine eindimensionale Gaußfunktion hat die Form

$$f(x)=\frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-\frac{1}{2}\frac{x^2}{\sigma^2}}\tag{2.33}$$

Die Fouriertransformierte ist dann

$$\mathcal{F}\left[f(x)\right]=\frac{1}{\sqrt{2\pi}\sigma}\int\mathrm{e}^{-\frac{1}{2}\frac{x^2}{\sigma^2}}\mathrm{e}^{-ikx}\mathrm{d}x=\frac{1}{\sqrt{2\pi}\sigma}\sqrt{2\pi}\sigma\mathrm{e}^{-\frac{1}{2}\sigma^2k^2}=\mathrm{e}^{-\frac{1}{2}\sigma^2k^2}\tag{2.34}$$

Bei zweimaliger Fouriertransformation ergibt sich dann entsprechend

$$\mathcal{F}\left[\mathcal{F}\left[f(x)\right]\right]=2\pi f(x)\tag{2.35}$$

Für die inverse Fouriertransformation gilt

$$\begin{aligned}\mathcal{F}^{-1}\left[f(k)\right]&=\frac{1}{2\pi}\int\mathrm{e}^{-\frac{1}{2}\sigma^2k^2}\mathrm{e}^{ikx}\mathrm{d}k\\&=\frac{1}{2\pi}\frac{\sqrt{2\pi}}{\sigma}\mathrm{e}^{-\frac{1}{2}\frac{x^2}{\sigma^2}}\\&=\frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-\frac{1}{2}\frac{x^2}{\sigma^2}}\\&=f(x)\end{aligned}\tag{2.36}$$

2.4. Wahrscheinlichkeitsrechnung

Da die Rechnungen in Kapitel 5 auf der Wahrscheinlichkeitsrechnung basieren, sollen hier einige wesentliche Definitionen und Ergebnisse zusammengefasst werden.

Sei Ω die zu betrachtende Ergebnismenge und $X \subset \Omega$ ein Ereignis. Dann bezeichnet $P(X) : \mathcal{P}(\Omega) \to [0, 1]$ die Wahrscheinlichkeit, die dem Ereignis zugeordnet wird. Für das leere bzw. das sichere Ereignis muss gelten

$$P(X = \emptyset) = 0\,, \quad P(X = \Omega) = 1 \tag{2.37}$$

Ist Ω eine kontinuierliche Menge, so ist

$$P(X) = \int_X f(x)\mathrm{d}x \tag{2.38}$$

und $f(x)$ bezeichnet die entsprechende Wahrscheinlichkeitsdichte.

Über diese Definition können ebenfalls mehrdimensonale Ereignisräume $X = (X_1, X_2, ..., X_N)$ gewählt werden. Die Wahrscheinlichkeitsdichte ist dann eine Funktion mehrerer Variablen $f(\boldsymbol{x}) = f(x_1, x_2, ..., x_N)$.

Möchte man die Wahrscheinlichkeit eines Ereignisses unter Voraussetzung eines anderen Ereignisses herausfinden, muss man die Wahrscheinlichkeitsdichte auf einen Unterraum projizieren. Sei $A \subset X$ der Unterraum, auf den projiziert wird und $Y = X \backslash A$ der restliche Raum. Dann definiert folgender Ausdruck die bedingte Wahrscheinlichkeit eines Ereignisses $\boldsymbol{y} \in Y$ bedingt auf ein Ereignis $\boldsymbol{a} \in A$:

$$f(\boldsymbol{y}|\boldsymbol{a}) = \frac{f(\boldsymbol{y} + \boldsymbol{a})}{f(\boldsymbol{a})}, \quad \text{mit } f(\boldsymbol{a}) = \textstyle\int_Y f(\boldsymbol{y} + \boldsymbol{a})\mathrm{d}y \tag{2.39}$$

In der Wahrscheinlichkeitsrechnung wird häufig der Mittelwert einer Funktion von Zufallsvariablen $g(\boldsymbol{x}) : \Omega \to \mathbb{R}$ benötigt. Er ist wie folgt definiert:

$$\langle g(\boldsymbol{x}) \rangle_{\boldsymbol{x}} = \int_\Omega g(\boldsymbol{x}) f(\boldsymbol{x})\mathrm{d}x \tag{2.40}$$

Dabei gibt das tiefer angehängte $\boldsymbol{x}$ an, über welche Variable gemittelt wird.

Die Mittelwerte $\langle x^n \rangle_{\boldsymbol{x}}$ nennt man Momente einer Wahrscheinlichkeitsverteilung. Existieren alle Momente, so bestimmen sie die Verteilung eindeutig. Von insbesonderer Bedeutung ist ebenfalls die charakteristische Funktion $\varphi(\boldsymbol{\alpha})$. Sie kann als Fouriertransformierte der Wahrscheinlichkeitsverteilung aufgefasst werden und kann direkt über die Momente der Verteilung ausgedrückt werden:

$$\varphi(\boldsymbol{\alpha}) = \left\langle \mathrm{e}^{i\boldsymbol{\alpha}\cdot\boldsymbol{x}} \right\rangle_{\boldsymbol{x}} = \sum_{n=0}^{\infty} \frac{i^n}{n!} \left\langle (\boldsymbol{\alpha} \cdot \boldsymbol{x})^n \right\rangle_{\boldsymbol{x}} = (2\pi)^n \mathcal{F}^{-1}\left[f(\boldsymbol{x})\right] \tag{2.41}$$

2.4.1. Charakteristische Funktion einer Gaußverteilung

Eine Gaußverteilung besitzt die Dichte

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-\frac{1}{2}\frac{x^2}{\sigma^2}} \tag{2.42}$$

Die charakteristische Funktion ist dann

$$\begin{aligned} \varphi(\alpha) &= \frac{1}{\sqrt{2\pi}\sigma} \int \mathrm{e}^{-\frac{1}{2}\frac{x^2}{\sigma^2}} \mathrm{e}^{ikx} \mathrm{d}x = \frac{1}{\sqrt{2\pi}\sigma} \sqrt{2\pi}\sigma \mathrm{e}^{-\frac{1}{2}\sigma^2 k^2} \\ &= \mathrm{e}^{-\frac{1}{2}\sigma^2 k^2} = 2\pi \mathcal{F}^{-1}\left[f(x)\right] = \mathcal{F}\left[f(x)\right] \end{aligned} \tag{2.43}$$

2.5. Statistische Eigenschaften der 2D-Turbulenz

Dieser Abschnitt soll nun einige wichtige Definitionen und Ergebnisse statistischer Untersuchungen von 2D-Turbulenz zusammenfassen, die für spätere Kapitel wichtig sind. Ausführlichere Herleitungen und Details finden sich z.B. in [Pop00]. Außerdem bietet [BE12] eine gute kurze Einführung und Übersicht zu 2D-Turbulenz.

2.5.1. Das Geschwindigkeitfeld

Die einzige Zustandsgröße in 2D-Turbulenz ist das Geschwindigkeitsfeld. Es wird im Allgemeinen als $\boldsymbol{u}(\boldsymbol{x}, t)$ bezeichnet. Seine zeitliche Evolution ist im Falle von 2D-Turbulenz vollständig durch die 2D-Wirbeltransportgleichung bestimmt. $\boldsymbol{u}(\boldsymbol{k}, t)$ sei die Fouriertransformierte des Feldes.

Wirbelstärke

Die dem Geschwindigkeitsfeld entsprechenden Wirbelstärken sind $\omega_z(\boldsymbol{x}, t) = \nabla \times \boldsymbol{u}(\boldsymbol{x}, t)$ und $\omega_z(\boldsymbol{k}, t) = i\boldsymbol{k} \times \boldsymbol{u}(\boldsymbol{k}, t)$. Diese Definition zeigt direkt, wie man von der Wirbelstärke zurück zum Geschwindigkeitsfeld transformieren kann:

$$\begin{aligned} \nabla \times \omega_z(\boldsymbol{x}, t)\boldsymbol{e}_z &= \nabla \times (\nabla \times \boldsymbol{u}(\boldsymbol{x}, t)) \\ &= \nabla \left(\nabla \cdot \boldsymbol{u}(\boldsymbol{x}, t)\right) - \Delta \boldsymbol{u}(\boldsymbol{x}, t) \\ &= -\Delta \boldsymbol{u}(\boldsymbol{x}, t) \end{aligned} \tag{2.44}$$

Dabei wurde ausgenutzt, dass $\boldsymbol{u}(\boldsymbol{x}, t)$ inkompressibel ist, also $\nabla \cdot \boldsymbol{u}(\boldsymbol{x}, t) = 0$. Im Fourierraum bedeutet dies

$$i\boldsymbol{k} \times \omega_z(\boldsymbol{k}, t)\boldsymbol{e}_z = \boldsymbol{k}^2 \boldsymbol{u}(\boldsymbol{k}, t) \tag{2.45a}$$

$$\frac{i\boldsymbol{k} \times \omega_z(\boldsymbol{k}, t)\boldsymbol{e}_z}{\boldsymbol{k}^2} = \boldsymbol{u}(\boldsymbol{k}, t) \tag{2.45b}$$

Mittlere Geschwindigkeit

Die mittlere Geschwindigkeit pro Raumrichtung u' bestimmt sich in 2D über

$$u' = \sqrt{\left\langle \frac{1}{2}\boldsymbol{u}(\boldsymbol{x},t)\cdot\boldsymbol{u}(\boldsymbol{x},t)\right\rangle_{\boldsymbol{u}}} = \sqrt{\langle E(\boldsymbol{x},t)\rangle_{\boldsymbol{u}}} = \sqrt{E} \tag{2.46}$$

E ist dabei die mittlere Energie und wird in Abschnitt 2.5.4 definiert.

Autokorrelation und weitere Korrelationsfunktionen

Eine wichtige Größe in der statistischen Turbulenz ist die Zwei-Punkt-Korrelation. Sie ist definiert als

$$R_{ij}(\boldsymbol{r},t) = \langle u_i(\boldsymbol{x}+\boldsymbol{r},t)u_j(\boldsymbol{x},t)\rangle_{\boldsymbol{u}} \tag{2.47}$$

Die Indizes i,j geben dabei die Raumrichtungen des Geschwindigkeitsfeldes an, nach dem die Korrelationsfunktion ausgewertet werden soll. Die Zwei-Punkt-Korrelation lässt sich aus Gründen der Isotropie in zwei skalare Autokorrelations-Funktionen zerlegen (siehe auch [Pop00]):

$$R_{ij}(\boldsymbol{r},t) = u'^2\left(g(r,t)\delta_{ij} + [f(r,t)-g(r,t)]\,\frac{r_i r_j}{r^2}\right) \tag{2.48}$$

Dies kann man nach $f(r,t)$ und $g(r,t)$ auflösen und erhält

$$f(r,t) = \frac{\langle u_x(\boldsymbol{x}+r\boldsymbol{e}_x,t)u_x(\boldsymbol{x},t)\rangle_{\boldsymbol{u}}}{\langle u_x^2\rangle_{\boldsymbol{u}}} \tag{2.49a}$$

$$g(r,t) = \frac{\langle u_y(\boldsymbol{x}+r\boldsymbol{e}_x,t)u_y(\boldsymbol{x},t)\rangle_{\boldsymbol{u}}}{\langle u_y^2\rangle_{\boldsymbol{u}}} = f(r,t) + \frac{1}{2}r\frac{\partial}{\partial r}f(r,t) \tag{2.49b}$$

Dabei ist die transversale Autokorrelation $g(r,t)$ komplett über die longitudinale Autokorrelation $f(r,t)$ bestimmt.

Häufig sieht man auch Mehrfachkorrelationen der Form $\langle u_i(\boldsymbol{x},t)u_j(\boldsymbol{x},t)u_k(\boldsymbol{x}+\boldsymbol{r},t)\rangle_{\boldsymbol{u}}$. In [KH38] haben Kármán und Howarth, aufbauend auf den Arbeiten von Taylor, deren zeitliche Evolution genauer untersucht. Genauso wie für das Geschwindigkeitsfeld kann man die Autokorrelation und die weiteren Korrelationsfunktionen auch für die Wirbelstärke auswerten.

Geschwindigkeitsinkrement und Sturkturfunktion

Eine andere wichtige Größe ist das Geschwindigkeitsinkrement, dessen Momente als Strukturfunktionen S_n bezeichnet werden:

$$S_n = \langle |\boldsymbol{u}(\boldsymbol{x}+\boldsymbol{r},t) - \boldsymbol{u}(\boldsymbol{x})|^n \rangle_{\boldsymbol{u}} \tag{2.50}$$

Für einige dieser Momente leitete Kolmogorov in seinen Arbeiten [Kol41c], [Kol41b] und [Kol41a] Skalengesetze her, darunter das berühmte $\frac{4}{5}$-Gesetz für 3D-Turbulenz. Es bezieht sich allerdings auf das longitudinale Geschwindigkeitsinkrement, also die Projektion u_r von $\boldsymbol{u}$ auf den Abstandsvektor $\boldsymbol{r}$.

$$S_3^{(L)} = \left\langle \left(u_r(\boldsymbol{x}+\boldsymbol{r},t) - u_r(\boldsymbol{x})\right)^3 \right\rangle_{\boldsymbol{u}} = -\frac{4}{5} E\,|\boldsymbol{r}| \tag{2.51}$$

Hier taucht die mittlere Energie E auf, die in Abschnitt 2.5.4 definiert wird.

2.5.2. Erweiterte Wirbeltransportgleichung

Die Wirbeltransportgleichung (2.26) wird nun um einige Parameter oder Annahmen erweitert, die sich später für die Simulation als nützlich erweisen. Die genaue Form ist hierbei

$$\begin{aligned}\frac{\mathrm{D}}{\mathrm{D}t}\omega_z(\boldsymbol{x},t) &= \mathcal{L}\omega_z(\boldsymbol{x},t) + F(\boldsymbol{x},t)\\ &= \left((-1)^{m+1}\mu\,(k_\mu)^{2m}\,\Delta^{-m} - \gamma + (-1)^{n+1}\nu\,(k_\nu)^{-2n}\,\Delta^{n}\right)\omega_z(\boldsymbol{x},t)\\ &\quad + F(\boldsymbol{x},t)\end{aligned} \tag{2.52}$$

Dabei ist Δ^{-1} der inverse Laplace-Operator. Diese Gleichung kann zwar auch komplett im Fourierraum hingeschrieben werden, wie später in Kapitel 6.1 gezeigt wird. Hier soll aber zunächst eine Schreibweise gewählt werden, wie sie für ein Pseudospektralverfahren (siehe Kapitel 3.4) verwendet werden kann. Diese hat dann die Form

$$\begin{aligned}&\frac{\mathrm{d}}{\mathrm{d}t}\omega_z(\boldsymbol{k},t) + \mathcal{F}\left[\boldsymbol{u}(\boldsymbol{x},t)\cdot\nabla\omega_z(\boldsymbol{x},t)\boldsymbol{e}_z\right]\\ &= \mathcal{L}\omega_z(\boldsymbol{k},t) + F(\boldsymbol{k},t)\\ &= \left(-\mu\left(\frac{(k_\mu)^2}{\boldsymbol{k}^2}\right)^m - \gamma - \nu\left(\frac{\boldsymbol{k}^2}{(k_\nu)^2}\right)^n\right)\omega_z(\boldsymbol{k},t) + F(\boldsymbol{k},t)\end{aligned} \tag{2.53}$$

Hier ist der lineare Term $\mathcal{L}$ durch die Hypoviskosität μ, die Hyperviskosität ν und einen konstanten Reibungsterm γ ersetzt worden. An den Modenzahlen k_μ und k_ν bleibt durch die gewählte Definition die Viskosität bei Änderung der Exponenten m und n gleich. Effektiv werden hierdurch nur die Viskositäten skaliert. Höhere Potenzen bei den Viskositäten sorgen dafür, dass die Viskosität an den Randbereichen der Simulation sehr viel stärkere Auswirkungen hat und in der Mitte vernachlässigbar ist. Dadurch wird der sogenannte inertiale Bereich möglichst groß, in dem sich die Energiekaskade und die Enstrophiekaskade ausbilden (siehe Abschnitt 2.5.4). Für detailliertere Analysen von Hyper- und Hypoviskosität siehe z.B. [LCP05] oder [KC12].

2.5.3. Kraftterm

Der Kraftterm ist üblicherweise auf wenige Fouriermoden beschränkt und kann verschiedene Formen annehmen. Zwei davon sollen hier vorgestellt werden. Beide Kräfte sollen nur auf einem schmalen Kreis aus Fouriermoden wirken. Der Kreis soll dabei den Radius k_f haben und Breite k_A. Die hier definierte Funktion Θ liefert 1 bei einer Mode innerhalb des so definierten Kreises.

$$\Theta(\boldsymbol{k}) = \begin{cases} 1 & |\boldsymbol{k} - \boldsymbol{k}_f| \leq k_A \\ 0 & \text{sonst} \end{cases} \tag{2.54}$$

Jede der vorgestellten Kräfte erzeugt die in Abschnitt 2.5.4 beschriebene inverse Energiekaskade und die direkte Enstrophiekaskade.

Stochastische Kraft

Die eine Variante ist die sogenannte stochastische Kraft der Form

$$F_s(\boldsymbol{k}, t) = f_A \Theta(\boldsymbol{k}) \mathrm{e}^{i\phi(t)} \tag{2.55}$$

Sie ist zeitlich delta-korreliert, also $\langle F_s(\boldsymbol{k}, t) F_s(\boldsymbol{k}, t') \rangle_{\boldsymbol{k}} \propto \delta(t - t')$. Dies wird in der Simulation dadurch erreicht, dass die Phase $\phi(t)$ des Kraftfeldes zu jedem Simulationsschritt aus Zufallsdaten neu bestimmt wird, während die Amplitude in Abhängigkeit der Mode $\boldsymbol{k}$ fest vorgegeben ist.

Deterministische Kraft

Die andere Variante ist die sogenannte deterministische Kraft. Sie lässt sich nicht durch einen solch einfachen Ausdruck beschreiben. Ihre Funktionsweise ist derart, dass die Amplitude der Fouriermoden auf einem schmalen Kreis in jedem Simulationsschritt

konstant auf der vorgegeben Amplitude f'_A gehalten wird, aber die Phase vollständig deterministisch durch die Wirbeltransportgleichung bestimmt wird. Also $\frac{\mathrm{d}}{\mathrm{d}t}\left|\omega(\boldsymbol{k},t)\right| = 0$ bzw. $\left|\omega(\boldsymbol{k},t)\right| = f'_A$, falls $\Theta(\boldsymbol{k}) = 1$. Im Folgenden wird die Kraft-Amplitude für beide Kräfte stets f_A genannt, obwohl beide Amplituden stark unterschiedliche Bedeutungen (und auch Einheiten) haben und nicht miteinander vergleichbar sind.

2.5.4. Energie- und Enstrophie-Kaskade

Die mittlere Energie pro Einheitsmasse des Geschwindigkeitsfeldes wird definiert als

$$E(t) = \frac{1}{2}\left\langle \left|\boldsymbol{u}(\boldsymbol{x},t)\right|^2 \right\rangle_{\boldsymbol{x}} = \int_0^\infty E(\boldsymbol{k},t)\mathrm{d}^n k \tag{2.56}$$

mit

$$\begin{aligned} \text{2D:} \qquad E(\boldsymbol{k},t) &= \frac{1}{2}\int \left|\boldsymbol{u}(\boldsymbol{k},t)\right|^2 \mathrm{d}\Omega \\ &= \frac{1}{2}\int \left|\boldsymbol{u}(\boldsymbol{k},t)\right|^2 k\mathrm{d}\varphi \end{aligned} \tag{2.57a}$$

$$\begin{aligned} \text{3D:} \qquad E(\boldsymbol{k},t) &= \frac{1}{2}\int \left|\boldsymbol{u}(\boldsymbol{k},t)\right|^2 \mathrm{d}\Omega \\ &= \frac{1}{2}\iint \left|\boldsymbol{u}(\boldsymbol{k},t)\right|^2 k^2 \sin\theta \mathrm{d}\varphi\mathrm{d}\theta \end{aligned} \tag{2.57b}$$

und die mittlere Enstrophie über

$$\mathcal{E}(t) = \frac{1}{2}\left\langle \left|\boldsymbol{\omega}(\boldsymbol{x},t)\right|^2 \right\rangle_{\boldsymbol{x}} \tag{2.58}$$

Im Falle von zwei Dimensionen untersuchte Kraichnan in seinen Arbeiten [Kra67] und [Kra71] den Energietransfer und Enstrophietransfer genauer, der zu der Energie-Kaskade (inverse Kaskade) und der Enstrophie-Kaskade (direkte Kaskade) führt. Die Kaskaden lassen sich über folgende Gleichungen beschreiben:

$$E(\boldsymbol{k}) = C\epsilon^{\frac{2}{3}}\boldsymbol{k}^{-\frac{5}{3}} \tag{2.59a}$$

$$E(\boldsymbol{k}) = C'\eta^{\frac{2}{3}}\boldsymbol{k}^{-3} \tag{2.59b}$$

C und C' sind Konstanten, ϵ ist die Energiedissipationsrate und η die Enstrophiedissipationrate. Die Energiekaskade transportiert Energie von großen Wellenzahlen hin zu kleinen Wellenzahlen und die Enstrophiekaskade transportiert Enstrophie von kleinen

Wellenzahlen zu großen. In Abbildung 2.1 sieht man die gleichzeitige Existenz beider Kaskaden. Entscheidend zur Kaskadenbildung ist der Energiefluss bzw. der Enstrophiefluss. Die Simulation hatte hierbei noch nicht den stationären Zustand erreicht und deshalb weicht der Exponent für die Enstrophiekaskade vom erwarteten Wert ab. Für eine Dikussion, warum der Exponent der Energiekaskade betragsmäßig größer ist als $-\frac{5}{3}$ siehe Kapitel 4.5.1. Die eingezeichneten Längenskalen werden in Kapitel 2.5.5 näher erläutert.

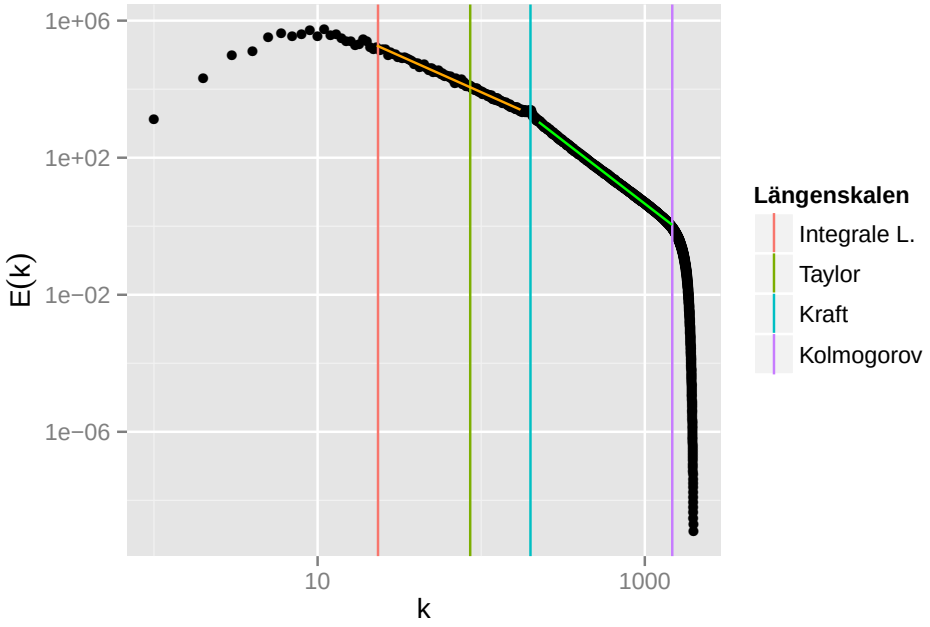


Abbildung 2.1.: Gleichzeitige Existenz der Energiekaskade (orange) mit Exponent $-2,13$ sowie der Enstrophie-Kaskade (grün) mit Exponent $-3,67$ in einer Simulation, die noch nicht den stationären Zustand erreicht hat. (Simulation: `4096both`, siehe Anhang A)

Energiefluss

Zum Ausrechnen des Flusses benötigt man man zunächst die Wirbeltransportgleichung (2.53) ohne Viskosität und ohne Kraft, um allein den Transfer-Term zu untersuchen:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\omega_z(\boldsymbol{k},t)\boldsymbol{e}_z &= -\mathcal{F}\left[\boldsymbol{u}(\boldsymbol{x},t)\cdot\nabla\omega_z(\boldsymbol{x},t)\boldsymbol{e}_z\right]\\ &= \mathcal{F}\left[\nabla\times(\boldsymbol{u}(\boldsymbol{x},t)\times\omega_z(\boldsymbol{x},t)\boldsymbol{e}_z)\right]\\ &= i\boldsymbol{k}\times\mathcal{F}\left[\boldsymbol{u}(\boldsymbol{x},t)\times\omega_z(\boldsymbol{x},t)\boldsymbol{e}_z\right]\end{aligned} \tag{2.60}$$

Nun betrachtet man die zeitliche Änderung der Energie

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}E(\boldsymbol{k},t) &= \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(\boldsymbol{u}^*(\boldsymbol{k},t)\cdot\boldsymbol{u}(\boldsymbol{k},t)\right)\\ &= \mathrm{Re}\left(\boldsymbol{u}^*(\boldsymbol{k},t)\cdot\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}(\boldsymbol{k},t)\right)\end{aligned} \tag{2.61}$$

und rechnet mit Hilfe von Gleichung (2.45b) die Ableitung der Geschwindigkeit aus:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}(\boldsymbol{k},t) &= \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}(\boldsymbol{k},t)\\ &= \frac{i\boldsymbol{k}}{\boldsymbol{k}^2}\times\frac{\mathrm{d}}{\mathrm{d}t}\omega_z(\boldsymbol{k},t)\boldsymbol{e}_z\\ &= -\frac{\boldsymbol{k}}{\boldsymbol{k}^2}\times\left(\boldsymbol{k}\times\mathcal{F}\left[\boldsymbol{u}(\boldsymbol{x},t)\times\omega_z(\boldsymbol{x},t)\boldsymbol{e}_z\right]\right)\\ &= \mathcal{F}\left[\boldsymbol{u}(\boldsymbol{x},t)\times\omega_z(\boldsymbol{x},t)\boldsymbol{e}_z\right]-\frac{\mathcal{F}\left[\boldsymbol{u}(\boldsymbol{x},t)\times\omega_z(\boldsymbol{x},t)\boldsymbol{e}_z\right]\cdot\boldsymbol{k}}{\boldsymbol{k}^2}\boldsymbol{k}\\ &= \mathcal{P}(\boldsymbol{k})\mathcal{F}\left[\boldsymbol{u}(\boldsymbol{x},t)\times\omega_z(\boldsymbol{x},t)\boldsymbol{e}_z\right]\end{aligned} \tag{2.62}$$

Im vorletzten Schritt wurde hierbei die Regel für das doppelte Kreuzprodukt angewendet

$$\boldsymbol{k}\times(\boldsymbol{k}\times\boldsymbol{a}) = \boldsymbol{k}\left(\boldsymbol{k}\cdot\boldsymbol{a}\right)-\boldsymbol{a}\left(\boldsymbol{k}\cdot\boldsymbol{k}\right) = \left(\boldsymbol{k}\cdot\boldsymbol{a}\right)\boldsymbol{k}-\boldsymbol{a}\boldsymbol{k}^2 \tag{2.63}$$

Weiterhin wurde Projektionsoperator $\mathcal{P}(\boldsymbol{k})$ benutzt, der ein Vektorfeld auf seinen divergenzfreien Anteil projiziert.

$$\mathcal{P}(\boldsymbol{k})\boldsymbol{F} = \left(\mathbf{1}-\boldsymbol{e}_{\boldsymbol{k}}\boldsymbol{e}_{\boldsymbol{k}}\cdot\right)\boldsymbol{F} \tag{2.64}$$

Insgesamt ergibt sich damit die Energieänderung

$$\begin{aligned}T(k,t) &= \int\frac{\mathrm{d}}{\mathrm{d}t}E(\boldsymbol{k},t)\mathrm{d}\Omega\\ &= \int\mathrm{Re}\left(\boldsymbol{u}^*(\boldsymbol{x},t)\cdot\mathcal{P}(\boldsymbol{k})\mathcal{F}\left[\boldsymbol{u}(\boldsymbol{x},t)\times\omega_z(\boldsymbol{x},t)\boldsymbol{e}_z\right]\right)\mathrm{d}\Omega\end{aligned} \tag{2.65}$$

Der Energiefluss $\Pi(k)$ errechnet sich über Aufintegrieren der Energieänderung und Ensemblemittelung

$$\Pi(k) = \left\langle\int_k^\infty T(k')\mathrm{d}k'\right\rangle_{\boldsymbol{u}} \tag{2.66}$$

Da ohne Kraft und Viskosität die Energie erhalten bleibt, muss gelten

$$\Pi(0) = \left\langle\int_0^\infty T(k')\mathrm{d}k'\right\rangle_{\boldsymbol{u}} = 0 \tag{2.67}$$

Enstrophiefluss

Die Rechnung zum Enstrophiefluss verläuft ähnlich der zum Energiefluss. Zunächst berechnet man wieder die zeitliche Änderung der Enstrophie

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{E}(\boldsymbol{k},t) &= \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(\omega_z^*(\boldsymbol{k},t)\omega_z(\boldsymbol{k},t)\right) \\ &= \mathrm{Re}\left(\omega_z^*(\boldsymbol{k},t)\frac{\mathrm{d}}{\mathrm{d}t}\omega_z(\boldsymbol{k},t)\right)\end{aligned} \tag{2.68}$$

Dies führt unter Einsetzen von Gleichung (2.60) auf

$$\begin{aligned}U(k,t) &= \int \frac{\mathrm{d}}{\mathrm{d}t}\mathcal{E}(\boldsymbol{k},t)\mathrm{d}\Omega \\ &= \int \mathrm{Re}\left[\omega_z^*(\boldsymbol{k},t)\boldsymbol{e}_z \cdot i\boldsymbol{k}\times\mathcal{F}\left[\boldsymbol{u}(\boldsymbol{x},t)\times\omega_z(\boldsymbol{x},t)\boldsymbol{e}_z\right]\right]\mathrm{d}\Omega \\ &= \int \mathrm{Re}\left[\omega_z^*(\boldsymbol{k},t)\mathcal{F}\left[\boldsymbol{u}(\boldsymbol{x},t)\times\omega_z(\boldsymbol{x},t)\boldsymbol{e}_z\right]\cdot(\boldsymbol{e}_z\times i\boldsymbol{k})\right]\mathrm{d}\Omega\end{aligned} \tag{2.69}$$

Den Enstrophiefluss $\Lambda(k)$ berechnet man ebenso wie den Energiefluss über Aufintegrieren und Mittelung

$$\Lambda(k) = \left\langle \int_0^k U(k')\mathrm{d}k' \right\rangle_{\boldsymbol{u}} \tag{2.70a}$$

$$\Lambda(0) = \left\langle \int_0^\infty U(k')\mathrm{d}k' \right\rangle_{\boldsymbol{u}} = 0 \tag{2.70b}$$

Die zweite Gleichung ist Resultat der Enstrophieerhaltung ohne äußere Kraft oder Viskosität.

Energie- und Enstrophiedissipationsrate

Die Energie wird in der modifizierten Navier-Stokes-Gleichung (2.53) ausschließlich durch den linearen Term $\mathcal{L}$ dissipiert. Dessen Terme wirken auf unterschiedlichen Modenbereichen besonders stark. Die Hypoviskosität μ wirkt besonders bei kleinen k und die Hyperviskosität bei besonders großen k. Das Besondere ist, dass durch den linearen Term im Fourierraum keine Mischung der einzelnen Moden stattfindet. Dieser besitzt also nur einen indirekten Einfluss auf die Bildung der Energie- und

Enstrophiekaskade. Für jeden der Terme kann man eine eigene Energiedissipationsrate errechnen. Für die mittelere Energiedissipationsrate gilt

$$\begin{aligned}\epsilon = \langle \epsilon(\boldsymbol{x},t)\rangle_{\boldsymbol{u},\boldsymbol{x}} &= \left\langle \frac{\mathrm{d}}{\mathrm{d}t} E(\boldsymbol{x},t)\right\rangle_{\boldsymbol{u},\boldsymbol{x}} \\ &= \left\langle \boldsymbol{u}(\boldsymbol{x},t)\cdot\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}(\boldsymbol{x},t)\right\rangle_{\boldsymbol{u},\boldsymbol{x}} = \langle \boldsymbol{u}(\boldsymbol{x},t)\cdot\mathcal{L}\boldsymbol{u}(\boldsymbol{x},t)\rangle_{\boldsymbol{u},\boldsymbol{x}}\end{aligned} \tag{2.71}$$

In dieser Umformung wurde lediglich der lineare Term der Navier-Stokes-Gleichung berücksichtigt, da nur dieser zur Energiedissipation beiträgt. Interessiert man sich nun für die mittlere Energiedissipationsrate durch die Hyperviskosität bzw. Hypoviskosität, so berücksichtigt man vom linearen Term nur den hyper- bzw. hypoviskosen Anteil und kommt auf

$$\epsilon_\nu = \left\langle \boldsymbol{u}(\boldsymbol{x},t)\cdot(-1)^{n+1}\nu\Delta^n\boldsymbol{u}(\boldsymbol{x},t)\right\rangle_{\boldsymbol{u},\boldsymbol{x}} \tag{2.72a}$$

$$\epsilon_\mu = \left\langle \boldsymbol{u}(\boldsymbol{x},t)\cdot(-1)^{m+1}\mu\Delta^{-m}\boldsymbol{u}(\boldsymbol{x},t)\right\rangle_{\boldsymbol{u},\boldsymbol{x}} \tag{2.72b}$$

Genauso lässt sich die mittlere Enstrophiedissipationsrate definieren:

$$\begin{aligned}\eta = \langle \epsilon(\boldsymbol{x},t)\rangle_{\boldsymbol{u},\boldsymbol{x}} &= \left\langle \frac{\mathrm{d}}{\mathrm{d}t}\mathcal{E}(\boldsymbol{x},t)\right\rangle_{\boldsymbol{u},\boldsymbol{x}} \\ &= \left\langle \omega_z(\boldsymbol{x},t)\frac{\mathrm{d}}{\mathrm{d}t}\omega_z(\boldsymbol{x},t)\right\rangle_{\boldsymbol{u},\boldsymbol{x}} = \langle \omega_z(\boldsymbol{x},t)\cdot\mathcal{L}\omega_z(\boldsymbol{x},t)\rangle_{\boldsymbol{u},\boldsymbol{x}}\end{aligned} \tag{2.73a}$$

$$\eta_\nu = \left\langle \omega_z(\boldsymbol{x},t)(-1)^{n+1}\nu\Delta^n\omega_z(\boldsymbol{x},t)\right\rangle_{\boldsymbol{u},\boldsymbol{x}} \tag{2.73b}$$

$$\eta_\mu = \left\langle \omega_z(\boldsymbol{x},t)(-1)^{m+1}\mu\Delta^{-m}\omega_z(\boldsymbol{x},t)\right\rangle_{\boldsymbol{u},\boldsymbol{x}} \tag{2.73c}$$

2.5.5. Längenskalen

Integrale Länge

Historisch stammen die ersten Arbeiten zur statistischen Beschreibung Turbulenz von Taylor ([Tay35]). Er definierte in seiner Arbeit ähnlich wie Prandl zuvor und später Kolmogorov für Turbulenz charakteristische Längenskalen. Eine davon ist die integrale Längenskala. Sie ist ein Maß für die größten Wirbelstrukturen.

$$L_{xx}(t) = \int_0^\infty f(r,t)\mathrm{d}r = \frac{1}{\langle u_x^2\rangle_{\boldsymbol{u}}}\int_0^\infty \langle u_x(\boldsymbol{x}+r\boldsymbol{e}_x,t)u_x(\boldsymbol{x},t)\rangle_{\boldsymbol{u}}\,\mathrm{d}r \tag{2.74}$$

Die Korrelationsfunktion, über die integriert wird, hat bei $r = 0$ ihr Maximum und fällt bei größeren Abständen auf 0. Durch Integration über diese Korrelationsfunktion erhält man eine Länge, die etwa einer mittleren Wirbelgröße entspricht, also großskalige Strukturen beschreibt.

Über die integrale Länge lässt sich auch eine Zeit definieren, die integrale Zeitskala. Dies geschiet über die mittlere Geschwindigkeit u':

$$T_L = \frac{\langle L_{xx} \rangle}{u'} \tag{2.75}$$

Taylor-Länge

Eine weitere Länge, die Taylor in seiner Arbeit [Tay35] einführte, ist die sogenannte Taylor-Länge. Die Taylor-Länge sollte ursprünglich die Größe der kleinsten Wirbel beschreiben. Dies stimmt jedoch für hohe Reynoldszahlen nicht mehr. Vielmehr befindet sich die Taylor-Länge zwischen der Kolmogorov-Länge λ_η und der integralen Länge L. Ihre Definition ist

$$\lambda_\tau = \sqrt{\frac{2u'^2}{\left\langle \frac{\mathrm{d}}{\mathrm{d}x} \boldsymbol{u}(\boldsymbol{x}, t) \cdot \boldsymbol{e}_x \right\rangle_{\boldsymbol{u}}}} \tag{2.76}$$

Die mittlere Geschwindigkeit u' wurde zuvor in Gleichung (2.46) definiert. Im Gegensatz zu den anderen Längen ist die Taylor-Länge allerdings einzig über das Geschwindigkeitsfeld definiert und benötigt keine weiteren Parameter aus der Navier-Stokes-Gleichung.

Kolmogorov-Länge

Von etwas größerer Bedeutung ist die Kolmogorov-Längenskala. In [Kol41c] stellt Kolmogorov die Hypothese auf, dass bei hohen Reynoldszahlen die turbulente Bewegung statistisch isotrop ist und diese allein durch die Viskosität und die Energiedissipation bestimmt ist. Dementsprechend definiert er die Kolmogorov-Länge, die etwa mit der kleinsten Größe der Wirbel übereinstimmt, ausschließlich über diese beiden Parameter der Navier-Stokes-Gleichung. Seine Definition ist

$$\lambda_\eta = \left(\frac{\nu^3}{\epsilon_\nu} \right)^{\frac{1}{4}} \tag{2.77}$$

Möchte man nun auch Hyperviskosität berücksichtigen, benötigt man eine angepasste Variante, die auch z.B. in [LCP05] verwendet wird. Zur Herleitung benutzt man eine

Dimensionsanalyse. Zunächst bestimmt man die Einheit der Hyperviskosität ν. Es muss gelten

$$[\epsilon_\nu] = \left[\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}^2\right] = \frac{m^2}{s^3} \stackrel{!}{=} [\boldsymbol{u} \cdot \nu\Delta^n\boldsymbol{u}] = \frac{m}{s}\,[\nu]\,\frac{1}{m^{2n}}\frac{m}{s} = [\nu]\,\frac{1}{s^2 m^{2n-2}} \tag{2.78}$$

Und damit folgt dann

$$[\nu] = \frac{m^{2n}}{s} \tag{2.79}$$

Die Kolmogorov-Länge mit Hyperviskosität λ_η soll die Einheit Länge haben und für $n = 1$ die originale Definition erfüllen. Daher wählt man den Ansatz

$$[\lambda_\eta] = m \stackrel{!}{=} \left[\left(\frac{\nu^3}{\epsilon_\nu}\right)^{\frac{1}{l}}\right] = \left(\frac{\left(\frac{m^{2n}}{s}\right)^3}{\frac{m^2}{s^3}}\right)^{\frac{1}{l}} = m^{\frac{6n-2}{l}} \tag{2.80}$$

Die vollständige Definition lautet damit

$$\lambda_\eta = \left(\frac{\nu^3}{\epsilon_\nu}\right)^{\frac{1}{6n-2}} \tag{2.81}$$

2.5.6. Reynoldszahl

Im Zusammenhang mit Hypo- und Hyperviskosität lässt sich die normale Definition der Reynoldszahl $Re = \frac{\lambda_\tau u'}{\nu}$ nicht benutzen, was bereits aus Dimensionsgründen klar ist. Stattdessen könnte man eine Definition verwenden, welche die Reynoldszahl mit der integralen Länge und der Kolmogorov-Länge verbindet [Pop00]

$$Re_L = \left(\frac{L_x x}{\eta}\right)^{\frac{4}{3}} \tag{2.82}$$

Dies liefert sehr kleine Reynoldszahlen im Bereich $\approx 10^1$ und ist unabhängig von den Exponenten n und m der Hyper- bzw. Hypoviskosität. Eine andere Definition ist die unter anderem in [LCP05] benutzte

$$Re_\nu = \left(\frac{k_\eta}{k_f}\right)^{\frac{6n-2}{3}} \tag{2.83}$$

k_η ist hierbei die der Kolmogorov-Länge entsprechende Wellenzahl mit $k_\eta = \frac{2\pi}{\lambda_\eta}$. Diese Definition hat den Nachteil, dass die Reynoldszahl exponentiell in n anwächst, obwohl der Turbulenzgrad ab einem bestimmten Exponenten nicht weiter sichtbar steigt. Eine weitere Dikussion zu Reynoldszahlen in Turbulenz mit Hyperviskosität findet sich in [KC12].

2.6. Grafikbeschleuniger und deren Programmierung

Auf Grund der besonderen Hardwarearchitektur und der daraus resultierenden besonderen Rechenleistung von Grafikbeschleunigern findet man diese momentan als Alternative zu großen CPU-Clustern. Der Grund hierfür ist einfach, denn ein moderner Grafikbeschleuniger (z.B. Tesla K20 mit GK110, Kepler Architektur) berechnet ca. 3,52 TFlops (Operationen mit Gleitkommazahlen pro Sekunde) in `float`-Genauigkeit und ca. 1,17 TFlops in `double`-Genauigkeit, während eine moderne CPU (Intel Core i7 3770K) mit 4 Kernen bei ca. 115GFlops in `double`-Genauigkeit liegt. Außerdem beträgt die Speicherbandbreite von GPUs bis zu 208GiB/s, während eine typische CPU mit DDR3-2133 RAM ein theoretisches Maximum von nur 17GiB/s hat. Um nach diesen groben Werten an die Leistung einer Grafikkarte heranzukommen, bräuchte man ca. 10 dieser CPUs.

Hier soll nun eine kurze Übersicht über den Aufbau und die Programmierung von Grafikbeschleunigern folgen, die beschreibt, warum diese eine Alternative zur Berechnung parallisierbarer Probleme auf CPU-Clustern darstellen. Eine detailliertere Einführung zur Programmierung von Grafikbeschleunigern findet man in [NVIa] oder [Kir07]. Als Hardware-Referenz zur Beschreibung und zum Vergleich dient hier ein moderner Telsa K20 Grafikbeschleuniger, der für wissenschaftliche Rechnungen konzipiert ist.

2.6.1. GPU-Architektur

Der Hardwareaufbau einer GPU unterscheidet sich grundlegend von der einer CPU. Eine moderne CPU besteht aus mehreren Prozessor-Kernen, die parallel verschiedene Threads ausführen können. Man spricht hier von einer MIMD-Struktur (Multiple Instruction Multiple Data). Im Gegensatz dazu besteht eine GPU aus vielen parallel arbeitenden SIMD (Single Instruction Multiple Data) Multiprozessoren. Außerdem existiert auf GPUs zumeist keine Verzweigungsvorhersage und die Instruktionspipelines sind sehr kurz. Beim Warten auf Lese- oder Schreiboperationen auf den globalen Speicher der GPU wird daher zwischen verschiedenen Threads umgeschaltet, bis die Daten zur Verfügung stehen. Der L1-Cache der GPU ist teilweise programmierbar. Das heißt, man kann per Instruktionen Daten in den Cache lesen und von verschiedenen Threads darauf zugreifen.

Diese Architektur erreicht ihre volle Geschwindigkeit allerdings nur, wenn der Algorithmus gut parallelisierbar ist, per Hand optimiert wird und im Vergleich zu den Berechnungen möglichst wenig Speicherzugriffe benötigt.

2.6.2. Organisation der GPU

Die Multiprozessoren der GPU (auch genannt SM für „Streaming Multiprocessor“) sind so organisiert, dass sie jeweils einen Block aus Threads ausführen. Diese Blöcke

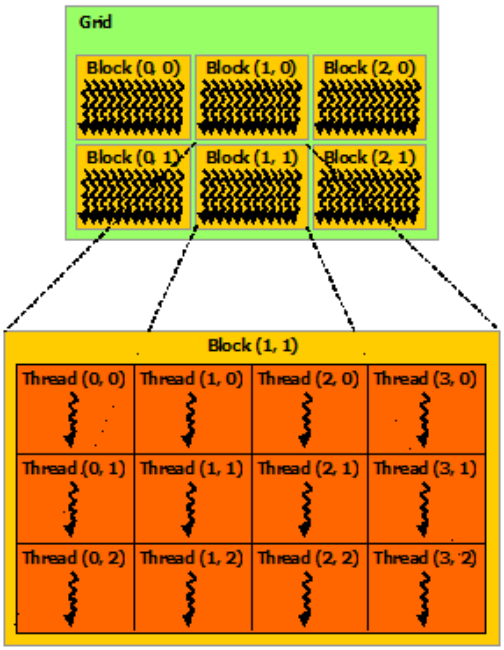


Abbildung 2.2.: Organisation der Threads in ein Gitter aus Blöcken

sind wiederum in ein Gitter eingebettet (Abbildung 2.2) und werden nacheinander auf die Multiprozessoren verteilt. Beim Aufruf einer Funktion auf der GPU muss der Programmierer spezifizieren, wieviele Blöcke er starten möchte und wieviele Threads jeder Block enthalten soll. Die Gitter- bzw. Blockgröße kann hierbei dreidimensional gewählt werden. Die Anzahl der Threads pro Block ist dabei je nach Hardware auf 512 oder 1024 beschränkt. Für die einzelnen Dimensionen der Blöcke und des Gitters gibt es weitere Beschränkungen, die in [NVIa] nachgelesen werden können. Jeder Multiprozessor kann nicht alle seiner Threads gleichzeitig ausführen, sondern lässt diese in kleineren Gruppen à 32 Threads, sogenannten Warps in SIMD-Struktur laufen. Während der Ausführung eines Threads ist es möglich, den Index innerhalb des Blocks bzw. Gitters abzufragen. Somit kann jeder Thread bei gleichem Programmcode auf unterschiedlichem Speicher arbeiten.

2.6.3. Speicherstruktur der GPU

Der Speicherstruktur eines Grafikbeschleunigers ist ähnlich wie bei einer normalen CPU aufgebaut. Es gibt einen globalen Speicher (in Abbildung 2.3 DRAM genannt), der auf aktueller Hardware einige GiB groß ist. Dieser Speicher ist über den PCIe-Bus (PCI Express-Bus) an den normalen Hauptspeicher des Computers (auch genannt Host-Speicher) angebunden oder erlaubt darüber auch Datenaustausch mit anderen

	GeForce 480 GTX	GeForce 680 GTX
Host → Host	5598,04 MiB/s	12517,1 MiB/s
Host → Global	5095,19 MiB/s	10815,9 MiB/s
Global → Global	66213,1 MiB/s	64846,7 MiB/s
Global GPU 1 $\overset{\text{Normal}}{\rightarrow}$ Global GPU 2	2971,32 MiB/s	-
Global GPU 1 $\overset{\text{Peer}}{\rightarrow}$ Global GPU 2	4745,45 MiB/s	-

Tabelle 2.1.: Datenübertragungsgeschwindigkeiten auf einem Rechner mit mehreren GeForce 480 GTX und auf einem anderen Rechner mit einer Geforce 680 GTX

GPUs. PCIe 3.0 ermöglicht auf einem 16x Link momentan eine Datentransferrate von maximal 32GiB/s. Diese Kapazität liegt dabei deutlich über den 17GiB/s, die ein aktuelles DDR3-2133 RAM leistet. Die zuvor erwähnte Tesla K20 GPU greift auf ihren globalen Speicher mit bis zu 208GiB/s zu.

Um einen kleinen Überblick über „echte“ gemessene Geschwindigkeiten zu bekommen sind in Tabelle 2.1 die Datenübertragungsgeschwindigkeiten einer GeForce 480 GTX und einer GeForce 680 GTX angegeben, die in zwei unterschiedlichen Rechnern eingebaut waren. Die Datenübertragungsgeschwindigkeit des globalen Speichers ist hier ca. 12x bzw. 5x so groß wie die des Hostspeichers. Die Datenübertragungsrate zwischen zwei Grafikkarten ist im normalen Modus nur gut halb so schnell wie der Datenzugriff auf den Hostspeicher. Dies hat den Grund, dass in diesem Modus die Daten zunächst über den PCIe-Bus in den Hostspeicher kopiert werden und anschließend auf die andere Grafikkarte. Dabei können sich die Datenübertragungen überlappen und daher ist die Gesamtgeschwindigkeit etwas höher, als dies bei streng sequentieller Übertragung der Fall wäre. Mit sogennantem „Peer“-Zugriff kann hingegen direkt von einer Grafikkarte an eine andere Grafikkarte über den PCIe-Bus kopiert werden. Dieser Übertragungsmodus ist fast so schnell, wie die Übertragung vom Hostspeicher zur Grafikkarte.

Die Zugriffe der Multiprozessoren der GPU auf ihren globalen Speicher werden alle durch den L2-Cache geleitet. Auf der Tesla K20 besitzt dieser eine Größe von 1,5MiB und eine Bandbreite von 512B/clk. Die „Core Clock“ ist bei diesem Prozessor mit 706MHz getaktet. Zusätzlich existiert pro Multiprozessor ein kleiner, 64KiB großer konfigurierbarer L1-Cache sowie ein 48KiB großer Read-Only Cache. Über den Read-Only Cache können programmatisch ausschließlich Daten verarbeitet werden, die vom Programm als nicht beschreibbar gekennzeichnet wurden (`const __restricted`). Dieser Cache kann außerdem implizit von der Textur-Einheit mitbenutzt werden. Der L1 Cache ist wie zuvor erwähnt teilweise programmierbar und lässt sich in zwei Teile unterteilen. Ein Teil dient als tatsächlicher L1-Cache für Speicherzugriffe. Der andere Teil (auch „shared“-Speicher genannt) kann frei von den Threads innerhalb eines Blocks zum Datenaustausch untereinander verwendet werden. Der L1-Cache besitzt

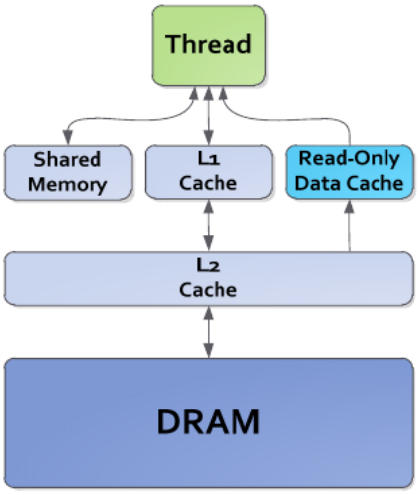


Abbildung 2.3.: Speicherorganisation der GPU

eine mit Registerzugriffen vergleichbare Zugriffszeit.

2.6.4. Programmierung von Grafikbeschleunigern

Für die Programmierung von Grafikbeschleunigern gibt es momentan zwei weit verbreitete Möglichkeiten:

- **CUDA**: CUDA [Kir07] steht für Compute Unified Device Architecture und ist als Wrapper um den gcc-Compiler implementiert. CUDA erweitert den C-Standard um einige Schlüsselwörter, welche die Programmierung der GPU ermöglichen. Über diese kann man definieren, welche C-Funktionen der Compiler für die Grafikkarte in Bytecode übersetzen soll. Dieser Bytecode wird an geeigneter Stelle im Programm hinterlegt und vor der ersten Ausführung der Funktion an die GPU übertragen. Der Aufruf dieser Funktionen im C-Programmcode gestaltet sich über einen kurzen Zusatz, der angibt, wieviele Blöcke und Threads zur Ausführung benötigt werden. Die große Einschränkung von CUDA ist, dass nur NVIDIA-Grafikbeschleuniger damit programmiert werden können.
- **OpenCL**: OpenCL [MGM11] funktioniert nach fast dem gleichen Prinzip wie CUDA. Auch hier wurde der C-Standard um einige Schlüsselwörter erweitert. Hier wird der C-Quellcode allerdings erst zur Laufzeit der OpenCL-Bibliothek übergeben und dort von einem hardwarespezifischen Compiler übersetzt. Daher müssen Funktionen, die in diesem Prgrammcode definiert wurden, über eine OpenCL-Bibliotheksfunktion per Funktionsname aufgerufen werden. Durch diese Architektur ist OpenCL im Gegensatz zu CUDA für alle verfügbaren

Grafikbeschleuniger einsetzbar, bietet dabei aber weniger Möglichkeiten, den Code hardwarenah zu optimieren.

Wenn man durch äußere Vorgaben auf NVIDIA-Hardware festgelegt ist, sollte man CUDA gegenüber OpenCL vorziehen, da der CUDA-Compiler besser optimierten Code erzeugt und so die spezifischere Programmierung und Optimierung für die verwendete Hardware ermöglicht wird. In [Dan+10] finden sich mehrere Benchmarks, welche den Geschwindigkeitsvorteil von CUDA gegenüber einer OpenCL-Implementierung belegen. Unter anderem ist in diesem Test die FFT-Implementierung auf OpenCL um etwa Faktor 20 langsamer als die CUDA-Variante.

3. Die Simulation

3.1. Anforderungen

Die Hauptanforderung an die Simulation ist, dass sie möglichst performant und genau die Wirbeltransportgleichung (2.53) simulieren soll. Vorab ist nicht bekannt, welche Auflösungen bei welcher Parameterwahl notwendig sind, um den Simulationsfehler gering zu halten. Insbesondere bei der Simulation der direkten Enstophiekaskade, bei der immer das vollständige Energiespektrum simuliert wird, ändert sich der Exponent der Kaskade mit Wahl der Auflösung. Deshalb soll die Simulation so ausgelegt sein, dass sie wie andere 2D-Turbulenz-Simulationen [BE12] ebenfalls Auflösungen von bis zu 32768×32768 unterstützt. Zur Simulation der inversen Kaskade wird später eine Auflösung von 4096×4096 gewählt werden, um eine ausreichende räumliche Auflösung zu erzielen. Bei dieser Auflösung würde die fertige Simulation auf einem Prozessor-Kern ca. 400 Tage benötigen, um 500 statistisch unabhängige Datenfelder zu berechnen. Aus diesem Grund ist es erforderlich, die Simulation zu parallelisieren. Die dafür zur Verfügung stehende Hardware wird in Kapitel 3.2 beschrieben.

Ein Hauptaugenmerk bezüglich der Programmlaufzeit liegt auf dem numerischen Verfahren der Integration. Verfahren höherer Ordnung erzeugen einen kleineren Integrationsfehler. Hält man diesen konstant, so lassen sich mit Verfahren höherer Ordnung größere Simulationszeitschritte wählen. Je höher die Ordnung eines Verfahrens ist, je höher ist auch die Anzahl seiner Stufen und damit ergibt sich auch ein immer größerer Berechnungsaufwand. Welchen Einfluss dies hat und welche Integrationsalgorithmen die schnellsten sind, wird in Kapitel 3.3 untersucht. Zur Berechnung der Zeitableitung benutzt man in der 2D-Turbulenz häufig ein Pseudospektralverfahren, wie es in Kapitel 3.4 beschrieben ist. Dieses benötigt pro Ausführungsschritt mehrere Fouriertransformationen zur Berechnung der nichtlinearen Terme. Die Fouriertransformation besitzt ein Laufzeitverhalten von $\mathcal{O}(N^2 \log(N))$ bezüglich der Auflösung, während der restliche Code des Algorithmus nur $\mathcal{O}(N^2)$ benötigt. Sie ist außerdem die einzige Komponente, die einen komplexen Datenaustausch (`All-to-All`) zwischen den Recheneinheiten verursacht. Daher bestimmt neben dem verwendeten Integrationsschema die Performance der verteilten Fouriertransformation hauptsächlich die Simulationsgeschwindigkeit. Da auf Grafikbeschleunigern noch keine frei verfügbaren Implementationen der zweidimensionalen Fouriertransformation für GPUs existieren, beschreibt Kapitel 3.5 zunächst, wie man diese verteilen kann und analysiert im Anschluss deren Performance.

Abgesehen von Performance und Genauigkeit sollte die Simulation möglichst einfach zu bedienen sein. Dies bedeutet, dass die Parameter der Simulation frei über Kommandozeile wählbar sein sollten, damie sie sich einfach per Skript starten lässt. Dies ist wichtig zur Ausführung auf dem CPU-Cluster. Eine Echtzeitausgabe der Simulationsdaten und des Energiespektrums ist ebenfalls notwendig, um den Einfluss verschiedener Simulationsparameter schnell testen zu können. Die Datenspeicherung sollte in einem Format stattfinden, dass eine spätere Auswertung aus Skriptsprachen heraus einfach gestaltet.

3.2. Hardware

Zur verteilten Implementation der Simulation standen folgende Hardwaresysteme zur Verfügung:

1. **Morfeus CPU-Cluster** Der Morfeus Cluster der IVV-Naturwissenschaften bietet die Möglichkeit, Prozesse auf sämtlichen Einzelplatzrechnern der beteiligten Institute zu starten. Bei einer Gesamtzahl von 2885 Knoten besitzt er eine theoretische Gesamtleistung von 4779 GFlops. Die Rechner sind untereinander mit Gigabit-Ethernet angebunden. Zur Speicherung größerer Datenmengen stehen lokale Festplatten und ein über SSH erreichbarer Dateiserver mit momentan 2,5TiB freiem Platz zur Verfügung.
2. **PALMA CPU-Cluster** Der PALMA-Cluster besteht aus 296 Rechenknoten mit insgesamt 3528 Prozessorkernen. Der größte Teil davon sind Knoten mit 8 oder 12 Kernen. Zusätzlich existieren 6 Knoten mit 32 bzw. 40 Kernen. Die einzelnen Knoten besitzen einen Hauptspeicher von entweder 24GiB oder 48GiB RAM. Die Knoten sind untereinander mittels QDR Infiniband verbunden. Es wird für den kompletten Cluster eine theoretische Gesamtleistung von 30 TFlops angegeben. Als Datenspeicher stehen 180TiB Festplattenspeicher zur Verfügung, der über ein „Lustre“-Dateisystem verteilt ist und von dem etwa 40TiB momentan frei sind.
3. **Multiple Grafikbeschleuniger** Ein besonderer Rechner im PALMA-Cluster besitzt 4 eingebaute NVIDIA GeForce GTX 480 Grafikbeschleuniger, die parallel betrieben werden können. Jeder von ihnen besitzt einen globalen Speicher von $1,5$GiB RAM. Anders als die übrigen Rechenknoten besitzt dieser spezielle Rechner 96GiB RAM und ansonsten die gleiche Anbindung (Infiniband, Festplattenspeicher) wie die übrigen Knoten des PALMA-Clusters. Alle 4 Karten besitzen zusammen eine theoretische Leistung von $4 \times 1345\text{GFlops} = 5,38\text{TFlops}$. Der Nachteil dieser Grafikbeschleuniger ist, dass sie eine um 75% verminderte Leistung bei `double`-Rechengenauigkeiten gegenüber den für wissenschaftliche Rechnungen konzipierten Tesla-Karten besitzen, wie man auch in Benchmarks hierzu sieht [NVId].

Eine kurze Überschlagsrechnung zeigt, wie groß die benötigte Datenmenge ist: Für eine Auflösung von 4096×4096 benötigt ein Datensatz mit nur `float` Genauigkeit 64MiB. Für eine Ensemblemittelung bietet es sich durchaus an, über 500 statistisch unabhängige Datensätze zu haben, also 31GiB. Für mehrere Simulationsläufe fallen also einige hundert GiB an. Diese Datenmengen würden theoretisch auf allen Systemen speicherbar sein. Da der PALMA CPU-Cluster von beiden CPU-Clustern sowohl bei der Anbindung an den Datenspeicher als auch bei den verwendeten CPUs wesentlich leistungsstärker ist, ist dieser dem Morfeus CPU-Cluster vorzuziehen.

Als interessante Alternative zum CPU-Cluster bietet sich die Möglichkeit, auf mehreren Grafikbeschleunigern verteilt zu rechnen. Da diese nur einen globalen Speicher von $1,5$GiB besitzen, müssen die Daten ab einer Feldgröße von 8192×8192 (entspricht 6 Felder pro GPU) verteilt werden, damit genügend temporäre Puffer für Fouriertransformation und Integrationsverfahren zur Verfügung stehen. Das kleine Rechenbeispiel mit der theoretischen Leistung zeigt, dass hier mit nur wenigen Karten eine sehr viel höhere Leistung erreicht werden könnte, als im CPU-Cluster. Diese Angaben sind allerdings nur vorsichtig zu betrachten, da beim Datenaustausch der Karten untereinander die Bandbreite des PCIe-Bus bzw. die Speicherbandbreite des Hauptspeichers begrenzend wirken. Inwieweit dies das Laufzeitverhalten der Simulation beeinflusst, soll im Folgenden herausgefunden werden. Daher soll die Simulation sowohl für Ausführung auf dem PALMA-Cluster als auch auf mehreren Grafikbeschleuniger konzipiert und implementiert werden.

3.3. Numerische Integration gewöhnlicher Differentialgleichungen

Nun soll es zunächst darum gehen, welche Integrationsalgorithmen zur Verfügung stehen. Von der Wahl des Verfahrens hängt ab, wieviele temporäre Felder zur Berechnung benötigt werden und damit der Gesamtspeicherbedarf der Simulation. Ebenso erhöht sich der Rechenaufwand mit der Anzahl der benötigten Stufen des Verfahrens linear. Die Anzahl der Stufen ist dabei mindestens so hoch wie die Ordnung des Verfahrens. Verfahren höherer Ordnung besitzen einen kleineren Integrationsfehler und können daher bei gleichem Fehler mehr Zeit pro Simulationschritt simulieren, als ein Verfahren geringerer Ordnung. Man kann von vornherein keine Aussage treffen, wie gut die einzelnen Verfahren auf den unterschiedlichen Hardwarearchitekturen skalieren und welches Verfahren vorzuziehen ist. Deshalb sollen hier nun mehrere Verfahren vorgestellt werden und ihre Performance miteinander verglichen werden.

Da die Wirbeltransportgleichung als partielle Differentialgleichung in der Zeit vorliegt, sollen nun verschiedene Integrationsverfahren erläutert werden, die für die Zeitintegration in Frage kommen. Dazu wird angnommen, dass man diese mit Hilfe eines Pseudospektralverfahrens als gewöhnliche Differentialgleichung in der Zeit betrachten

kann. Zum Abschluss dieses Kapitels soll dann die Performance der verschiedenen Algorithmen auf den unterschiedlichen Hardwareplattformen gegenübergestellt werden.

Gesucht wird also die numerische Approximation einer gewöhnlichen Differentialgleichung der Form

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{x}(t), t) \tag{3.1}$$

Die Anfangsbedingungen seien hierbei vollständig bekannt. In [BW03], [Lam91] oder [Pre+86] sind verschiedene Verfahren zur Lösung beschrieben. Die gebräuchlichsten Verfahren für Vorwärtsintegration sind die Runge-Kutta-Verfahren. Unter ihnen gibt es zahlreiche Varianten und Erweiterungen, die in den folgenden Abschnitten kurz beschrieben werden.

3.3.1. Runge-Kutta-Verfahren

Das Runge-Kutta-Verfahren ist benannt nach Carl Runge [Run95] und Martin Wilhelm Kutta [Kut01]. Um eine Differentialgleichung der Form (3.1) zu lösen wird zunächst eine Zeitdiskretisierung vorgenommen. Man betrachtet dabei nur zu ausgewählten Zeitpunkten t_i die Daten $\boldsymbol{x}_i \approx \boldsymbol{x}(t_i)$, die eine Näherung für den echten Wert zu diesem Zeitpunkt darstellen. Dabei sei die Anfangsbedingung $\boldsymbol{x}_0 = \boldsymbol{x}(t_0)$ vollständig bekannt. Das Runge-Kutta-Verfahren benutzt nun folgendes Schema, um zu einem bekannten $\boldsymbol{x}_i$ den nachfolgenden Zeitschritt $\boldsymbol{x}_{i+1}$ zu berechnen:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + h \sum_{j=1}^{s} b_j \boldsymbol{k}_{ij} \tag{3.2a}$$

$$\boldsymbol{k}_{ij} = \boldsymbol{f}(\boldsymbol{x}_i + h \sum_{l=1}^{s} a_{jl} \boldsymbol{k}_{il}, t_n + h c_j) \tag{3.2b}$$

h ist hierbei die Schrittweite des Zeitschritts $h = t_{i+1} - t_i$. Bei jedem Integrationsschritt wird ein Fehler gemacht, der sogenannte lokale Diskretisierungsfehler $\tau_i = \max\{\boldsymbol{x}_i - \boldsymbol{x}(t_i)\}$. Über seine Taylorentwicklung lässt sich festlegen, bis zu welcher Ordnung $\tau \leq \mathcal{O}(h^p)$ ein jeweiliges Verfahren die Differentialgleichung näherungsweise löst. Setzt man voraus, dass $\boldsymbol{f}$ und die Verfahrensvorschrift stetig sind, so bezeichnet p hierbei die Konvergenzordnung. Herleitungen verschiedener Verfahrensvorschriften unterschiedlicher Konvergenzordnungen finden sich in [BW03].

Es ist üblich, die Verfahrensvorschriften in sogenannten Butcher-Tableaus (Tabelle 3.1) anzugeben.

c_1	a_{11}	a_{12}	$\dots$	a_{1s}
c_2	a_{21}	a_{22}	$\dots$	a_{2s}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	a_{s1}	a_{s2}	$\dots$	a_{ss}
	b_1	b_2	$\dots$	b_s

Tabelle 3.1.: Allgemeines Butcher-Tableau zur Beschreibung einer Runge-Kutta-Verfahrensvorschrift

Besonders einfach zu implementieren sind explizite Verfahren. Dort gilt $a_{jl} = 0$ für alle $l \geq j$, und damit kann man die $\boldsymbol{k}_{ij}$ eines Zeitschrittes für alle j nacheinander berechnen. Häufig findet hiervon das "klassische"Runge-Kutta-Verfahren vierter Ordnung (RK4) Anwendung, wie es in Tabelle 3.2 zu sehen ist.

0	
	1

(a) Euler

0		
1	1	
	$\frac{1}{2}$	$\frac{1}{2}$

(b) Heun

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{3}$

(c) RK4

Tabelle 3.2.: Butcher-Tableau einiger klassischer Runge-Kutta-Verfahren

Um einen möglichst kleinen Diskretisierungsfehler zu erhalten muss man entweder ein Verfahren hoher Ordnung benutzen oder den Zeitschritt möglichst gering wählen. Dabei ist zu berücksichtigen, dass die Stufe s des Verfahrens schneller steigt, als die Ordnung p, sodass dann schon ab der Ordnung $p = 5$ kein Verfahren mit $s = 5$ existiert.

3.3.2. Adaptiver Zeitschritt

Um eine größtmögliche Simulationsgeschwindigkeit zu erreichten, muss man einen möglichst großen Zeitschritt wählen, wodurch sich ebenfalls der Diskretisierungsfehler erhöht. Ein optimaler Zeitschritt lässt sich nicht immer im Voraus festlegen und kann sich im Laufe der Simulation ändern. Für die Runge-Kutta-Verfahren benötigt man keine äquidistanten Zeitschritte. Daher ist es sinnvoll, zur Laufzeit ein Kriterium zu haben, welches einem eine aktuelle obere Grenze des Diskretisierungsfehlers für einen bestimmten Zeitschritt h angibt. Eine solche Abschätzung geben sogenannte eingebettete Verfahren, wie z.B. das Runge-Kutta-Fehlberg-Verfahren (RKF45, Tabelle

3.3) [Feh70], das Cash-Karp-Verfahren (RKCK45, Tabelle 3.4) [CK90] oder das Dormand-Prince-Verfahren (DOPRI54, Tabelle 3.5) [DP80]. Weitere eingebettete Verfahren bis zur Ordnung 8 sind in [PD81] zu finden.

0						
$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$				
$\frac{12}{13}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$			
1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$		
$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	
	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{5}$	0
	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$

Tabelle 3.3.: Butcher-Tableaus des RKF45-Verfahrens. Die obere Ergebniszeile enthält das Ergebnis vierter Ordnung und die zweite Zeile fünfter Ordnung.

0						
$\frac{1}{5}$	$\frac{1}{5}$					
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$				
$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{6}{5}$			
1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{70}{27}$	$\frac{35}{27}$		
$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$	
	$\frac{37}{378}$	0	$\frac{250}{621}$	$\frac{125}{594}$	0	$\frac{512}{1771}$
	$\frac{2825}{27648}$	0	$\frac{18575}{48384}$	$\frac{13525}{55296}$	$\frac{277}{14336}$	$\frac{1}{4}$

Tabelle 3.4.: Butcher-Tableaus des RKCK45-Verfahrens. Die obere Ergebniszeile enthält das Ergebnis vierter Ordnung und die zweite Zeile fünfter Ordnung.

RKF45 und RKCK45 sind beides sechs-stufige Verfahren. Sie minimieren den Fehler vierter Ordnung. Die fünfte Ordnung wird nur zur Fehlerbestimmung mitberechnet. Den Fehler erhält man über Differenzbildung der beiden Ergebnisse. DOPRI54 ist im Gegensatz dazu ein sieben-stufiges Verfahren und minimiert den Fehler der fünften Ordnung. Hier wird der Fehler als Differenz zum Ergebnis der vierten Ordnung bestimmt.

Sei nun der Diskretisierungsfehler ϵ des Verfahrens nach einem Integrationsschritt bekannt, so schätzt man beim adaptiven Verfahren über diesen Fehler einen nächsten Zeitschritt ab $h_{\text{next}} = \beta \left(\frac{\epsilon_0}{\epsilon}\right)^{\frac{1}{p}}$ mit $p = 5$ (siehe [Gea71] oder [HNW93, S. 167]). β ist hierbei ein Sicherheitsparameter mit $\beta \leq 1$, damit der gewählte Zeitschritt immer etwas kleiner ist, als der optimale. Man wählt in der Regel β zwischen 0,8 und 0,9. Ist ϵ

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Tabelle 3.5.: Butcher-Tableaus des DOPRI54-Verfahrens. Die obere Ergebniszeile enthält das Ergebnis fünfter Ordnung und die zweite Zeile vierter Ordnung.

größer als eine vorgegebene Schranke ϵ_0, so muss der Integrationsschritt mit einem kleineren Zeitschritt wiederholt werden. In diesem Fall ist $p = 4$. Der Algorithmus dazu ist in Abbildung 3.1 schematisch dargestellt.

3.3.3. Integrierender Faktor (IFRK)

Die Verfahren mit integrierendem Faktor sind zurückzuführen auf [Law67]. Sie eignen sich für Differentialgleichungen der Form

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \mathcal{L}\boldsymbol{x} + \mathcal{N}(\boldsymbol{x}, t) \tag{3.3}$$

Dabei bezeichnet $\mathcal{L}$ einen linearen Operator und $\mathcal{N}$ eine nichtlineare Funktion. Der besondere Vorteil in diesen Verfahren besteht darin, dass sie den linearen Anteil der Differentialgleichung exakt lösen. Besonders hilfreich ist dies, wenn der lineare Operator Eigenwerte stark unterschiedlicher Größenordnung besitzt. Ohne integrierenden Faktor müsste sich der Simulationsschritt sonst stets nach dem größten Eigenwert richten. Verfahren mit integrierendem Faktor sorgen in Abhängigkeit der Parameterwahl für die Simulation für einen erheblichen Zuwachs in der simulierten Zeit pro Integrationsschritt.

Transformiert man Gleichung (3.3) mit $\boldsymbol{y} = \mathrm{e}^{-\mathcal{L}t}\boldsymbol{x}$, so erhält man

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{y} = \mathrm{e}^{-\mathcal{L}t}\mathcal{N}\left(\mathrm{e}^{\mathcal{L}t}\boldsymbol{v}, t\right) \tag{3.4}$$

Entsprechend [Law67] kann man auf diese Gleichung ein Runge-Kutta-Verfahren anwenden, wie in Abschnitt 3.3.1 beschrieben. Die resultierenden Gleichungen für $\boldsymbol{x}_i$

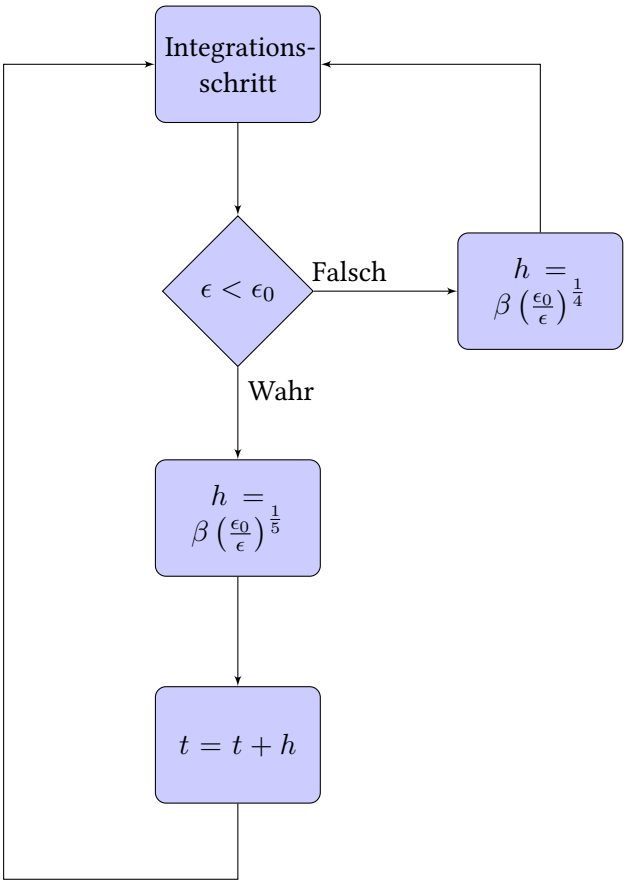


Abbildung 3.1.: Schema zur Integration mit adaptivem Zeitschritt

besitzen dann folgende Form:

$$\boldsymbol{p}_{ij} = \boldsymbol{x}_i + h \sum_{l=1}^{s} a_{jl} \boldsymbol{k}_{il} \tag{3.5a}$$

$$\boldsymbol{k}_{ij} = \mathrm{e}^{-c_j h \mathcal{L}} \mathcal{N} \left(\mathrm{e}^{c_j h \mathcal{L}} \boldsymbol{p}_{ij}, t_i + c_j h \right) \tag{3.5b}$$

$$\boldsymbol{x}_{i+1} = \mathrm{e}^{h \mathcal{L}} \left(\boldsymbol{x}_i + h \sum_{l=1}^{s} b_l \boldsymbol{k}_{il} \right) \tag{3.5c}$$

Ebenso wie die normalen Runge-Kutta-Verfahren kann auch ein eingebettetes Verfahren mit adaptivem Zeitschritt verwendet werden.

3.3.4. Exponentielle Zeitableitung (ETDRK)

Das Verfahren der exponentiellen Zeitableitung funktioniert ähnlich dem Verfahren mit integrierendem Faktor. Die etwas komplexere Herleitung dieser Verfahren ist in [CM02], [Kro05] und [HO10] beschrieben. Desweiteren gibt [MW05] eine sehr gute Übersicht über verschiedene Verfahren, deren Herleitung und Besonderheiten bei der Implementation. Eine Beispielimplementation findet sich z.B. in [BSW07].

Das Integrationsschema muss gegenüber dem Runge-Kutta-Schema etwas erweitert werden und nimmt dann folgende Form an:

$$\boldsymbol{k}_{ij} = \sum_{l=1}^{s} a_{jl}(h\mathcal{L})h\mathcal{N}\left(\boldsymbol{k}_{il}, t_{i-1} + c_l h\right) + u_j(h\mathcal{L})\boldsymbol{x}_i \tag{3.6a}$$

$$\boldsymbol{x}_{i+1} = \sum_{l=1}^{s} b_l(h\mathcal{L})h\mathcal{N}\left(\boldsymbol{k}_{il}, t_{i-1} + c_l h\right) + v(h\mathcal{L})\boldsymbol{x}_i \tag{3.6b}$$

Nun sind a, b keine Konstanten mehr, sondern Funktionen von $h\mathcal{L}$. Die u_j und v befinden sich in der zusätzlichen Spalte dargestellt. Tabelle 3.6 zeigt exemplarisch ein vierstufiges Verfahren nach [Kro05].

0					1
$\frac{1}{2}$	$\frac{1}{2}\varphi_{1,2}$				$\varphi_{0,2}$
$\frac{1}{2}$	$\frac{1}{2}\varphi_{1,2} - \varphi_{2,2}$	$\varphi_{2,2}$			$\varphi_{0,2}$
1	$\varphi_1 - 2\varphi_2$	0	$2\varphi_2$		φ_0
	$\varphi_1 - 3\varphi_2 + 4\varphi_3$	$2\varphi_2 - 4\varphi_3$	$2\varphi_2 - 4\varphi_3$	$-\varphi_2 + 4\varphi_3$	φ_0

Tabelle 3.6.: Butcher-Tableau für das exponentielle Verfahren ETDRK4-B nach [Kro05]

Dabei berechnet sich φ_l über die Rekursionsformel

$$\varphi_0(z) = \mathrm{e}^z \tag{3.7a}$$

$$\varphi_1(z) = \frac{\mathrm{e}^z - 1}{z} \tag{3.7b}$$

$$\varphi_l(z) = \frac{\varphi_{l-1}(z) - \frac{1}{(l-1)!}}{z^l} \tag{3.7c}$$

und $\varphi_{lj}(z)$ ist eine Abkürzung für $\varphi_{lj}(z) = \varphi_l(c_j z)$. Diese Funktionen sind für kleine z numerisch nicht trivial zu berechnen. In der Literatur sind verschiedene

Methoden dazu bekannt [MW05], [KT05]. Eine davon bildet das Rand-Integral mittels der Cauchyschen Integrationsformel:

$$\varphi_l(z) = \frac{1}{2\pi i} \int_\Gamma \varphi_l(\lambda)(\lambda - z)^{-1} \mathrm{d}\lambda \tag{3.8}$$

Der Rand kann beliebig gewählt werden, z.B. als Kreis mit äquidistanten Punkten um den Punkt z. Ein eingebettetes Verfahren, wie es für eine adaptive Schrittweite notwendig wäre, findet sich in der Literatur leider nicht.

3.3.5. Vergleich der Integrationsverfahren

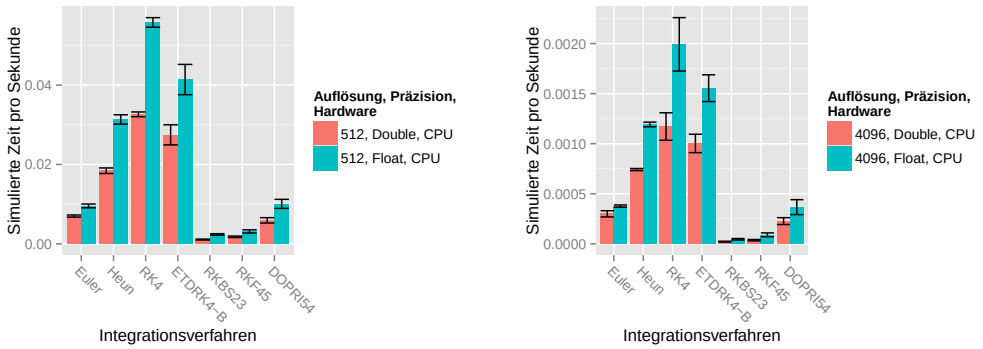
Zum Vergleich der Integrationsverfahren wird ein konkreter Simulationszustand einer Simulation genommen und über eine kurze Zeit weitersimuliert. Dabei wird die simulierte Zeit gemessen. Die Performance eines Integrationsverfahrens lässt sich dann durch die simulierte Zeit pro Sekunde bestimmen. Auf dem CPU-Cluster wurde die Simulation jeweils mit der Anzahl von Prozessoren ausgeführt, welche dort die maximale Performance liefert. Auf Grund der in Kapitel 3.5.2 und 3.7.2 beschriebenen Hardwarelimitierungen werden absoluten Zahlen vermutlich nicht direkt vergleichbar sein mit denen von anderen CPU-Clustern.

Eine allgemeines Problem der Performance-Messung ist es, dass die nicht-eingebetteten Verfahren keine Möglichkeit bieten den Integrationsfehler abzuschätzen. Bei diesen muss man die Schrittweite solange erhöhen, bis der Algorithmus offensichtlich nicht mehr korrekt integriert, um einen optimalen Zeitschritt zu finden. Der dadurch entstehende Diskretisierungsfehler wird zwar von der Viskosität gedämpft, könnte aber zu nicht einschätzbaren Simulationsfehlern führen, die sich erst später bei der Auswertung bemerkbar machen und als physikalischer Effekt fehlinterpretiert werden.

Die eingebetteten Verfahren bieten hingegen die Möglichkeit, den Fehler auf einen fest vorgegebenen Wert zu beschränken (in diesem Fall auf 1% relative Genauigkeit der einzelnen Moden im Fourierraum). Anhand dieses Fehlers bestimmen sie ihren Zeitschritt, der dadurch wesentlich geringer ist, als bei nicht-eingebetteten Verfahren. Daher lässt sich Performance beider Verfahrenstypen in dieser Messung nur schlecht untereinander vergleichen.

CPU-Cluster

Abbildung 3.2 zeigt die simulierte Zeit pro Sekunde verschiedener Integrationsverfahren. Hier sieht man deutlich, dass Verfahren höherer Ordnung stets zu bevorzugen sind. Bei den nicht-adaptiven Verfahren schneidet von den getesteten das Runge-Kutta-Verfahren 4. Ordnung und bei den adaptiven Verfahren das Dormand-Prince-Verfahren



(a) Auflösung 512×512

(b) Auflösung 4096×4096

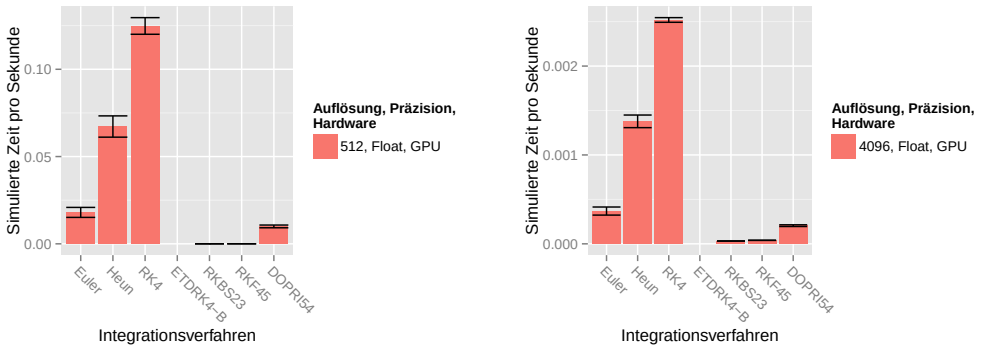
Abbildung 3.2.: Simulierte Zeit pro Sekunde für verschiedene Integrationsverfahren auf einem CPU-Cluster. Die vorgegebenen Zeitschritte h für die nicht-adaptiven Verfahren waren bei `float`-Genauigkeit: Euler: 3×10^{-5}s / Heun: 1.8×10^{-4} / RK4: $6,2 \times 10^{-4}$ / ETDRK4: $5,0 \times 10^{-4}$ und bei `double`-Genauigkeit: 4×10^{-5}s / Heun: 1.9×10^{-4} / RK4: $6,2 \times 10^{-4}$ / ETDRK4: $5,3 \times 10^{-4}$. In allen getesteten Verfahren kam der integrierende Faktor aus Kapitel 3.3.3 zum Einsatz. (Simulation: `integration`, siehe Anhang A)

5. Ordnung am besten ab. Dieses Ergebnis ist deshalb interessant, weil ab Verfahren 4. Ordnung die Zahl der Stufen schneller steigt als die der Ordnung. Es gibt daher kein Verfahren 5. Ordnung mit nur 5 Stufen. Trotzdem überwiegt hier scheinbar die Wirkung des größer wählbaren Zeitschritts. Die Verwendung von `float`-Genauigkeit bringt gegenüber der `double`-Genauigkeit in dieser Messung einen Geschwindigkeitsvorteil von über 30%.

Grafikbeschleuniger

Abbildung 3.3 zeigt die gleiche Messung wie auf dem CPU-Cluster auf einem Grafikbeschleuniger durchgeführt. Das ETDRK4-Verfahren wurde wegen seiner Komplexität hier nicht implementiert. Bei den Messungen zeigt sich genau das gleiche Muster wie beim CPU-Cluster. Auch hier erzielen Verfahren höherer Ordnung eine wesentlich höhere Simulationsgeschwindigkeit.

Die konkrete Auswahl der verwendeten Hardware und des Integrationsverfahrens für die Simulation hängt von der Wahl der Simulationsparameter ab und wird detailliert in Kapitel 4.1 erläutert.



(a) Auflösung 512×512 **(b)** Auflösung 4096×4096

Abbildung 3.3.: Simulierte Zeit pro Sekunde für verschiedene Integrationsverfahren auf einer GPU. Die vorgegebenen Zeitschritte h für die nicht-adaptiven Verfahren waren bei `float`-Genauigkeit: Euler: 3×10^{-5}s / Heun: 1.8×10^{-4} / RK4: $6,2 \times 10^{-4}$. In allen getesteten Verfahren kam der integrierende Faktor aus Kapitel 3.3.3 zum Einsatz. (Simulation: `integration`, siehe Anhang A)

3.4. Pseudospektralverfahren

Um zu verstehen, warum man ein Pseudospektralverfahren anwendet, muss man zunächst verstehen, warum man hier nicht mit finiten Differenzen arbeitet. Bei der Methode der finiten Differenzen entwickelt man den Differentialoperator $\frac{\partial}{\partial x}$ um den Abstand Δx. Um möglichst kleine räumliche Frequenzen auflösen zu können, muss man dabei möglichst viele Ordnungen berücksichtigen. Eine einfachere Methode ist es, Ableitungen im Fourierraum zu berechnen, da dort alle auflösbaren Frequenzen vorhanden sind. Die Fouriertransformation stelle eine lineare Operation dar und deshalb können alle Summanden einer Gleichung einzeln in den Fourierraum transformiert werden. Das Gleichungssystem

$$\frac{\partial}{\partial t}\boldsymbol{f}(\boldsymbol{x},t) = \mathcal{L}\left(\boldsymbol{\nabla}\right)\boldsymbol{f}(\boldsymbol{x},t) \tag{3.9}$$

wird somit zu

$$\frac{\partial}{\partial t}\boldsymbol{f}(\boldsymbol{k},t) = \mathcal{L}\left(i\boldsymbol{k}\right)\boldsymbol{f}(\boldsymbol{k},t) = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{f}(\boldsymbol{k},t) \tag{3.10}$$

Da im Fourierraum nun keine weitere räumliche Ableitung mehr vorkommt, kann man die partielle Ableitung durch eine gewöhnliche Ableitung ersetzen und mit den Methoden aus Kapitel 3.3 integrieren. Dieses Verfahren nennt man Spektralverfahren. Taucht in der Differentialgleichung noch ein nichtlinearer Term auf, so benötigt man zur Lösung ein sogenanntes Pseudospektralverfahren. Dabei wird für einen

Integrationsschritt so viel wie möglich im Fourierraum und der nichtlineare Term im Realraum berechnet. Für ein Gleichungssystem der Form

$$\frac{\partial}{\partial t} = \mathcal{L}\left(\boldsymbol{\nabla}\right) \boldsymbol{f}(\boldsymbol{x}, t) + \mathcal{N}(\boldsymbol{f}(\boldsymbol{x}, t), \nabla \boldsymbol{f}(\boldsymbol{x}, t), ..., \nabla^n \boldsymbol{f}(\boldsymbol{x}, t)) \tag{3.11}$$

wird dabei nach folgendem Schema ein Integrationsschritt durchgeführt

1. Berechne alle benötigten Ableitungen des aktuellen $\boldsymbol{k}$ im Fourierraum
2. Transformiere die Ableitungen in den Realraum
3. Berechne den nichtlinearen Anteil
4. Transformiere das Ergebnis zurück in den Fourierraum
5. Addiere den linearen und nichtlinearen Term

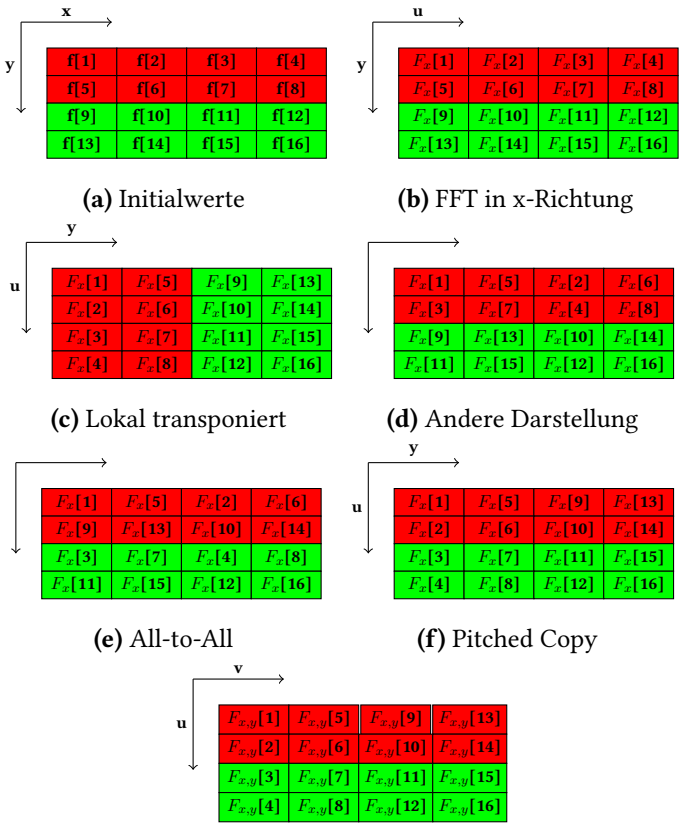
3.5. Verteilte zweidimensionale Fouriertransformation (2D-FFT)

Für die verteilte Ausführung der Fouriertransformation müssen die Daten auf die beteiligten Recheneinheiten verteilt werden. Zur die Implementierung CPU-Cluster stellt die FFTW-Bibliothek [FJ05] bereits eine verteilte FFT unter Benutzung der MPI-Schnittstelle [GLS99] zur Verfügung. Die von der FFTW vorgegebene Verteilung erfolgt dabei nach einem Schema, wobei jede Recheneinheit einen zusammenhängenden Block aus Zeilen im Realraum bzw. Spalten im Fourierraum verwaltet.

Möchte man stattdessen die Fouriertransformation auf auf NVIDIA-Hardware durchführen, so verwendet man die von NVIDIA bereitgestellte CUDA FFT-Bibliothek [NVIb] (cuFFT), da diese zur Zeit die schnellste Implementation ist, auch wenn andere Varianten in OpenCL z.B. mittlerweile auf bis zu 90% der Leistung herankommen [Li+11]. Die cuFFT-Bibliothek unterstützt im Gegensatz zur FFTW keine parallele Ausführung der Fouriertransformation auf mehreren Grafikbeschleunigern. Dieses Problem kann mit Hilfe eines Algorithmus gelöst werden, wie er in [CCM10], [NMM12] und [NSM12] für die dreidimensionalen FFT beschrieben wird. Da hiervon bisher keine frei verfügbare Bibliothek existiert, soll dieser Algorithmus nun zunächst auf die zweidimensionale FFT umgeschrieben und anschließend implementiert werden.

3.5.1. Datenlayout und Algorithmus

Die Verteilung der Daten soll nun so erfolgen, wie dies in Abbilung 3.4 für ein kleines Beispielfeld zu sehen ist.



(a) Initialwerte **(b)** FFT in x-Richtung

(c) Lokal transponiert **(d)** Andere Darstellung

(e) All-to-All **(f)** Pitched Copy

(g) FFT in y-Richtung

Abbildung 3.4.: Beispiel einer verteilten Fouriertransformation auf einem Datenfeld von 4x4 Elementen. Die Daten liegen dabei auf zwei Recheneinheiten, hier durch rot bzw. grün dargestellt. Die Daten liegen pro Recheneinheit zusammenhängend im Speicher, zuerst entlang der Reihen (row-major). Die eingezeichneten Koordinatenachsen geben (sofern möglich) die x,y-Richtungen im Ortsraum bzw. die entsprechenden u,v-Richtungen im Fourierraum an.

Der Algorithmus funktioniert so, dass zunächst die erste FFT entlang der Zeilen lokal auf jeder Recheneinheit durchgeführt wird. Diese Aufgabe kann von einer allgemeinen Bibliothek wie z.B. der cuFFT erledigt werden. Die Ergebnisse dieser ersten Transformation werden mit F_x bezeichnet.

Anschließend werden die Daten lokal transponiert. Dabei vertauschen sich die Dimensionen der lokalen Daten. In Abbildung 3.4d sieht man, wie die Daten nach dieser Transformation lokal im Speicher liegen.

Danach müssen nun die Zeilen ausgetauscht werden. Von der MPI bekannt ist der All-to-All Algorithmus. Dieser transferiert aus einem N elementigen Array jedes n-te Element an die n-te Recheneinheit. In diesem Beispiel hält jede Recheneinheit 2 Zeilen, die entsprechend an die erste bzw. zweite Recheneinheit übertragen werden müssen.

Nach dieser Übertragung liegen die Daten auf ihrer richtigen Recheneinheit, aber noch falsch sortiert. Hier helfen sogenannte "Pitched Copies"weiter. In diesem Beispiel wird angenommen, dass die Daten eine Breite von 2 zusammenhängenden Elementen haben und die Zeilenanfänge sich im Abstand von 4 Elementen befinden. Quelle sind Element 1 und 3. Ziel der Kopie sind die einzelnen Zeilen als zusammenhängende Bereiche von 4 Elementen.

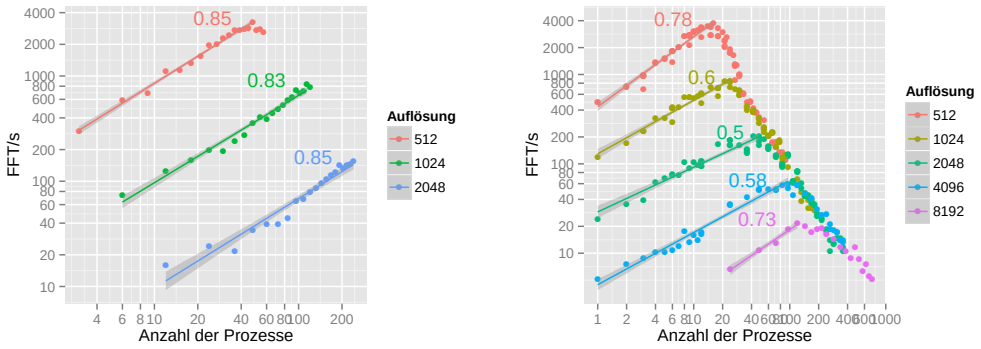
In Abbildung 3.4f sind die Daten nach diesem Kopiervorgang zu sehen. Sie liegen schon richtig im Speicher, um nun entlang ihrer Zeilen, also in y-Richtung per FFT transformiert zu werden. Im resultierenden Feld sind die beiden Raumrichtungen nun transponiert im Speicher gegenüber einer normalen zweidimensionalen FFT.

Dieser hier für ein 4x4 Datenfeld beispielhaft durchgeführte Algorithmus lässt sich problemlos auf große zweidimensionale Datenfelder erweitern:

1. FFT entlang der Zeilen (x-Richtung)
2. Lokal transponieren
3. All-to-All Zeilen übertragen
4. "Pitched Copy"
5. FFT entlang der Zeilen (y-Richtung)

3.5.2. Performance

Dieser Algorithmus für die verteilte FFT wurde nun mit Hilfe von CUDA (Kapitel 2.6) und der CUDA FFT-Bibliothek [NVIb] (cuFFT) für NVIDIA-GPUs implementiert und auf einem Multi-GPU-System getestet. Zum Vergleich diente die verteilte FFT auf dem PALMA-Cluster über FFTW [FJ05] mit MPI [GLS99].



(a) Alte Messung **(b)** Neue Messung

Abbildung 3.5.: Geschwindigkeit in Fourier-Transformationen pro Sekunde der Verteilten FFT auf dem CPU-Cluster.

CPU-Cluster

Zunächst soll einmal das Resultat auf dem CPU-Cluster genauer betrachtet werden. Die Daten für diese Performance-Analyse wurden zu zwei verschiedenen Zeitpunkten erhoben (Abbildung 3.5). In beiden Messungen skaliert die Fouriertransformation scheinbar nach einem Potenzgesetz FFTs$/s = F_1(N)P^\alpha$, was man in doppelt logarithmischer Darstellung gut erkennt. Der Exponent α ist hierbei bei den Graphen eingetragen. Im Mittel über die getesteten Auflösungen der zweiten Messung ist $\alpha = 0,64 \pm 0,081$. $F_1(N)$ entspricht jeweils der von der Auflösung abhängigen Leistung eines einzelnen Prozesses. Der durch P Prozesse erreicht Speedup ist in dieser Darstellung P^α. Ein Exponent von $\alpha = 1$ würde einem optimalen Speedup entsprechen.

Im Gegensatz zu der älteren Messung ist die Geschwindigkeit der verteilten FFT der neueren Messung doppelt so schnell. Allerdings bricht die Performance schon bei der Hälfte der gesteten Prozesszahlen exponentiell ein. Es wird entgegen der Erwartung keine Sättigung erreicht. Dies ist nicht durch ein Softwareproblem zu erklären, da sich die Software und der Quellcode des Benchmarks innerhalb der zwei Wochen zwischen den Messungen nicht geändert haben. Ein Austausch der MPI-Bibliothek (OpenMPI 1.6.2, MVAPICH2 1.8.1 oder eine auf dem Cluster vorinstallierte MPI-Version) brachte keine Verbesserung dieses Verhaltens. Um ein Problem einzelner Rechenknoten auszuschließen, wurden unterschiedlichste Knotenkonfigurationen (z.B. jeder Prozess auf einem unterschiedlichen Knoten oder möglichst wenige Knoten) getestet, aber das Problem blieb gleichermaßen bestehen. Es wird daher vermutet, dass es sich um eine Hardwarelimitierung des Infiniband-Netzwerkes handelt. Ein Umstieg auf das unsichere Datagram-Protokoll „ofud“ in OpenMPI zeigte bei ersten Tests keinen Leistungseinbruch an dieser Stelle, aber auf Grund von Paketverlusten war ein zuverlässiges Arbeiten der FFT unmöglich.

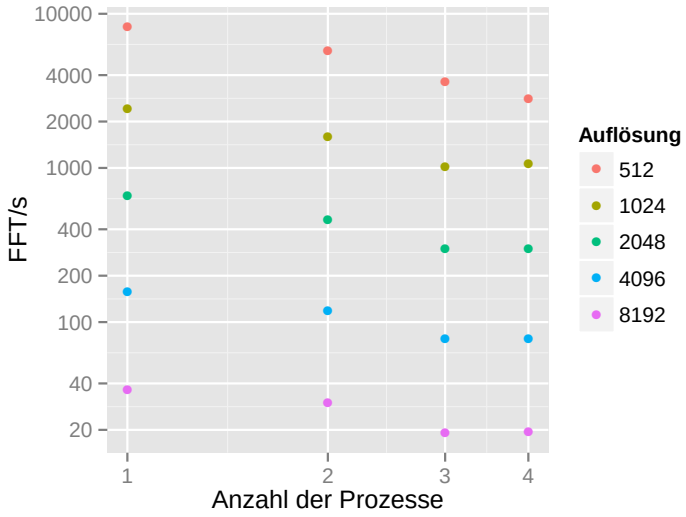


Abbildung 3.6.: Geschwindigkeit in Fourier-Transformationen pro Sekunde der Verteilten 2D-FFT auf einem Multi-GPU System (mit MVAPICH2 1.9a2).

Multi-GPU System

Abbildung 3.6 zeigt die Ergebnisse der Performance-Messung der verteilten FFT auf dem Multi-GPU System.

Diese zeigt leider ein negatives Skalierungsverhalten unter Hinzunahme weiterer GPUs. Dies wäre vielleicht auf den ersten Blick nicht zu erwarten gewesen, denn die 3D-FFT zeigt in [CCM10] und vor allem in [NMM12] bzw. [NSM12] ein sehr gutes Skalierungsverhalten. Für diesen Unterschied gibt es zwei Gründe. Der Hauptgrund ist der, dass sowohl in 2D als auch in 3D nur eine einzige Datenkommunikation nach dem All-To-All Schema stattfinden muss. Daher und weil die 2D-FFT stark von der Geschwindigkeit der Datenübertragung limitiert wird, sinkt der Anteil der Datenkommunikation an der gesamten Berechnungsdauer der FFT und damit skaliert die 3D-FFT besser mit der Anzahl der GPUs. Außerdem besitzt die in [NSM12] verwendete FFT zur Datenübertragung Low-Level Infiniband-Routinen und optimierte CUDA-Kopieralgorithmen, da sonst auch ein schlechteres Skalierungsverhalten beobachtet wurde. Betrachtet man die reine Performance der Fouriertransformation, so sollte man möglichst wenig Grafikbeschleuniger einsetzen, um eine optimale Performance zu erzielen. Auf Grund des begrenzten globalen Speichers ist dies allerdings nicht immer möglich, da zum Beispiel selbst eine Tesla K20 nur 5GiB Hauptspeicher besitzt, der bei `float`-Genauigkeit nur für 20 Felder in einer Auflösung von 8192×8192 ausreicht. Ebenfalls relativiert sich dieses Skalierungsverhalten, wenn die weiteren Berechnungen innerhalb der Simulation lokal durchgeführt werden können, wie man

später in Kapitel 3.7.2 sieht.

Trotz des schlechten Skalierungsverhaltens liegt man auch mit 4 GPUs bei allen gemessenen Auflösungen mit $N > 512$ noch ungefähr beim Maximum der Leistung des PALMA CPU-Clusters. Bei Einsatz einer Grafikkarte ist man sogar etwa doppelt so schnell. Dazu sei angemerkt, dass die eingesetzten Grafikkarten (GeForce GTX 480) zur Zeit nicht die aktuellsten Modelle sind und nicht für den wissenschaftlichen Einsatz konzipiert wurden. Dies zeigt deutlich den Vorteil, den der Einsatz von Grafikkarten gegenüber CPU-Clustern bringt, vor allem, wenn man zusätzlich die Anschaffungs und Betriebskosten beider Systeme gegenüberstellt.

Die noch recht gute Performance beim Einsatz von zwei GPUs ist dadurch zu erklären, dass der sogenannte „Peer"-Zugriff zwischen diesen beiden Grafikkarten aktiviert war, der auch von der MVAPICH2-Bibliothek unterstützt wird. Dies bringt einen enormen Geschwindigkeitsvorteil bei der direkten Datenübertragung über den PCIe-Bus, wie bereits in Kapitel 2.6.3 gezeigt wurde.

3.6. Implementation

3.6.1. Software und Bibliotheken

Die Software- und Bibliotheksauswahl gestaltet sich ziemlich interessant, da hier eine große Auswahl an verschiedenen Bibliotheken zur Verfügung steht, die alle unterschiedliche Vor- und Nachteile haben.

Echtzeitausgabe

Um eine hochperformante Datenausgabe des Wirbelstärkefelds zu erzielen bietet es sich an, die Daten direkt an die Grafikkarte zu übertragen und per OpenGL auszugeben. Der relativ neue OpenGL 3.0 Standard erlaubt es hier, die Daten per sogenanntem Shader-Programm aufzuarbeiten, sodass nur ein kurzer Kopiervorgang über den schnellen PCIe-Bus notwendig ist, bevor die Grafikkarte die Ausgabe nahezu selbständig übernehmen kann, bevor die Anwendung ihre Simulation fortführt. Die eigentliche Darstellung läuft asynchron ab. Die Ausgabe des Energiespektrums gestaltet sich leider etwas aufwändiger, da hierbei einzelne Punkte gezeichnet werden müssen und auf Achsenbeschriftungen nicht verzichtet werden kann. Deswegen wird hierfür die Bibliothek „dislin" verwendet [Mic97]. Diese besitzt ein einfach zu programmierendes Interface, um Punkt- oder Liniendiagramme zu erzeugen. Beschränkt man sich auf eine Datenausgabe, die nicht jeden Berechnungsschritt stattfindet, so bemerkt man kaum eine Verlangsamung der Simulation.

Die Datenausgabe der GPU-Simulation ist besonders schnell implementiert, da hier der zur Berechnung benutzte Datenpuffer direkt zur Ausgabe verwendet werden kann.

Über ein Kompatibilitäts-Interface zu OpenGL lässt sich der Datenpuffer für CUDA sperren und per Grafik-Shader darstellen.

Speicherformat

Die Datenspeicherung sollte in einem Format erfolgen, das es ermöglicht, die Daten möglichst strukturiert zu speichern. Dies bedeutet, dass auch mehrere Datenfelder in einer Datei liegen können sollten und diese Datenfelder mit Attributen versehen werden können. Außerdem sollte es ein Format sein, dass von verschiedenen scriptbaren Statistikprogrammen verarbeitet werden kann, um die spätere Auswertung zu vereinfachen. Hierfür bietet sich das Format „HDF5“ an [FCY99] welches z.B. auch in „R“ [Cha08] unterstützt wird. Es bietet außerdem den Vorteil, Dateien im MPI-Verbund zu bearbeiten. Das heißt konkret, dass Lese- und Schreibvorgänge eines Feldes so synchronisiert werden können, dass jeder Prozess seinen ihm zugeordneten Datenblock einer Datei lesen bzw. schreiben kann.

Auf der GPU ist die Datenspeicherung asynchron implementiert. Das bedeutet, dass die Berechnung der Zeitintegration auf der GPU fortfahren kann, während die Felddaten zum Hauptspeicher des Rechners transferiert werden, um sie später auf die Festplatte serialisieren zu können.

Programmiersprache

Die Wahl der Implementationssprache fällt auf C++, da diese Sprache zum einen sehr effizienten Programmcode erzeugt und zum anderen sehr gute objektorientierte Strukturierungsmöglichkeiten bietet. Ferner ist es eine der wenigen Sprachen, die mit CUDA für die Grafikkartenprogrammierung zusammenarbeitet. Als Sprachvariante ist im Hinblick auf Code-Ästhetik C++11 die beste Wahl, da neue Spracherweiterungen wie `auto`, `rvalue`-Referenzen oder variadische Templates die Entwicklung eines kurzen und performanten Programmcodes vereinfachen. Die beste Unterstützung bietet hier der GNU G++ Compiler. Als direkte Konkurrenz dazu unterstützt der Intel C++ Compiler bisher leider nur einen geringen Teil von C++11 und hatte bei ersten Tests Probleme, mit der Bibliothek „Eigen“ zusammenzuarbeiten. Der Geschwindigkeitsvorteil des Intel C++ Compilers gegenüber dem GNU C++ Compiler hängt stark davon ab, wie der Programmcode aufgebaut ist, der übersetzt werden soll. In einzelnen synthetischen Benchmarks kann der Vorteil bei bis zu 50% liegen [Bot12], bei echten Integrationsalgorithmen liegt die Performance-Steigerung eher bei 5% [NMH06] oder weniger [Kor+12].

MPI

Zur parallelen Ausführung der Simulation empfiehlt sich die Verwendung einer MPI-Bibliothek. Diese bietet Operationen wie `scatter`, `gather` und `broadcast` zur Verteilung der Daten und Synchronisierung der Prozesse. Zwei große Kandidaten hierfür sind OpenMPI [Gab+04] und MVAPICH2 [Hua+06]. Beide sind optimiert für den Einsatz auf Clustern mit Infiniband und unterstützen dabei in ihren aktuellen Versionen komplett MPI-2. Da die API standardisiert ist, kann man durch einfaches Neukompilieren der Anwendung zwischen den beiden Bibliotheken wechseln. Ein besonderer Vorteil von MVAPICH2 ist, dass die neue Version speziell auch mit Speicheradressen von GPUs arbeiten kann und mittels GPU-Direkt [NVIc] zwischen einzelnen Grafikkarten oder zwischen der Grafikkarte und dem Infiniband RDMA-Puffer kopieren kann. Dies hat gegenüber dem herkömmlichen Übertragungsmodus einen Geschwindigkeitsvorteil von über 30% [Wan+11] bei kollektiven Operationen und wird stetig weiter optimiert [Sin+11] [Ji+12]. Deswegen wird bei der GPU-Implementation ausschließlich die MVAPICH2 bei paralleler Ausführung auf mehreren GPUs verwendet.

Parallele Rechnungen (CPU)

Zum eigentlichen Rechnen mit den Daten auf der CPU kommt die relativ neue und eher unbekannte Bibliothek „Eigen“ zum Einsatz [Eig]. Diese bietet einfache Möglichkeiten, mit Vektoren, Matrizen und Arrays parallel zu rechnen und dabei die SSE/SSE2-Instruktionen der CPU auszunutzen, um die Berechnungen zu beschleunigen. SSE-Instruktionen können 4 `float`-Datenwerte parallel berechnen und SSE2-Instruktionen jeweils 2 `double`-Datenwerte. Zusätzlich werden die mathematischen Operationen durch geschicktes Ausnutzen von C++ Templates und Ausdrucksobjekten so arrangiert, dass unnötige Speicherzugriffe vermieden werden und der Prozessorcache optimal ausgenutzt werden kann. Der Pseudocode in Listing 3.1 veranschaulicht dies anhand einer Multiplikation und anschließender Addition. Kleinere Schleifen fester Länge werden außerdem automatisch ausgerollt, um auf Verzweigungsoperationen zu lasten der Code-Größe zu verzichten.

Fouriertransformation

Zur Berechnung der schnellen Fouriertransformationen (FFTs) gibt es zahlreiche frei verfügbare Bibliotheken. Eine davon, die auch für den Einsatz auf CPU-Clustern optimiert ist, ist die FFTW [FJ05]. Sie bietet verschiedene auf den jeweiligen Prozessortyp angepasste Geschwindigkeits-Optimierungen sowie eine leicht programmierbare MPI-Schnittstelle. Die FFTW gibt für den verteilten Einsatz automatisch die Verteilung der Daten auf die verschiedenen Prozesse vor. Sie bietet bei der Transformation die

```
Eigen::ArrayXf<N> a,b,c,d;
d = a * b + c;
```

(a) Programmcode in Arrayschreibweise

```
float t[N];
for (int i = 0; i < N; ++i)
    t[i] = a[i] * b[i];
for (int i = 0; i < N; ++i)
    d[i] = t[i] + c[i];
```

(b) Unoptimierter Pseudocode

```
for (int i = 0; i < N; ++i)
    d[i] = a[i] * b[i] + c[i];
```

(c) Optimierter Pseudocode

Listing 3.1: Veranschaulichung der Optimierungen innerhalb der Bibliothek „Eigen“ mittels Ausdrucksobjekten.

Möglichkeit, eine Transponierung der Daten zu sparen und den damit einhergehenden Kommunikationsaufwand. Bei Transformationen von realen Daten bietet sie die Möglichkeit, die Datenmenge im Fourierraum zu halbieren. Dabei werden nicht nur Speicherzugriffe bei der Transformation vermieden, sondern es wird dadurch auch die Berechnungszeit der Simulation auf den Daten im Fourierraum halbiert. Für die Grafikkarte stellt NVIDIA eine eigene optimierte FFT-Bibliothek namens „cuFFT“ zur Verfügung. Diese unterstützt auch die Halbierung der Daten im Fourierraum, bietet aber keine Möglichkeit, die Daten auf mehreren Grafikkarten zu verteilen. Dies kann aber relativ einfach über einen Algorithmus, wie er in Kapitel 3.5 beschrieben steht, unter Benutzung der „cuFFT“-Bibliothek für die anfallenden eindimensionalen Transformationen selbst implementiert werden.

Video-Ausgabe

Als zusätzliches Feature ist die Ausgabe von Simulations-Videos implementiert worden. Dabei werden die Zeitschritte so koordiniert, dass eine vorgegebene Simulationsgeschwindigkeit mit ausreichender FPS-Zahl im Video dargestellt werden kann. Die Kodierung des Videos erfolgt hierbei mittels `ffmpeg` [FFM] bzw. der darin enthaltenen Bibliothek `libavcodec`. Der verwendete Codec ist H.264 (auch MPEG-4 Teil 10 oder AVC genannt), da dieser eine sehr gute Qualität pro Datenrate erzielt.

Bei der GPU-Implementierung erfolgt die Datenübertragung in den Hauptspeicher ebenfalls wie bei der Datenausgabe asynchron.

Weitere Bibliotheken

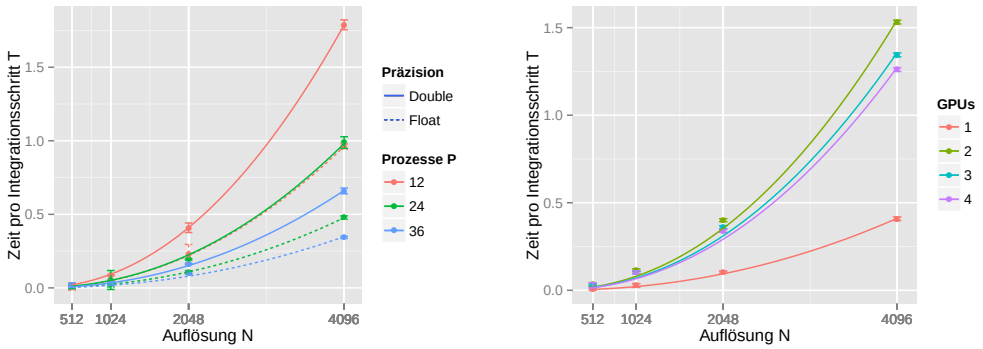
Als weitere Bibliothek wurde die „boost“-Bibliothek verwendet [Boo]. Diese bietet für C++ sehr viele Klassen für verschiedenste Anwendungsgebiete. Diese wurden für Programm-Optionen, Zeitmessung, Smart-Pointer und MPI verwendet. Außerdem wurde die Wrapper-Library „glew“ verwendet, um für den Zugriff auf OpenGL-Erweiterungen nicht jede Funktion einzeln laden müssen.

3.7. Skalierungsverhalten auf verschiedenen Hardwaresystemen

Nun soll das Skalierungsverhalten der Simulation auf verschiedenen Hardwaresystemen analysiert werden. Als verfügbare Hardwaresysteme standen hier der PALMA CPU-Cluster mit 12 CPU-Kernen pro Rechenknoten und ein Multi-GPU-System zur Verfügung. Die relevanten System-Größen sind die Anzahl der über MPI gestarteten Prozesse P und die Auflösung $N \times N$. Auf dem PALMA CPU-Cluster lief jeweils ein Prozess pro CPU-Kern und beim Multi-GPU-System ein Prozess pro GPU. Die Größen zur Messung des Skalierungsverhaltens sind die Anzahl der Integrationsschritte $I(N, P)$, die pro Sekunde durchgeführt werden können bzw. deren Inverses, die Zeit pro Integrationsschritt $T(N, P) = 1/I(N, P)$. Die gewählten Simulationsparameter finden sich im Anhang A unter `resfield`. Der gewählte Integrator war „DOPRI54“, ein Integrationsverfahren fünfter Ordnung (siehe Kapitel 3.3.2).

3.7.1. Skalierung in Abhängigkeit von der Auflösung

Man betrachtet zuerst die Abhängigkeit der Zeit pro Integrationsschritt T von der Auflösung N. Dies ist in Abbildung 3.7 graphisch für eine unterschiedliche Anzahl von Prozessen dargestellt. Wie man sieht, entspricht das Laufzeitverhalten bei unterschiedlichen Prozesszahlen und Präzisionen nahezu $T(N, P) = C'(P)\mathcal{O}(N^2 \log(N))$. Dies gilt sowohl auf dem CPU-Cluster als auch auf dem Multi-GPU-System in sehr guter Näherung und entspricht dabei dem theoretischen algorithmischen Laufzeitverhalten der Fouriertransformation. Auf beiden Systemen wurde hier mit dem Runge-Kutta-Schema DOPRI54 fünfter Ordnung integriert. Andere Integrationsschemata zeigen hier identisches Skalierungsverhalten und sind deshalb nicht extra aufgeführt. Man sieht hieran, wie die gesamte Simulation von der Performance der Fouriertransformation dominiert wird. Auf dem CPU-Cluster wurde für die Messung OpenMPI verwendet, da sie die bessere Performance besaß, wie im nächsten Abschnitt in Abbildung 3.8 zu sehen ist.



(a) CPU-Cluster, OpenMPI 1.6.2 **(b)** Multi-GPU-System

Abbildung 3.7.: Zeit pro Integrationsschritt T in Abhängigkeit von der Auflösung N. Die durchgezogene Linie stellt das Ergebnis einer Ausgleichsrechnung mit $T(N,P) \propto N^2 \log(N)$ dar. (Simulation: `resfield`, siehe Anhang A)

3.7.2. Skalierung in Abhängigkeit von der Prozesszahl

Nun soll die Laufzeit der Simulation in Abhängigkeit der Prozesszahl P untersucht werden. Hier wurden die Performancemessungen wieder sowohl auf dem CPU-Cluster als auch auf dem Multi-GPU System durchgeführt.

CPU-Cluster

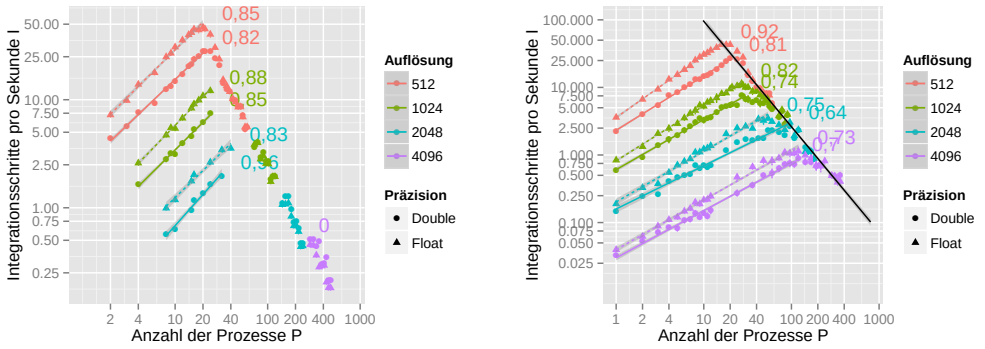
Abbildung 3.8 zeigt Ergebnisse dieses Performance-Tests für verschiedene Auflösungen in Abhängigkeit von P. Für den CPU-Cluster ergeben sich hier in doppelt logarithmischer Darstellung bis zum Performance-Maximum nahezu Geraden. Dies bedeutet, dass die Leistung nach einem Potenzgesetz der Form $I(N,P) = C(N)P^a$ steigt, wie es schon bei den FFTs in Kapitel 3.5.2 zu sehen war. Dabei ist $C(N)$ die Zeit pro Integrationsschritt, die ein CPU-Kern erreicht und P^a der für die jeweilige Anzahl von Prozessen erreichte Speedup. Ein optimales lineares Skalierungsverhalten bekäme man beim Exponenten $a = 1$.

Das komplette Skalierungsverhalten auf dem CPU-Cluster ist unter Mittelwertbildung der mit OpenMPI gemessenen Exponenten

$$I(N,P) = C(N)P^a = C\frac{P^a}{N^2\log(N)}, \quad \text{mit } a = 0,76 \pm 0,047 \tag{3.12}$$

mit $C_{\texttt{float}} = (8,60 \pm 0,831) \times 10^6$ und $C_{\texttt{double}} = (6,41 \pm 0,936) \times 10^6$.

Für den Exponenten a zeigt sich bei der MVAPICH2 kein Trend, während die OpenMPI scheinbar bei 1024×1024 mit `double`-Genauigkeit einen leichten Einbruch erleidet.



(a) MVAPICH2 1.8.1 **(b)** OpenMPI 1.6.2

Abbildung 3.8.: Anzahl der Integrationsschritte pro Sekunde I in Abhängigkeit der verwendeten Prozessorkerne P, doppelt logarithmisch dargestellt. Die Koeffizienten geben die Geradensteigung der Ausgleichsgeraden an und entsprechen dem Exponenten des exponentiellen fallenden Geschwindigkeitsanstiegs. (Simulation: `resfield`, siehe Anhang A)

Allerdings sind die Fits auch etwas schlechter auf Grund der leichten Abflachung im oberen Bereich. Die genauen Daten dieses Fits sind in Anhang B, Tabelle B.2 zu finden. Das Maximum liegt bei beiden Bibliotheken bei der gleichen Prozesszahl. Die Geschwindigkeit beider Bibliotheken ist ebenfalls nahezu identisch. Wie bei der Untersuchung der Performance der FFT in Kapitel 3.5.2 zeigt sich in dieser Messung auch eine harte Skalierungsgrenze, ab der die Performance wieder exponentiell fällt. Für diese Grenze lässt sich ein linearer Fit durchführen:

$$I_{\text{Grenz}} = GP^b = (4003,05 \pm 570,475)\, P^{(-1,6\pm0,032)} \tag{3.13}$$

Über diese Grenzgeschwindigkeit kann man die optimale Anzahl von Prozessoren berechnen:

$$C(N)P_{\text{opt}}^a = GP_{\text{opt}}^b \tag{3.14a}$$

$$P_{\text{opt}}^{a-b} = \frac{G}{C(N)} \tag{3.14b}$$

$$P_{\text{opt}}(N) = \left(\frac{G}{C(N)}\right)^{\frac{1}{a-b}} = \left(\frac{GN^2\log(N)}{C}\right)^{\frac{1}{a-b}} \tag{3.14c}$$

Hieraus ergibt sich ein maximal erreichbarer Speedup von

$$S_{\text{max}}(N) = P_{\text{opt}}(N)^a = \left(\frac{GN^2\log(N)}{C}\right)^{\frac{a}{a-b}} \tag{3.15}$$

und ebenso eine maximale Simulationsgeschwindigkeit von

$$\begin{aligned} I_{\max}(N) = C(N)P_{\text{opt}}^{a} &= C\frac{1}{N^2\log(N)}\left(\frac{GN^2\log(N)}{C}\right)^{\frac{a}{a-b}} \\ &= \left(G^a\left(\frac{N^2\log(N)}{C}\right)^{-(a-b)+a}\right)^{\frac{1}{a-b}} = \left(G^a\left(\frac{N^2\log(N)}{C}\right)^{b}\right)^{\frac{1}{a-b}} \end{aligned} \tag{3.16}$$

Diese Formel gibt die Möglichkeit, Simulationslaufzeiten für größere Auflösungen abzuschätzen.

Abbildung 3.9 zeigt die optimale Anzahl der Prozessoren $P_{\text{opt}}(N)$, den maximal erreichbaren Speedup $S_{\max}(N)$ sowie die maximale Simulationsgeschwindigkeit $I_{\max}(N)$ in Abhängigkeit von der Auflösung graphisch dargestellt. Dabei ist angnommen, dass sich sowohl der Skalierungsexponent a als auch der Exponent der Grenze b nicht mit N ändern.

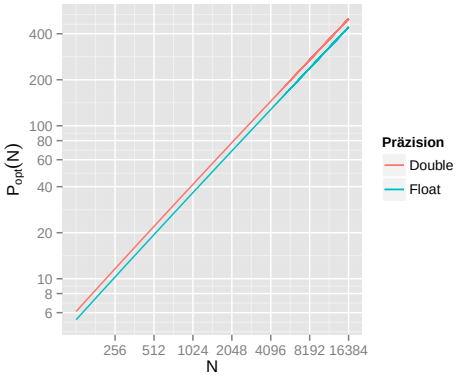
Multi-GPU System

Abbildung 3.10 zeigt nun die Ergebnisse des Performance-Tests auf dem Multi-GPU System.

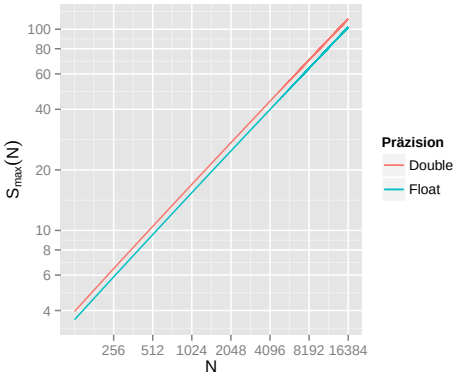
Ein direkter Vergleich mit dem PALMA CPU-Cluster ist natürlich nur möglich beim gleichen Integrationsverfahren und der gleichen Rechenpräzision. In diesem Fall ist dies das DOPRI54-Verfahren und `float`-Genauigkeit. Zunächst sieht man, dass eine einzelne GPU gut 50 mal schneller ist als ein CPU-Kern. Dabei ist sie etwa doppelt so schnell wie der CPU-Cluster bei maximaler Leistung. Im parallelen Betrieb zweier GPUs macht sich die zusätzliche Datenkommunikation durch einen starken Geschwindigkeitseinbruch auf etwa 25% der Leistung einer GPU bemerkbar. Nimmt man weitere GPUs hinzu, skaliert die Simulation Positiv mit einem Exponenten von $a \approx 0,25$. Da die Fouriertransformation selbst jedoch stark negativ skaliert, bedeutet dies, dass der zusätzliche Berechnungsaufwand der Simulation gegenüber den verwendeten Fouriertransformationen nicht vernachlässigbar ist. Diese zusätzlichen Berechnungen sind auf der Grafikkarte trivial parallelisierbar und größtenteils durch die Bandbreite des globalen Speichers begrenzt. Trotz des großen Geschwindigkeitseinbruchs ist man dennoch nur etwa 15% unterhalb der maximalen Leistung des PALMA CPU-Clusters. Die für diese Tests verwendete Grafikkarte (GeForce 480 GTX) ist eigentlich nicht für den wissenschaftlichen Betrieb konzipiert und modernere Tesla-Karten würden hier vermutlich wesentlich besser abschneiden. Nimmt man die Anschaffungs und Betriebskosten hinzu, bieten damit Grafikbeschleuniger eine gute Alternative zum Betrieb auf dem CPU-Cluster.

Im Vergleich zu [LY12] scheint die Performance dieser Implementation wesentlich besser. Dort wurde für eine Auflösung von 2048×2048 ein Speedup von nur 14 erreicht, während der hier erzielte Speedup bei gut 51 liegt. Für die Performance von zwei GPUs wurde dort ein Speedup von $1,8$x angegeben. In dieser neuen Implementation wird für diesen Fall ein Speedup von etwa 33 erreicht.

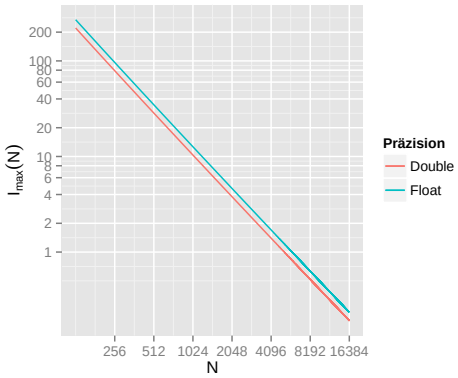
Die verschiedenen Integrationsverfahren untereinander zeigen alle ein identisches Skalierungsverhalten. Sie unterscheiden sich nur um einen konstanten Faktor, der von der Wahl des Verfahrens abhängt. Wie schon aus Abschnitt 3.3 bekannt ist, sind von den gesteten Verfahren diejenigen höherer Ordnung die schnellsten. Das RK4-Verfahren ist bei dieser Messung deshalb schneller als das DOPRI54-Verfahren, weil der Zeitschritt wesentlich größer gewählt wurde, als es das DOPRI54-Verfahren erlaubte, da dieses den relativen Diskretisierungsfehler auf 1% beschränkte.



(a) Optimale Prozesszahl

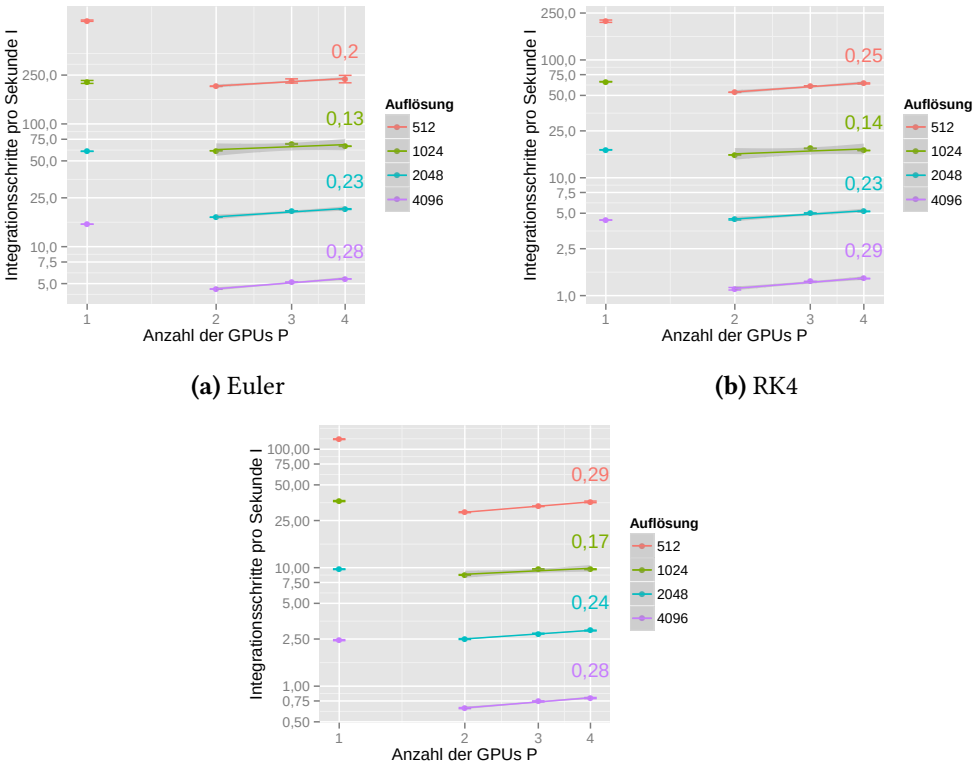


(b) Maximal erreichbarer Speedup



(c) Maximal erreichbare Simulationsgeschwindigkeit

Abbildung 3.9.: Optimale Anzahl der Prozesse $P_{\text{opt}}(N)$, den maximal erreichbaren Speedup $S_{\text{max}}(N)$ sowie die maximale Simulationsgeschwindigkeit $I_{\text{max}}(N)$ in Abhängigkeit von der Auflösung bei paralleler Ausführung der Simulation auf dem PALMA CPU-Cluster. (Simulation: `resfield`, siehe Anhang A)



(a) Euler

(b) RK4

(c) DOPRI54

Abbildung 3.10.: Anzahl der Integrationsschritte pro Sekunde I in Abhängigkeit der verwendeten GPUs P mit verschiedenen Integrationsalgorithmen, doppelt logarithmisch dargestellt. Die Koeffizienten geben die Geradensteigung der Ausgleichsgeraden an und entsprechen dem Exponenten des exponentiellen fallenden Geschwindigkeitsanstiegs. (Simulation: `resfield`, siehe Anhang A)

4. Simulationsdaten

Es wurden nun einige Simulationsläufe mit verschiedener Auflösung und unterschiedlichen Kraftparametern durchgeführt. Aus diesen Daten werden einige statistische Größen ermittelt, die bereits in Kapitel 2.5 hergeleitet wurden. Sie dienen zum einen dazu, die Korrektheit der Simulation zu prüfen, denn eine formale Beweisbarkeit der Korrektheit ist leider nicht möglich. Zum anderen sind einige dieser Größen Grundlage weiterer statistischer Analysen im anschließenden Kapitel.

4.1. Auswahl der Simulationsparameter

4.1.1. Physikalische Parameter

Zunächst einmal muss geklärt werden, welche physikalischen Parameter man testen möchte. Für diese Arbeit soll das Regime der inversen Energiekaskade untersucht werden. Die Wellenlänge der Kraft und die Viskosität wurden so gewählt, dass die inverse Energiekaskade zwar das Energiespektrum dominiert, aber die direkte Enstrophiekaskade nicht vollständig unterdrückt ist. Beide Parameter wurden für alle Messungen konstant gehalten. Zur genaueren Übersicht sind die Simulationsparameter der einzelnen Simulationen in Anhang A aufgeführt.

Unter anderem soll auch untersucht werden, wie sich physikalische Größen bei Änderung der Simulationsauflösung verhalten. Dafür wird eine Auflösung von 512×512 sowie eine Auflösung von 4096×4096 gewählt. Kraft und Viskosität sollen dabei gleich sein, so dass sich durch die höhere Auflösung nur die räumliche Genauigkeit erhöht. Dies ist insbesondere für die Auswertung der bedingten Mittelwerte in Kapitel 5 wichtig.

Außerdem soll für beide implementierten Kräfte ein möglichst breites Kraftspektrum untersucht werden, anfangen von Kräften, die grade stark genug sind, um die inverse Kaskade zu erzeugen bis hin zu Kräften, die am Rande der Rechengenauigkeit des Integrationsverfahrens liegen.

4.1.2. Rechengenauigkeit, Hardware, Integrationsverfahren

Als Rechengenauigkeit muss `double` gewählt werden, um ein möglichst breites Kraftsprektrum abzudecken. In Tests zeigte sich, dass dadurch Kräfte gewählt werden können,

die in ihrem Kraftparameter um bis zu 3 Größenordnungen stärker sind, als bei `float`-Genauigkeit.

Da leider keine Tesla-Hardware mit akzeptabler `double`-Geschwindigkeit zur Verfügung stand wurde für die Simulation der PALMA CPU-Cluster eingesetzt. Auf normalen Grafikkarten, die nicht für den wissenschaftlichen Einsatz vorgesehen sind, sind die Rechnungen mit `double`-Genauigkeit auf $\frac{1}{8}$ der `float`-Geschwindigkeit limitiert. In Linpack-Benchmarks erreicht so zum Beispiel eine Geforce GTX 680 nur eine effektive Geschwindigkeit von 30% gegenüber einer Tesla Fermi M2090 [NVId].

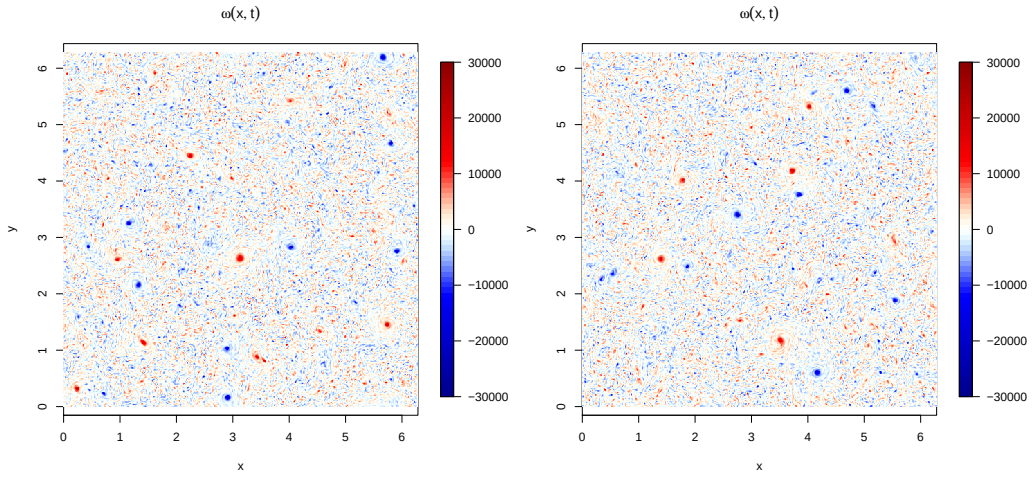
Nach der Auswertung von Kapitel 3.3.5 fällt die Wahl des Integrators entweder auf ein Runge-Kutta RK4 oder DOPRI54 Verfahren. In diesem Fall soll letzteres benutzt werden, um numerische Fehler des Integrators programmatisch begrenzen können (in diesem Fall auf 1% maximalen relativen Fehler der einzelnen Fouriermoden). Dies ist vor allem deshalb sinnvoll, da nicht für jede Krafteinstellung eine optimale Einstellung des Zeitschritts gesucht werden muss, sondern dieser automatisch vom Integrationsalgorithmus vorgegeben werden kann.

Eine ebenfalls implementierte Alternative zur Abschätzung des Zeitschritts ist das CFL-Kriterium [CFL28], das auch mit nicht-eingebetteten Verfahren funktioniert. Dieses besitzt allerdings einen einstellbaren Sicherheitsparameter, der für jede Kraftstärke speziell gewählt werden muss, damit ein optimaler Zeitschritt erreicht wird und die Simulation korrekt arbeitet. Dies bringt leider keinen wesentlichen Vorteil gegenüber der direkten Einstellung des Zeitschritts und verlangsamt dabei die Simulationsgeschwindigkeit.

4.2. Einfluss der Auflösung

Die gewählte Auflösung der Simulation hat verschiedene Auswirkungen auf die Laufzeit und das Resultat der Simulation. Mit höheren Auflösungen kleinere Strukturen und damit auch größere Fouriermoden simulieren. Die physikalische Relevanz hängt dabei davon ab, wieviele Informationen diese Fouriermoden tragen. Der Einfluss von Hypo- und Hyperviskosität beschränkt durch die starke Dämpfung äußerer Moden die Simulation auf ein einstellbares Band, sodass man zunächst nicht davon ausgeht, dass größere Auflösungen die Qualität der Simulation wesentlich beeinflussen. Um nun herauszufinden, welchen Einfluss die Auflösung tatsächlich auf verschiedene statistische Größen hat, wurden alle Simulationen mit den Auflösungen 512×512 und 4096×4096 durchgeführt.

In Abbildung 4.1 sieht man zwei turbulente Felder der inversen Kaskade mit gleichen Parametern aber unterschiedlichen Auflösungen gegenübergestellt. Qualitativ ist am Wirbelstärkefeld kein Unterschied zu bemerken. In den nachfolgenden Abschnitten werden trotzdem einige kleinere Unterschiede für verschiedene Auflösungen festgestellt.



(a) 512×512 **(b)** 4096×4096

Abbildung 4.1.: Wirbelstärkefeld zweier Simulationen mit gleichen Parametern aber unterschiedlicher Auflösung. (Simulation: `resfield`, siehe Anhang A)

4.3. Berechnung von Ensemblemitteln

Zur Auswertung vieler statistischer Größen wird das in Kapitel 2.4 beschriebene Ensemblemittel benötigt

$$\langle g(\boldsymbol{u}(\boldsymbol{x},t))\rangle_{\boldsymbol{u}} = \int_U g(\boldsymbol{u}(\boldsymbol{x}))f(\boldsymbol{u}(\boldsymbol{x}))\mathrm{d}u \tag{4.1}$$

Die Wahrscheinlichkeitsdichte $f(\boldsymbol{u}(\boldsymbol{x}))$ beschreibt dabei, mit welcher Wahrscheinlichkeit ein bestimmtes Feld $\boldsymbol{u}(\boldsymbol{x},t) \in U$ vorkommt. Um dieses Ensemblemittel aus Simulationsdaten bilden zu können, wird angenommen, dass N beliebige nicht miteinander korrelierte Felder $\boldsymbol{u}_i(\boldsymbol{x}) = \boldsymbol{u}(\boldsymbol{x},t_i)$ verschiedener Simulationszeiten t_i mit gleicher Wahrscheinlichkeit auftreten, also

$$\langle g(\boldsymbol{u}(\boldsymbol{x},t))\rangle_{\boldsymbol{u}} \approx \frac{1}{N}\sum_i g(\boldsymbol{u}_i) \tag{4.2}$$

Dabei muss natürlich zunächst gewährleistet werden, dass die Simulation den stationären Zustand erreicht hat. Wie man dies herausfindet, wird in Kapitel 4.4 genauer erklärt. Die Bedingung, dass zwei konkrete zeitlich aufeinanderfolgende Felder unkorreliert sind, wird dadurch angenähert, dass die Korrelation beider Wirbelstärkenfelder $\omega_i(\boldsymbol{x})\boldsymbol{e}_z = \nabla \times \boldsymbol{u}_i(\boldsymbol{x})$ unter eine Schranke fällt, z.B.

$$C(\omega_i,\omega_j) < 0,01 \tag{4.3}$$

Dies ist einfacher zu berechnen, als die Korrelation der Geschwindigkeitsfelder. Zu beachten ist, dass für gleiche Felder gilt $C(\omega_i, \omega_i) = 1$. Die Korrelation ist in diesem Fall gegeben als

$$C(\omega_i, \omega_j) = \frac{\left\langle \left(\omega_i - \langle\omega_i\rangle_{\boldsymbol{x}}\right)\left(\omega_j - \langle\omega_j\rangle_{\boldsymbol{x}}\right)\right\rangle_{\boldsymbol{x}}}{\sqrt{\left\langle\omega_i^2\right\rangle_{\boldsymbol{x}}\left\langle\omega_j^2\right\rangle_{\boldsymbol{x}}}} \tag{4.4}$$

mit

$$\langle g(\omega_i)\rangle_{\boldsymbol{x}} = \int g(\omega_i(\boldsymbol{x}))\mathrm{d}^2x \tag{4.5}$$

4.4. Stationärer Zustand

Statistische Ensemblemittel als Zeitmittel lassen sich erst aus einer Simulation berechnen, wenn sich das Energiespektrum nicht mehr wesentlich in seiner Form verändert. Man spricht dabei von einem stationären Zustand. In Abbildung 4.2 ist der zeitliche Verlauf des Exponenten der inversen Kaskade zu sehen. Da die Energiespektren der stochastischen Kraft in diesem Bereich nicht linear sind, ergibt sich für diesen Fall kein wirklich aussagekräftiger Exponent. Jedoch ist bei beiden Kräften deutlich zu sehen, dass für die untersuchten Parameterbereiche spätestens ab $t/T_{\text{integral}} = 100$ der stationäre Zustand erreicht ist.

Leider ist das Energiespektrum eine etwas unhandliche Größe, auf die man keine schöne Metrik anwenden kann. Außerdem ist die Berechnung des Exponenten der inversen Kaskade recht aufwändig, sodass es sich anbietet, andere Variablen in ihrem zeitlichen Verlauf zu beobachten. Notwendige Bedingung für die Stationarität des Spektrums ist natürlich, dass die Gesamtenergie konstant bleibt. In Abbildungen 4.3 bis 4.6 ist der Werteverlauf einiger wichtiger Größen, wie z.B. der Gesamtenergie oder der Energiedissipationsrate dargestellt.

Zur Erstellung dieser Graphen wurden für alle Parametereinstellungen die jeweils 100 ersten paarweise unter 1% korrelierten Wirbelstärkefelder verwendet, die nach dem Verfahren in Kapitel 4.3 bestimmt wurden. Man kann nach Analyse dieser Graphen sagen, dass spätestens im letzten Datensatz für jede untersuchte Größe der stationäre Zustand erreicht ist. Daher können für spätere Ensemblemittelung alle folgenden Datensätze verwendet werden.

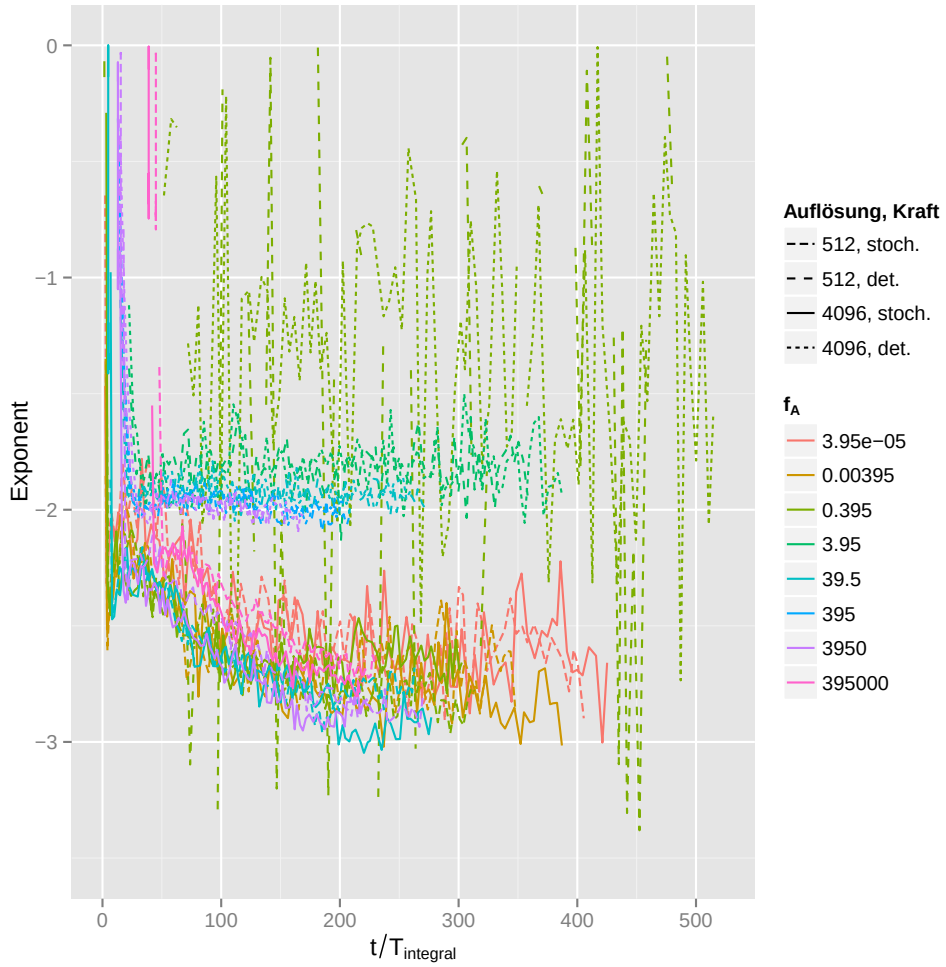
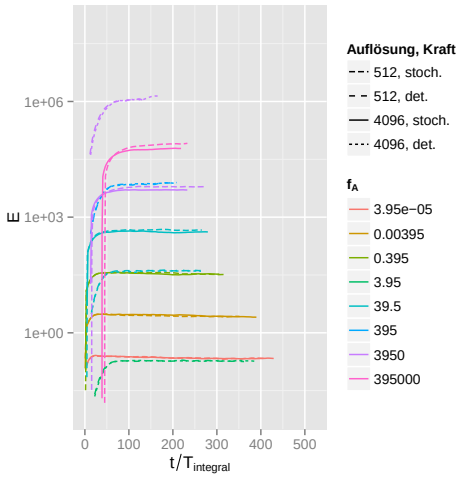
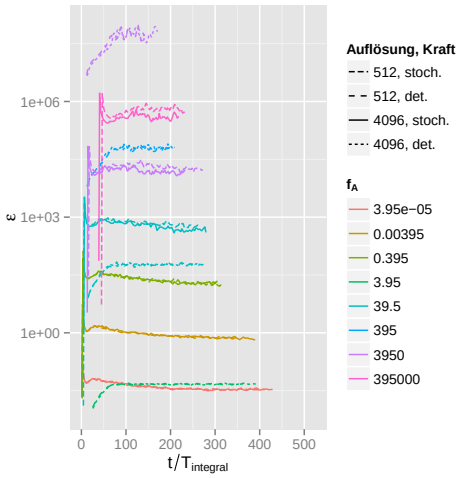


Abbildung 4.2.: Zeitlicher Verlauf des Exponenten der inversen Kaskade, berechnet über lineare Regression des Bereichs zwischen Integraler Länge und Kraft-Länge im Energiespektrum

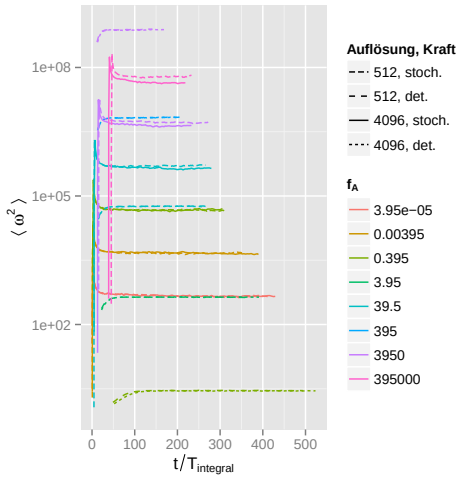


(a) Gesamtenergie pro Einheitsmasse (Kapitel 2.5.4)

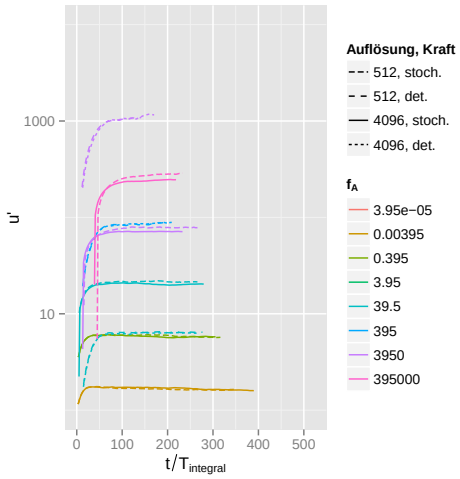


(b) Energiedissipationsrate (Kapitel 2.5.4)

Abbildung 4.3.: Zeitlicher Verlauf einiger statischtisch relevanter Größen, skaliert auf die Integrale Zeit (Simulation: `resforce`, siehe Anhang A)

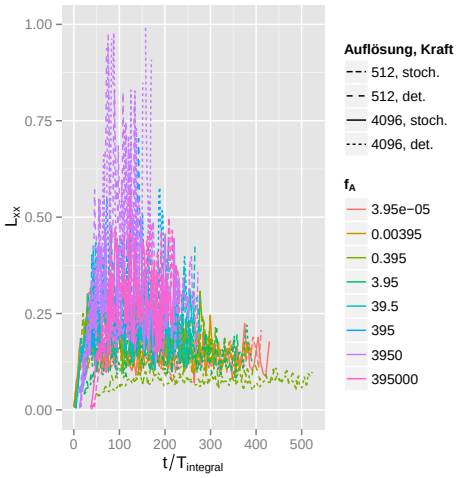


(a) Betragsquadrat der Wirbelstärke (Kapitel 2.5.1)

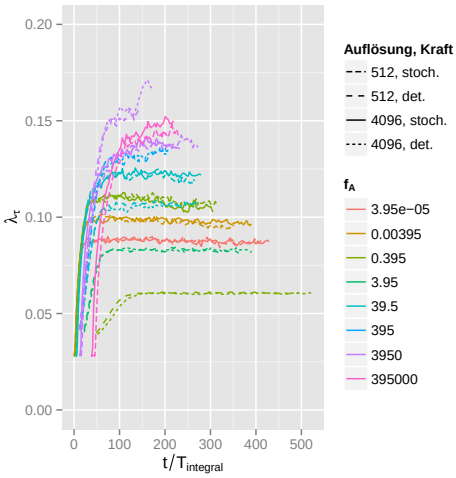


(b) Mittlere Geschwindigkeit (Kapitel 2.5.1)

Abbildung 4.4.: Zeitlicher Verlauf einiger statischtisch relevanter Größen, skaliert auf die Integrale Zeit (Simulation: `resforce`, siehe Anhang A)

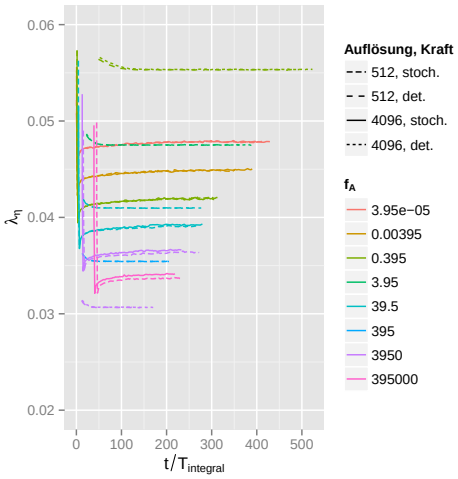


(a) Integrale Länge
(Kapitel 2.5.5)

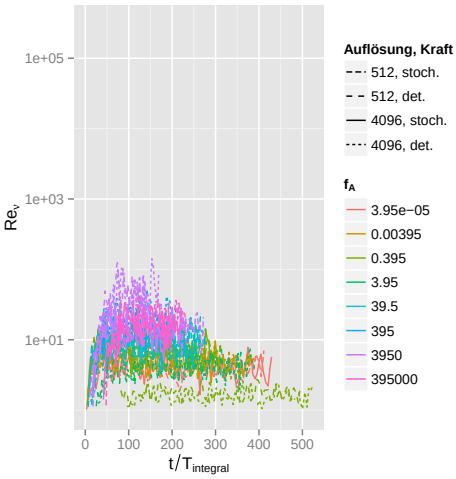


(b) Taylor-Länge
(Kapitel 2.5.5)

Abbildung 4.5.: Zeitlicher Verlauf einiger weiterer statischtisch relevanter Größen, skaliert auf die Integrale Zeit (Simulation: `resforce`, siehe Anhang A)



(a) Kolmogorov-Länge
(Kapitel 2.5.5)



(b) Hyperviskose Reynoldszahl
(Kapitel 2.5.6)

Abbildung 4.6.: Zeitlicher Verlauf einiger weiterer statischtisch relevanter Größen, skaliert auf die Integrale Zeit (Simulation: `resforce`, siehe Anhang A)

4.5. Energie- und Enstrophiekaskade

Wie in Kapitel 2.5.4 beschrieben, erwartet man das Auftreten einer Energie- und Enstrophiekaskade, deren Längen im Wesentlichen von der Viskosität und der Positon der Kraftmoden abhängen. Abbildung 4.7 zeigt die inverse Energiekaskade bei unterschiedlichen Kraftparametern. Die Enstrophiekaskade ist hier durch die Hyperviskosität stark unterdrückt, aber dennoch in den Graphen zu sehen.

4.5.1. Exponent des Inertialbereichs

Auffällig ist, dass die deterministische Kraft bei ähnlichen Kraftamplituden im Fourierraum einen wesentlich längeren und glatteren Inertialbereich erzeugt, als die stochastische. Berechnet man die Exponenten der inversen Kaskade für diese Simulationen, so erhält man die Werte, die in Tabelle 4.1 zu sehen sind. Entgegen der

Auflösung \ Kraftstärke	$4\pi^2 \cdot 10^{-2}$	$4\pi^2 \cdot 10^{-1}$	$4\pi^2 \cdot 10^{0}$	$4\pi^2 \cdot 10^{1}$	$4\pi^2 \cdot 10^{2}$
512	$-1,17$	$-1,78$	$-1,93$	$-2,13$	$-2,10$
4096	$-1,22$	$-1,78$	$-1,91$	$-2,03$	$-2,04$

Tabelle 4.1.: Exponenten der inversen Kaskade für verschiedene Kraftstärken und Auflösungen im Falle einer deterministischen Kraft. Bestimmung des Exponenten über lineare Regression im Energiespektrum zwischen der integralen Länge und der Kraft-Länge (Simulation: `resforce`, siehe Anhang A)

Kolmogorov-Theorie ist hier der Exponent für die meissten Fälle nicht nahe bei $-\frac{5}{3}$, sondern eher bei -2. In [Sco07] wird dies dadurch erklärt, dass die Simulationen die Kraftmoden, welche sich hier wegen der Hyperviskosität nicht im dissipativen Bereich befinden, sehr gut auflöst und sich eine kleine direkte Kaskade im Energiespektrum ausbildet, die aber keine charakteristischen Einflüsse im Wirbelstärkefeld zeigt.

4.5.2. Peak bei hohen Moden

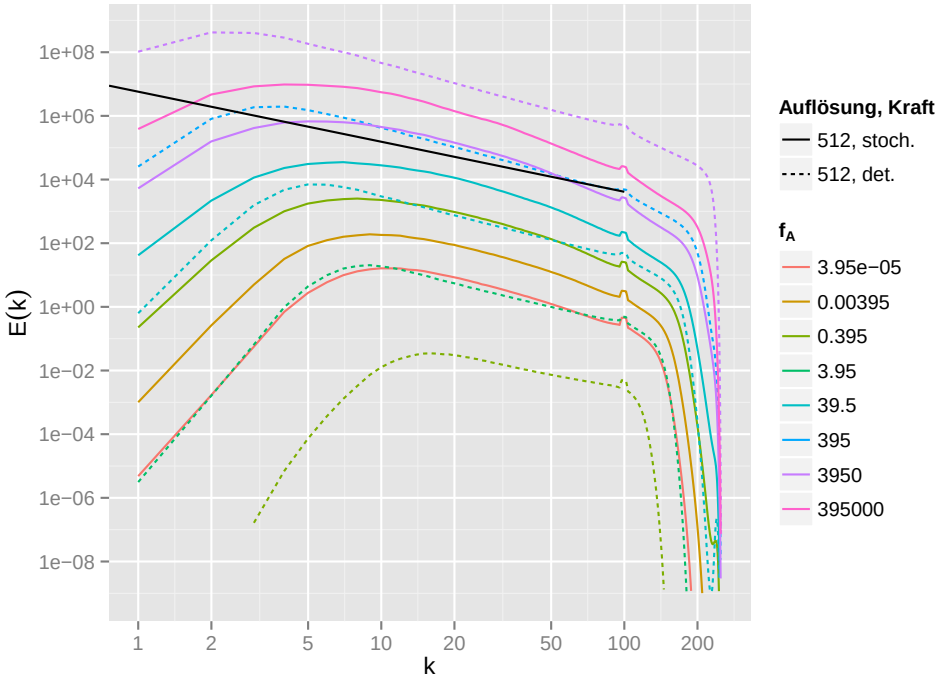
Bei der kleineren Auflösung von 512×512 findet man manchmal einen zusätzlichen Peak bei großen Fouriermoden k, wo eigentlich die Hyperviskosität greift. Dieser könnte zum einen ein Überschwinger der Fouriertransformation sein, der bei mehrmaliger Transformation von harten Kanten auftritt. In der Literatur ist dies als Gibbs-Wilbraham-Phänomen bekannt [HH79]. Zum anderen wäre es möglich, dass der Peak durch den sogenannten Bottleneck-Effekt entsteht, wie er in [Fal94] oder [Fri+08] beschrieben ist und in 3D sogar zu einem veränderten Spektrum mit $E(k) \propto k^2$ am Dissipationsrand führt. Dies muss nicht unbedingt negative Auswirkungen auf die Simulation haben, da die Fouriermoden dort sehr stark gedämpft werden, aber in

manchen Simulationen kam es bei Auftreten dieses Effektes zu einem sichtbaren Streifenmuster. Daher sollte man möglichst nur Simulationen zur weiteren Auswertung wählen, die diesen Effekt nicht zeigen.

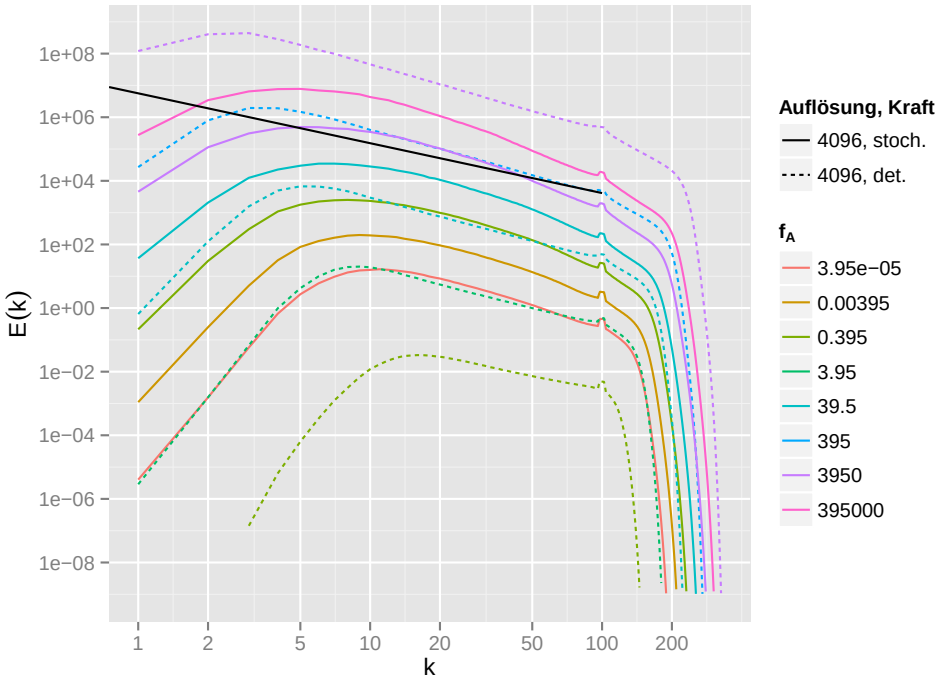
4.5.3. Energie- und Enstophiefluss

Für das Zustandekommen der Kaskaden ist der Energie- bzw. Enstrophiefluss verantwortlich. Diese beiden Größen sind in Abbildungen 4.8a und 4.8b zu sehen.

Der Fluss ist positiv, falls Energie oder Enstrophie zu größeren Modenzahlen transportiert wird und andernfalls negativ. Man sieht, dass von der Kraftmode k_f aus Energie in beide Richtungen transportiert wird, aber mehrheitlich zu kleineren Moden. Die Enstrophie hingegen wird ausschließlich zu höheren Moden transportiert.

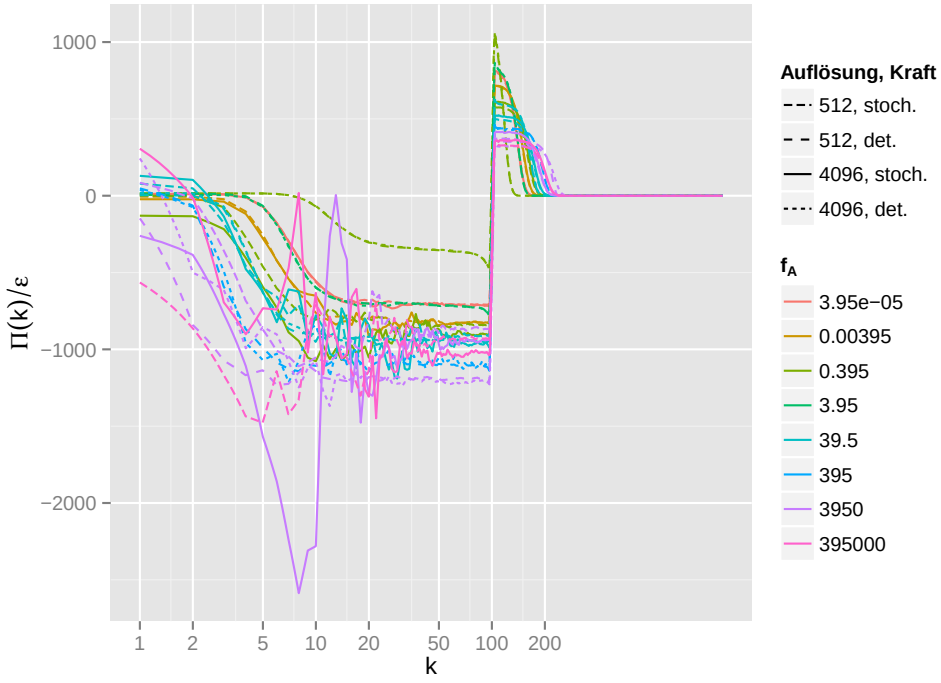


(a) 512×512

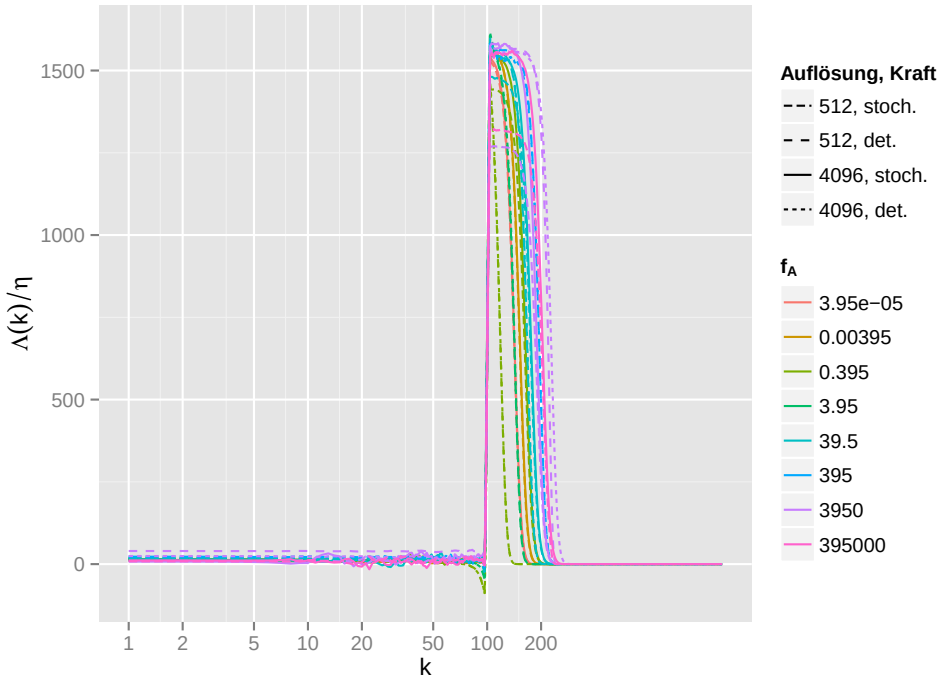


(b) 4096×4096

Abbildung 4.7.: Energienspektren verschiedener Simulationen bei unterschiedlichen Kraftstärken, Typen und Auflösungen. Die durchgezogene Linie entspricht der erwarteten Steigung von $-\frac{5}{3}$. (Simulation: `resforce`, siehe Anhang A)



(a) Energiefluss



(b) Enstrophiefluss

Abbildung 4.8.: Energie- und Enstrophiefluss verschiedener Simulationen bei unterschiedlichen Kraftstärken, Typen und Auflösungen. (Simulation: `resforce`, siehe Anhang A)

4.6. Wahrscheinlichkeitsdichte

Die Wahrscheinlichkeitsdichten für die Wirbelstärke $f_n(\omega_1, \boldsymbol{x}_1, ..., \omega_n, \boldsymbol{x}_n)$ spielen in der statistischen Turbulenz eine zentrale Rolle. Sie beschreiben die Wahrscheinlichkeit, an den Orten $\boldsymbol{x}_i$ jeweils die Wirbelstärke ω_i zu finden. Im Kapitel 5 werden einige Annahmen über Form und Symmetrie dieser Verteilungen gemacht, die hier an einfachen Beispielen untersucht werden.

4.6.1. 1-Punkt-Verteilung

In Abbildung 4.9 ist die Verteilung $f_1(\omega)$ halblogarithmisch dargestellt. Eine gaußförmige Verteilung entspräche in dieser Darstellung einer Parabel. Bei der stochastischen

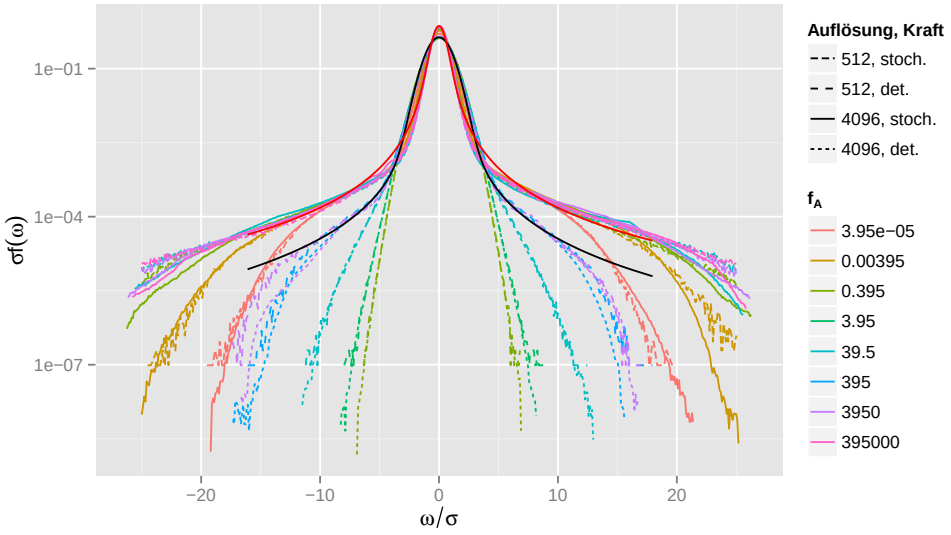


Abbildung 4.9.: Wahrscheinlichkeitsdichte $f_1(\omega)$ berechnet bei verschiedenen Kräften und Auflösungen. Als schwarze und rote Linie sind zwei stabile Verteilungen $f_s(\omega/\sigma)$ eingezeichnet mit den Parametern $\beta = 1,93$, $\gamma = 0,44$ für die schwarze Linie und $\beta = 1,6$, $\gamma = 0,22$ für die rote. (Simulation: `resforce`, siehe Anhang A)

Kraft nimmt für den gewählten Parameterbereich die Verteilung keine gaußsche Form an. Eventuell tritt dies erst für wesentlich kleiner gewählte Kraftamplituden ein. Je größer man die Kraftstärke wählt, je stärker ausgeprägt werden die Flügel der Verteilung. Der gleiche Effekt ist auch in [Sco07] zu sehen. Im Limes großer Kräfte scheinen die Verteilungen der stochastischen und deterministischen Kraft gegen zwei unterschiedliche Verteilungen zu streben. In Abbildung 4.9 sind daher zwei stabile Verteilungen eingezeichnet, die etwas besser mit der Verteilung der Simulationsdaten

übereinstimmen. Eine Einführung zu den stabilen Verteilungen findet man in [Nol12]. Parametrisiert werden die symmetrischen stabilen Verteilungen im Fourierraum über ihre charakteristische Funktion:

$$f_s(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \varphi(\alpha) \mathrm{e}^{-i\alpha x} \mathrm{d}\alpha, \quad \text{mit } \varphi(\alpha) = \mathrm{e}^{-W(\alpha)} = \mathrm{e}^{-|\gamma\alpha|^\beta} \tag{4.6}$$

Der Exponent $W(\alpha)$ der charakteristischen Funktion ist in Abbildung 4.10 dargestellt. Dort sieht man, dass bei stochastischer Kraft die charakteristische Funktion für Exponenten bis $\sigma\alpha = 10$ hinreichend gut durch eine stabile Verteilung beschrieben werden kann. Diese stabile Verteilung weicht allerdings stark von der Normalverteilung mit $\beta = 2$ und $\gamma = 0,5$ ab. Bei deterministischer Kraft hingegen weichen β und γ nur gering von der Normalverteilung ab. Dafür gilt die Übereinstimmung nur bis ca. $\sigma\alpha = 5$. Die Ränder der charakteristischen Funktion bei $W(\alpha) \approx 10$ beschreiben hochfrequentes Rauschen niedriger Amplitude in der Wahrscheinlichkeitsdichte und dürften aus numerischen Fehlern resultieren.

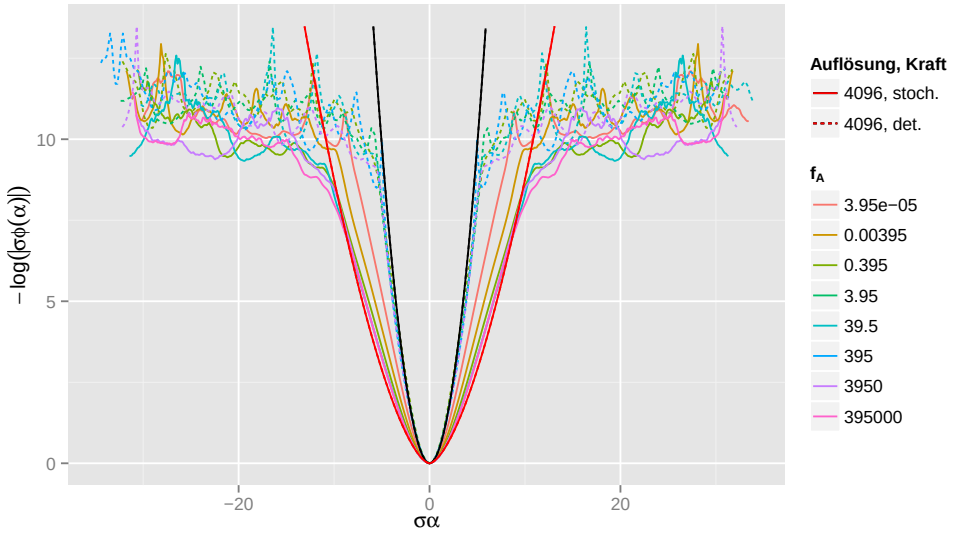
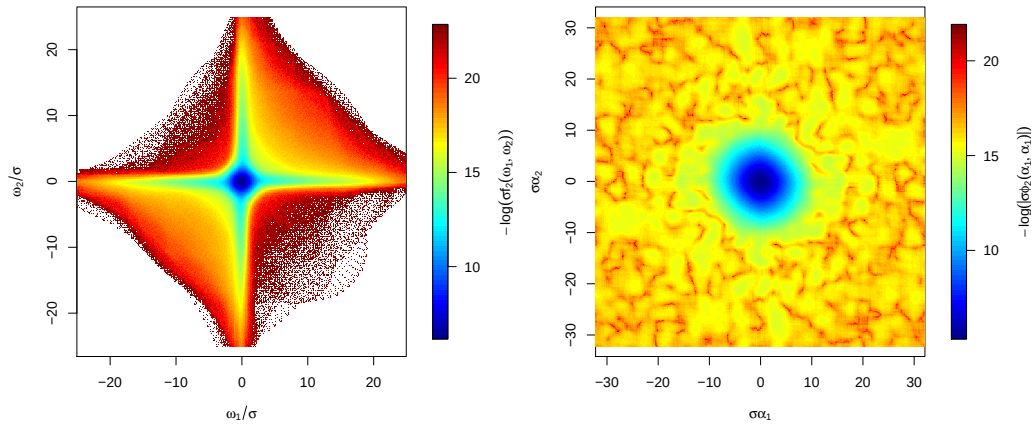


Abbildung 4.10.: Exponent der charakteristischen Funktion $\varphi(\alpha)$ berechnet bei verschiedenen Kräften. Die Parameter der eingezeichneten roten und schwarzen Kurve entsprechen denen aus Abbildung 4.9. (Simulation: `resforce`, siehe Anhang A)

4.6.2. 2-Punkt-Verteilung

Die Verteilung und der Exponent ihrer charakteristischen Funktion für zwei Punkte ist in Abbildung 4.11 dargestellt. Man sieht hier deutlich die aus Symmetrieüberlegungen



(a) Wahrscheinlichkeitsdichte **(b)** Exponent der charakteristischen Funktion

Abbildung 4.11.: Wahrscheinlichkeitsdichte $f_2(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2)$ und der Exponent der charakteristischen Funktion $W_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2) = -\log(\varphi(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2))$ für zwei Punkte im Abstand $\boldsymbol{x}_2 - \boldsymbol{x}_1 = \lambda_f \boldsymbol{e}_x$. (Simulation: `4096inverse`, $f_A = 4\pi^2 \cdot 10^{-2}$, siehe Anhang A)

erwartete Punktsymmetrie, die später in Kapitel 5.6 eine wesentliche Bedeutung für den Aufbau der charakteristischen Funktion erlangt. Die auftretenden Muster an den Randbereichen der Wahrscheinlichkeitsdichte sind vermutlich der Tatsache geschuldet, dass die Datenfelder im Ensemblemittel noch eine sehr geringe Korrelation zueinander aufweisen. Sie sind allerdings nur auf Grund der logarithmischen Darstellung zu sehen. Dies wird auch die Ursache des Musters im äußeren Bereich der charakteristischen Funktion sein.

4.7. 2-Punkt Wirbelstärkekorrelation

Wie in Kapitel 2.5.1 für Geschwindigkeitsfelder vorgestellt kann man auch für die Wirbelstärke die 2-Punkt Korrelationsfunktion $\langle \omega_z(\boldsymbol{x}, t)\omega_z(\boldsymbol{x} + \boldsymbol{r}, t)\rangle_{\boldsymbol{u},\boldsymbol{x}}$ berechnen. Dies geschieht am einfachsten im Fourierraum, da dort die Faltung einer einfachen

Multiplikation entspricht.

$$\begin{aligned} f_\omega(\boldsymbol{r},t) &= \langle \omega_z(\boldsymbol{x},t)\omega_z(\boldsymbol{x}+\boldsymbol{r},t)\rangle_{\boldsymbol{u},\boldsymbol{x}} \\ &= \left\langle \frac{1}{L^2}\int \omega_z(\boldsymbol{x},t)\omega_z(\boldsymbol{x}+\boldsymbol{r},t)\mathrm{d}^2x \right\rangle_{\boldsymbol{u}} \\ &= \left\langle \frac{1}{L^2}\mathcal{F}^{-1}_{\boldsymbol{k}}\left[\omega_z^*(\boldsymbol{k},t)\omega_z(\boldsymbol{k},t)\right](\boldsymbol{r},t) \right\rangle_{\boldsymbol{u}} \end{aligned} \tag{4.7}$$

Auf Grund der Isotropie macht es Sinn, diese Funktion radial zu mitteln und eindimensional in $r = |\boldsymbol{r}|$ darzustellen.

$$f_\omega(r) = \frac{1}{2\pi}\int f_\omega(\boldsymbol{r},t)\mathrm{d}\phi \tag{4.8}$$

In Abbildung 4.12 ist die 2-Punkt Wirbelstärkekorrelationsfunktion beispielhaft für eine Simulationsreihe zweidimensional dargestellt. Hier kann man sehen, wie sehr

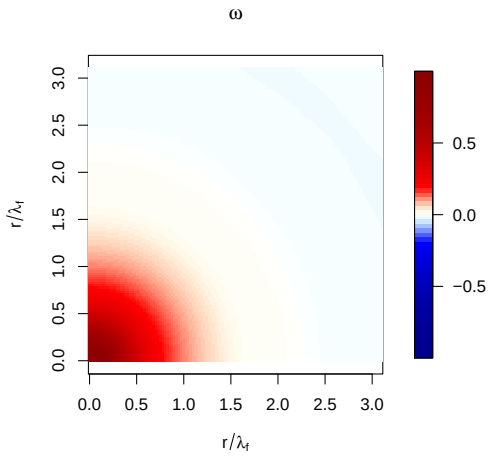
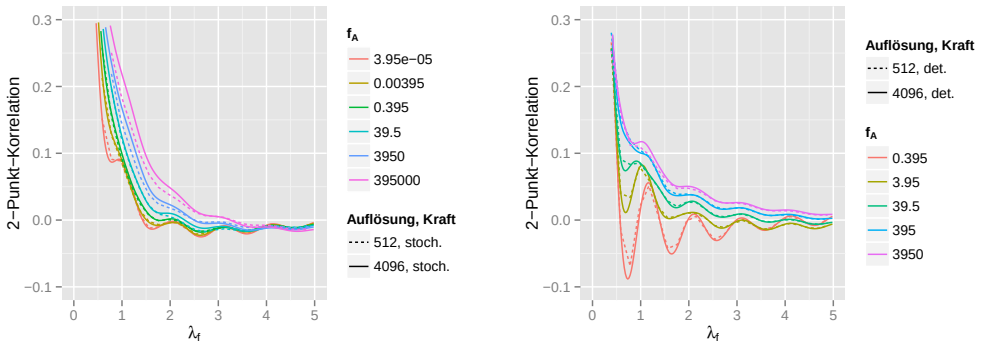


Abbildung 4.12.: 2-Punkt Wirbelstärkekorrelation in 2D. (Simulation: `resforce` mit $N_x = N_y = 4096$, $f_A = 4\pi^2 \cdot 10^0$ bei stochastischer Kraft, siehe Anhang A)

genau die Annahme der Isotropie erfüllt ist. In Abbildung 4.13 ist für verschiedene Kraftparameter die radial gemittelte 2-Punkt Wirbelstärkekorrelationsfunktion zu sehen. Bei beiden Krafttypen ist zu erkennen, dass bei kleineren Kraftamplituden stärkere Oszillationen mit der Kraftwellenlänge zu beobachten sind. Dies hängt mit der Länge und Stärke der inversen Energiekaskade zusammen. Wie in Abbildung 4.7 zu erkennen ist, verlängert sich mit zunehmender Kraftstärke der Inertialbereich und ebenso der Abstand zwischen maximaler Amplitude und der Amplitude der Kraftmode.



(a) **(b)**

Abbildung 4.13.: Radial gemittelte 2-Punkt Wirbelstärkekorrelation für verschiedene Kraftparameter. (Simulation: `resforce`, siehe Anhang A)

Dies sorgt dafür, dass der Peak im Energiespektrum, den die Kraft verursacht, im Ortsraum bei steigender Amplitude immer kleinere Auswirkungen hat.

In der Gegenüberstellung verschiedener Auflösungen sieht man deutlich, dass bei einer geringen Simulationsauflösung die Wirbelstärkekorrelation nur schlecht approximiert wird. Hier ist es ratsam, die Simulationsauflösung soweit zu erhöhen, bis man eine gute Auflösung der Wirbelstärkekorrelation erreicht. Zusätzlich sieht man eine leichte Verschiebung der Kurven gegeneinander, vermutlich resultierend aus numerischen Ungenauigkeiten.

Im Vergleich der stochastischen und deterministischen Kraft ist zu sehen, dass bei deterministischer Kraft deutlich stärkere Oszillationen auftauchen. Natürlich ist dabei zu berücksichtigen, dass man nur Kräfte mit ähnlichem Energiespektrum vergleicht, so z.B. $f_A = 4\pi^2 \cdot 10^{-6}$ stochastisch mit $f_A = 4\pi^2 \cdot 10^{-2}$ deterministisch, $f_A = 4\pi^2 \cdot 10^{-4}$ stochastisch mit $f_A = 4\pi^2 \cdot 10^{0}$ deterministisch oder auch $f_A = 4\pi^2 \cdot 10^{2}$ stochastisch mit $f_A = 4\pi^2 \cdot 10^{1}$ deterministisch.

4.8. Aufbereitung der Simulationsdaten

Die Auswertungsprogramme sind, um den Code der Simulation wiederverwenden zu können, wie diese in C++ geschrieben und unterstützen zu großen Teilen ebenso die Ausführung auf dem Cluster. Einige Auswertungen sind auch mit CUDA für Grafikkarten geschrieben, weil dies einen wesentlichen Geschwindigkeitsvorteil brachte. Alle statistischen Größen werden als Ensemblemittel über ihre entsprechende Formel berechnet, wie sie in Kapitel 2 oder 4.3 angegeben worden sind. Die Parallelisierung und Verteilung der Daten erfolgt hier ebenso wie bei der Simulation. Das heißt, dass die

Felder jeweils im Real- und Fourierraum auf die berechnenden Prozesse verteilt sind. Für die meisten statistischen Größen erfolgt die Berechnung über Mittelwertbildung relativ trivial. Von größerem Interesse sind hier die Besonderheiten der Implementation der Histogrammbildung oder der Interpolation.

4.8.1. Histogrammbildung

Will man Aussagen über die Wahrscheinlichkeitsdichten machen, so gibt es in der Numerik verschiedene Möglichkeiten diese zu approximieren. Eine davon ist die Histogrammbildung, die bei der Beschaffenheit der vorliegenden Daten sehr gute Ergebnisse liefert. Hier wird die kontinuierliche Wahrscheinlichkeitsdichte durch eine diskrete Verteilung approximiert. Dazu wird der Ergebnisraum in eine diskrete Anzahl von Teilmengen unterteilt. Diesen Vorgang nennt man „binning“. In der Regel sind die Bins gleich breit, überdecken den Ergebnisraum lückenlos und überschneiden sich nicht.

Ein solches Binning lässt sich auf einfache Art und Weise parallel durchführen. Dies soll am Beispiel der Histogrammbildung über die Wirbelstärken im Ensemblemittel $\langle\omega(\boldsymbol{x},t)\rangle_{\omega,\boldsymbol{x}}$ dargestellt werden. Es wird dazu angenommen, dass die Daten so verteilt sind, dass jeder Prozess einen ihm zugehörigen Teil des Wirbelstärkefeldes verwaltet. Die Werte pro Feld müssen nun in einem lokalen Histogramm einsortiert werden. Dafür kann man über eine einfache Operation mit Hilfe der „eigen“-Bibliothek jedem Wert seinen Histogrammindex zuordnen lassen: Dabei werden die Wirbel in `wsteps`

```
Eigen::Array<int, Eigen::Dynamic, Eigen::Dynamic> windex;
windex = ((wmax+field) / (2*wmax)*wsteps + 1.5f).cast<int>() - 1;
```

Listing 4.1: Zuordnung von Histogrammindex zur jeweiligen Wirbelstärke im Feld `field`

„bins“ von `-wmax` bis `wmax` eingeordnet. Anschließend muss mit einer lokalen Schleife das lokale Histogramm berechnet werden. Dieses soll im Array `count[wsteps]` liegen. Diese lokalen Histogramme kann man im Sinne des Ensemblemittels als Zeitmittel zunächst über alle vorliegenden Datenfelder berechnen, bevor man sie anschließend über alle Prozesse hinweg addiert und normiert. Das geschieht dann mittels einer `reduce`-Operation: Die Skalierung berücksichtigt hierbei die Bingröße, sodass eine

```
count = boost::mpi::all_reduce(world, count,
            std::plus<FieldUtils::X>());

count /= Nx * Ny * files.size() * (2.0 * wmax) / wsteps;
```

Listing 4.2: Reduzierung des lokalen Histogramms in `count` und anschließende Normierung

anschließende Integration über die Bins den korrekten Wert liefert.

4.8.2. Bedingte Mittelwerte

In Kapitel 5 geht es um die Näherung von bedingten Mittelwerten des Wirbelstärkefeldes durch die Wahrscheinlichkeitsdichte. Solche bedingten Mittelwerte lassen sich formal darstellen als

$$\langle \omega(\boldsymbol{x},t) | \omega(\boldsymbol{x}_1,t) = \omega_1, ..., \omega(\boldsymbol{x}_n,t) = \omega_n \rangle_\omega \tag{4.9}$$

Dabei wird an die Punkte $\boldsymbol{x}_1$ bis $\boldsymbol{x}_n$ die Bedingung gestellt, dass dort das Wirbelstärkefeld die Werte ω_1 bis ω_n annimmt. Auf Grund der Translationsinvarianz ist dies das gleiche wie

$$\langle \omega(\boldsymbol{x},t) | \omega(\boldsymbol{x}_1,t) = \omega_1, \omega(\boldsymbol{x}_1+\boldsymbol{r}_2,t) = \omega_2, ..., \omega(\boldsymbol{x}_1+\boldsymbol{r}_n,t) = \omega_n \rangle_{\omega,\boldsymbol{x}_1} \tag{4.10}$$

Die zusätzliche Annahme der Isotropie bietet außerdem die Möglichkeit, das Wirbelstärkefeld vor Ensemblemittelung beliebig zu rotieren

$$\langle \omega(R\boldsymbol{x},t) \rangle_\omega = \langle \omega(\boldsymbol{x},t) \rangle_\omega = \langle \omega(|\boldsymbol{x}|,t) \rangle_\omega \tag{4.11}$$

Der Algorithmus zur Implementation muss nun unter Ausnutzung der Translationsinvarianz alle Punkte $\boldsymbol{x}_1$ finden, auf die alle Bedingungen zutreffen. Der Vergleich der Wirbelstärken kann nicht exakt stattfinden, da nur endlich viele Simulationsdaten vorliegen. Man wählt daher einen Toleranzbereich $\Delta\omega$ um die Wirbelstärke herum, in dem Punkte akzeptiert werden. Die Abstandsvektoren müssen außerdem so angepasst werden, dass sie den nächstmöglichen Punkt in der gewählten Auflösung treffen. Unter Berücksichtigung von Isotropie kann man die vorliegende Simulationsdatenmenge einfach vervierfachen, indem man jedes Datenfeld nacheinander um 90°, 180°und 270°rotiert. Das Wirbelstärkefeld muss dann anschließend um alle gefundenen Punkte herum gemittelt werden. Dies geschieht dadurch, dass das die Punkte umgebende Feld jeweils so auf ein Ausgabefeld aufaddiert wird, sodass dort der Punkt $\boldsymbol{x}_1$ immer an der gleichen Stelle des Ausgabefeldes ist.

Dieser Algorithmus wurde für 1 bzw. 2 Punkte auf CPU-Cluster und GPU implementiert.

CPU-Cluster

Die CPU-Implementation benutzt wieder die Verteilung der Daten, die durch die FFT vorgegeben wird. Ganze Zeilen im Realraum liegen hierbei immer auf dem gleichen CPU-Kern. Dies ist zur Berechnung des auf 2 Punkte bedingten Mittelwerts wichtig:

Durch Isotropie wird angenommen, dass der Verbindungsvektor in x-Richtung liegen kann und somit in der gleichen Zeile. Zur Erinnerung sei gesagt, dass die Verteilung der Fouriertransformation vorsieht, dass Zeilen im Ortsraum immer ganz auf der gleichen Recheneinheit liegen. Nachdem der eigene Teil des Feldes durchsucht wurde, werden die gefundenen Punkte per `allreduce` an alle Prozesse verteilt. Jeder Prozess prüft nun, ob er eine Umgebung des jeweiligen Punktes besitzt und addiert diese an die richtige Stelle seines Ausgabefeldes. Abschließend werden alle Ausgabefelder addiert und durch die Anzahl der gefundenen Punkte geteilt, um den korrekten Mittelwert zu erhalten.

GPU

Die GPU-Implementation wurde nur für eine einzelne GPU durchgeführt, da die vorliegenden Felddaten leicht auf eine GPU passen. Die Punktsuche gestaltet sich hier so, dass jeder Thread auf der Grafikkarte einem Punkt im Quellfeld zugeordnet wird und für diesen Punkt überprüft, ob er auf die vorgegebenen Bedingungen passt. Die gefunden Punkte werden dann von den Threads in eine Punktliste geschrieben, deren aktueller Schreibindex durch eine atomare Operation inkrementiert wird.

Im zweiten Schritt wird über das Ausgabefeld parallelisiert. Jeder Thread ist nun dafür verantwortlich für alle Punkte aus der Punkteliste im Quellfeld den entsprechend passenden Punkt zu lesen und auf seinen Ausgabewert zu addieren. Die Punktliste wird hierbei pro Block abschnittsweise in den lokalen Speicher kopiert, um die Leseoperation daraus zu beschleunigen. Hier wird ausgenutzt, dass alle Threads die Punktliste sequentiell abarbeiten und dass Kopiervorgänge aus dem globalen Speicher schneller sind, wenn ein zusammenhängendes Stück von mehreren Threads gelesen wird. Über alle Ausgabefelder wird abschließend per CPU gemittelt, da dieser Vorgang von der Speicherbandbreite limitiert wird.

Vergleich der Laufzeiten

In Tabelle 4.2 werden die Laufzeiten der Implementierungen für CPU-Cluster und GPU bei verschiedene Auflösungen miteinander verglichen. Wie man sieht hängt die Geschwindigkeit wesentlich von der Auflösung ab. Mit höherer Auflösung erhöht sich ebenso die Anzahl der gefundenen Punkte und damit auch den Berechnungsaufwand. Man sieht, dass die CPU-Implementierung wesentlich schlechter mit der Auflösung skaliert. So braucht sie bei 8 CPU-Kernen und einer Auflösung von 4096 noch gut 31 mal länger als bei einer Auflösung von 512. Die GPU dahingegen benötigt nur knapp 14 mal länger. Die Vorteile der GPU dürften hier bei vor allem bei der höheren Speicherbandbreite (siehe Kapitel 2.6.3) liegen.

Auflösung	GPU	1 CPU	2 CPUs	4 CPUs	8 CPUs
512	1,14s	15,1s	14,4s	10,9s	12,0s
4096	15,7s	17min 11s	14min 47s	9min 53s	6min 18s

Tabelle 4.2.: Laufzeiten der Berechnung eines auf einen Punkt bedingten Mittelwertes mit $\omega_1 = 2\sigma$ und $\Delta\omega = 0,0005 \cdot 2\sigma$ über 100 Felder in einer 256×256 Pixel großen Umgebung. Simulation `resforce`, $f_A = 4\pi^2 \times 10^2$ und deterministischer Kraft (siehe Anhang A).

4.8.3. Statistik und Diagramme mit „R“

Zur Erstellung von Diagrammen und bedingt auch zur Weiterverarbeitung der berechneten Daten eignet sich besonders die Skriptsprache „R“ [Cha08], eine freie Variante der Skriptsprache „S“. Sie unterstützt viele Zusatzmodule, unter anderem eines, um HDF5-Dateien einzulesen (`hdf5`) oder ein anderers, um verschiedenste Datenplots zu erstellen (`ggplot2`). Vektorielle Operationen auf Arrays sind fest in die Sprache integriert, was das Schreiben von Schleifen für Berechnungen auf ein Minimum reduziert und eine gute interne Hardwarebeschleunigung ermöglicht. Desweiteren bietet sie sowohl alle Vorteile einer Skriptsprache inklusive dynamischer Evaluation, als auch eine Art Objektorientierung.

Zusätzlich bietet „R“ auf Grund der Tatsache, dass die meisten Funktionen vektoriell sind, die Möglichkeit, diese mit dem Befehl `pvec` parallel auf den verfübaren Prozessor-Kernen eines Rechners auszuführen. Dies beschleunigt lange Operationen, wie z.B. `dstable` auf einer Maschine nahezu linear.

4.9. Interpolation

Zur Bildung von guten Ensemblemitteln benötigt man viele Simulationsdaten. Da die Laufzeit der Simulation etwa bei $\mathcal{O}(N^2 \log(N))$ liegt, möchte man natürlich die Auflösung möglichst gering halten. Bei kleineren Auflösungen leidet damit auch die Auflösung der ausgewerteten Daten. Dies wurde in Kapitel 4.7 bei der Berechnung der 2-Punkt Wirbelstärkekorrelation gesehen (Abbildung 4.13). Da für die vorliegenden Simulationen nur weniger als 200 Fouriermoden aktiv sind, stellt sich die Frage, ob nicht eine Interpolation der Daten möglich ist. Da die Auswertungsprogramme in der Regel nur eine Laufzeit von $\mathcal{O}(N^2)$ besitzen, wäre dies auf jeden Fall von Vorteil. In diesem Kapitel sollen erst einmal zwei Interpolationsverfahren kurz vorgestellt, dann ihr Laufzeitverhalten auf GPU und CPU miteinander verglichen und abschließend ihre Qualität untersucht werden. Als Grundlage des Vergleiches soll die Berechnung des 1-Punkt bedingten Mittelwertes (siehe Abschnitt 5.3.1) dienen, dessen Auswertung der Hauptbestandteil der folgenden Kapitel sein wird.

4.9.1. Fourier-Interpolation

Die Fourierinterpolation gestaltet sich besonders einfach. Hier sind die Fourierkoeffizienten, die durch die neuen höheren Moden hinzukommen alle null. Deshalb muss das Feld im Fourierraum nur in der Mitte mit der richtigen Anzahl von Nullen aufgefüllt und anschließend zurücktransformiert werden.

4.9.2. Bikubische Interpolation

Die bikubische B-Spline Interpolation ist ein Spezialfall eines Mitchell-Netravali Filters [MN88] mit bikubischen Polynomen. In [MSW06] wurde diese gegen andere Interpolationsmethoden getestet und erzielte akkurate Resultate für die Berechnung von Lagrange-Teilchen. Für die Interpolation werden die folgenden zwei vektoriellen Funktionen benötigt:

$$v_1(t) = \frac{1}{6} \begin{pmatrix} 6 - 2B \\ 0 \\ -18 + 12B + 6C \\ 12 - 9B - 6C \end{pmatrix} \cdot \begin{pmatrix} 1 \\ |t| \\ |t|^2 \\ |t|^3 \end{pmatrix} \tag{4.12}$$

$$v_2(t) = \frac{1}{6} \begin{pmatrix} 8B + 24C \\ -12B - 48C \\ 6B + 30C \\ -B - 6C \end{pmatrix} \cdot \begin{pmatrix} 1 \\ |t| \\ |t|^2 \\ |t|^3 \end{pmatrix} \tag{4.13}$$

Im Falle der bikubischen B-Spline Interpolation ist $B = 1$ und $C = 0$. Aus diesen zwei Polynomen lässt sich ein zusammengesetztes Polynom $p(t)$ berechnen

$$p(t) = \begin{cases} v_1(t) & 0 \leq |t| < 1 \\ v_2(t) \quad \text{für} & 1 \leq |t| < 2 \\ 0 & \text{sonst} \end{cases} \tag{4.14}$$

Für eine Interpolation an Position $t = x - \lfloor x \rfloor$ zu den Stützstellen $x_i(x) = \lfloor x \rfloor + i$ muss nun berechnet werden

$$\tilde{f}(x) = \begin{pmatrix} f(x_2(x)) \\ f(x_1(x)) \\ f(x_0(x)) \\ f(x_{-1}(x)) \end{pmatrix} \cdot \begin{pmatrix} p(2-t) \\ p(1-t) \\ p(-t) \\ p(-1-t) \end{pmatrix} \tag{4.15}$$

Dieses Verfahren lässt sich auf der CPU mit SSE Vektoroperationen beschleunigen, da diese die auftretenden Skalarprodukte einfach parallelisieren. Auf der GPU besteht außer der trivialen Implementierung noch die Möglichkeit, ein spezielles Verfahren, wie es in [PF05] beschrieben steht, zu verwenden. Dabei wird die besondere Fähigkeit der GPU ausgenutzt, aus dem Texturspeicher linear interpolierte Werte zu lesen. Um die lineare Interpolation im bisherigen Algorithmus zu sehen, definiert man dazu

$$g_0(x) = p(-1-t) + p(-t) \quad h_0(x) = \frac{p(-t)}{g_0(x)} \tag{4.16}$$

$$g_1(x) = p(1-t) + p(2-t) \quad h_1(x) = \frac{p(2-t)}{g_1(x)} \tag{4.17}$$

und benutzt, dass $g_0(x) + g_1(x) = 1$ ist. Damit ist die bikubische Interpolation über drei lineare Interpolationen darstellbar:

$$\begin{aligned} \tilde{f}(x) = (1 - g_1(x)) \left[(1 - h_0(x)) f(x_{-1}(x)) + h_0(x) f(x_0(x))\right] \\ + g_1(x)) \left[(1 - h_1(x)) f(x_1(x)) + h_1(x) f(x_2(x))\right] \end{aligned} \tag{4.18}$$

Die Gewichtungsfunktionen g_0, h_0 und h_1 sind periodisch und lassen sich in einer eindimensionalen Textur speichern. Die beiden inneren Interpolationen können über einen linear interpolierten Texturzugriff automatisch von der GPU-Hardware ausgeführt werden. Diese Rechenmethode verringert den Arbeitsaufwand für die GPU beträchtlich und lässt sich problemlos auf mehrere Dimensionen erweitern.

Die Implementation der bikubischen Interpolation ist trivial über das Ausgabefeld parallelisierbar. Dabei wird durch zweidimensionale Texturzugriffe die Datenlokalität durch den Textur-Cache ausgenutzt. Der Code der Interpolation ist so kurz, dass es sich anbietet, die Interpolationsdaten vor der Verwendung nicht wieder zurück in den Grafikkartenspeicher zu schreiben, sondern direkt zu verarbeiten. Dadurch wird nicht nur Speicher, sondern auch viel Speicherbandbreite gespart, die in vielen Fällen den Algorithmus limitiert.

4.9.3. Laufzeitverhalten der Interpolation

Um eine Übersicht über die Beschleunigung durch die GPU zu erhalten, wurden nun auf einen Punkt bedingte Mittelwerte des Wirbelstärkefeldes $\langle \omega(\boldsymbol{x}, t) | \omega(\boldsymbol{x}_1, t) = \omega_1 \rangle_{\omega, \boldsymbol{x}_1}$ mit und ohne bikubische Interpolation berechnet (siehe Tabelle 4.3). Zunächst einmal schaut man sich die beiden normalen Fälle an, die nicht interpolieren. Bei der kleinen Auflösung von 512×512 ist die GPU etwa 15x schneller als ein CPU-Kern. Der Vorteil der GPU wird bei größeren Auflösungen stark ausgebaut. Sie ist bei einer Auflösung von 4096×4096 ca. 66 mal schneller als ein CPU-Kern. Selbst unter Hinzunahme weiterer CPU-Kerne gelangt man nicht in die Nähe der Geschwindigkeit der GPU (24 mal schneller als 8 CPU-Kerne). An diesem Geschwindigkeitsvorteil sind

	GPU	1 CPU	2 CPUs	4 CPUs	8 CPUs
Ohne Interpolation ($512 \to 512$)	1,14s	15,1s	14,4s	10,9s	12,0s
Ohne Interpolation ($4096 \to 4096$)	15,7s	17min 11s	14min 47s	9min 53s	6min 18s
Mit Point-Sampling ($512 \to 4096$)	1,72s	13min 10s	8min 46s	5min 4s	3min 32s
Mit bikubischer Interpolation ($512 \to 4096$)	2,4s	40min 31s	22min 15s	12min 46s	7min 50s

Tabelle 4.3.: Berechnungszeit eines auf einen Punkt bedingten Mittelwertes mit $\omega_1 = 2\sigma$ und $\Delta\omega = 0,0005 \cdot 2\sigma$ über 100 Felder in einer 256×256 Pixel großen Umgebung. Simulation `resforce`, $f_A = 4\pi^2 \times 10^2$ und deterministischer Kraft (siehe Anhang A).

vermutlich sowohl Speicherdurchsatz des Grafikkartenspeichers als auch Caching und parallele Ausführung der Grafikkarte verantwortlich, da der Algorithmus nahezu optimal parallelisierbar war, wie in Abschnitt 4.8.2 beschrieben wurde.

Nun soll das Point-Sampling betrachtet werden. Hier wird statt ordentlicher Interpolation einfach der näheste Punkt gesucht und dessen Wert genommen. Es ist interessant, dass die CPU dabei schneller ist, als bei einem größeren Quelldatenfeld. Dies kann damit zu tun haben, dass der Prozessor die Quelldaten dadurch besser cachen kann, dass häufig identische Punkte nacheinander aus dem Speicher gelesen werden. Die GPU zeigt das gleiche Verhalten noch stäker und ist hierbei um Faktor 9 schneller als bei einem größeren Quellfeld. Dies liegt vermutlich daran, dass der Cache von Grafikkarten für Texturzugriffe in einem kleinen zweidimensionalen Gebiet optimiert ist. Durch das Point-Sampling werden 64 mal mehr Punkte gelesen bzw. geschrieben. Dies bringt allerdings auf der GPU nur einen Geschwindigkeitsnachteil von ca. 50%, was dafür ziemlich gering ist. Die einfache lineare Interpolation dauert bei Tests auf der GPU gleich lang und ist deshalb hier nicht speziell aufgeführt. Dies bedeutet, dass die lineare Interpolation beim Texturzugriff auf die verwendeten Datenmengen nur unerheblich mehr Zeit benötigt.

Die bikubische Interpolation braucht auf der CPU trotz SSE-Optimierung 3 bis 4 mal so lange wie mit Point-Sampling. Dies liegt an der Implementation der bikubischen Interpolation, die pro Punkt neue Polynome berechnen muss. Hier bestünde eventuell Optimierungspotential, da häufig die gleichen Polynome benötigt werden. Die GPU-Implementation zeigt sich eher unbeeindruckt von der zusätzlichen Aufgabe, obwohl hier 5 zusätzliche Texturzugriffe (2 Interpolationsdaten + 4 linear interpolierte Feldwerte statt 1 (linear interpolierter) Feldwert) gegenüber dem Point-Sampling bzw. der linearen Interpolation nötig sind. Diese sind deshalb so schnell, weil die Grafikkarte in ihren Textureinheiten per Hardware interpoliert und die Felddaten durch den

Textur-Cache liest, der für zweidimensionale Zugriffe optimiert ist. Vermutlich wird immer noch ein Großteil der Rechenzeit dafür benötigt, die Daten über die PCIe-Bus zu übertragen.

Die Implementation der Fourier-Interpolation ist hier nicht direkt vergleichbar, da sie vor der Auswertung ein zusätzliches interpoliertes Datenfeld erzeugt und auf diesem arbeitet, wie ohne aktive Interpolation.

Von der Geschwindigkeit her wäre die bikubische Interpolation gegenüber der Variante ohne Interpolation deutlich vorzuziehen, da zum einen die Auswertung mit kleineren interpolierten Datenfeldern wesentlich schneller läuft und zum anderen die Simulation wegen ihrer Laufzeit von $\mathcal{O}(N^2 \log(N))$ stark von der kleineren Auflösung profitiert. Es bleibt nur noch auszuwerten, wie gut die Auswertung der interpolierten Daten mit den nicht interpolierten Daten übereinstimmt.

4.9.4. Güte der Interpolationen

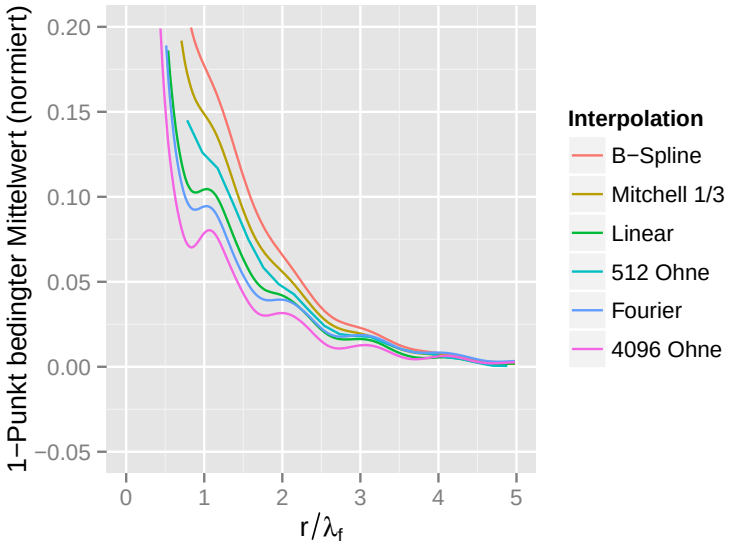
Nun sollen die vorgestellten Interpolationsarten miteinander verglichen werden. Da die Berechnung von bedingten Mittelwerten stark unter Auflösungsproblemen leidet, war dies die erste Wahl, um die verschiedenen Interpolationsarten zu testen. Abbildung 4.14 zeigt die Resultate zur Berechnung des auf einen Punkt bedingten Mittelwertes des Wirbelstärkefeldes $\langle \omega(\boldsymbol{x}, t) | \omega(\boldsymbol{x}_1, t) = \omega_1 \rangle_{\omega, \boldsymbol{x}_1}$. Zunächst einmal sieht man, dass die 512er Kurve („512 Ohne“) weit oberhalb der 4096er Kurve („4096 Ohne“) liegt, wobei man letztere gerne approximieren würde. Die Fourier-Interpolation kommt dieser Kurve am nächsten, die lineare Interpolation ist etwas schlechter. Beide Kurve liegen zwischen den 512er Originaldaten und den in höherer Auflösung berechneten 4096er Daten. Die bikubische B-Spline Interpolation ($B = 1$, $C = 0$) und die bikubische Interpolation mit ($B = \frac{1}{3}$, $C = \frac{1}{3}$), die von Mitchell und Netravali [MN88] vorgeschlagen wurde, sind beide weiter entfernt von den zu approximierenden Daten als die Original-Daten. Dass bikubische Interpolationen die Daten schlechter approximieren würde als lineare, war nicht zu erwarten.

In Abbildung 4.14b ist zu sehen, dass der über B-Spline-Interpolation approximierte bedingte Mittelwert vollständig radialsymmetrisch ist, wie dies durch die Isotropie gefordert ist. Dies stellt ein Indiz dafür dar, dass der Algorithmus fehlerfrei arbeitet. Systematische Verschiebungen würden hier direkt auffallen.

Die Resultate zur Berechnung des auf zwei Punkte bedingten Mittelwerts des Wirbelstärkefeldes $\langle \omega(\boldsymbol{x}, t) | \omega(\boldsymbol{x}_1, t) = \omega_1, \omega(\boldsymbol{x}_1 + \boldsymbol{r}, t) = \omega_2 \rangle_{\omega, \boldsymbol{x}_1}$ sind für die beiden besten Interpolationsmethoden, also die Fourier-Interpolation und die lineare Interpolation in Abbildung 4.15 zu sehen. Zunächst sieht man die Auflösungsprobleme in der Original-Auflösung von 512×512 deutlich. Im zweiten Bild sind die Daten einer 4096×4096 Simulation zu sehen, die man gerne durch Interpolation erreicht hätte. Bei den beiden

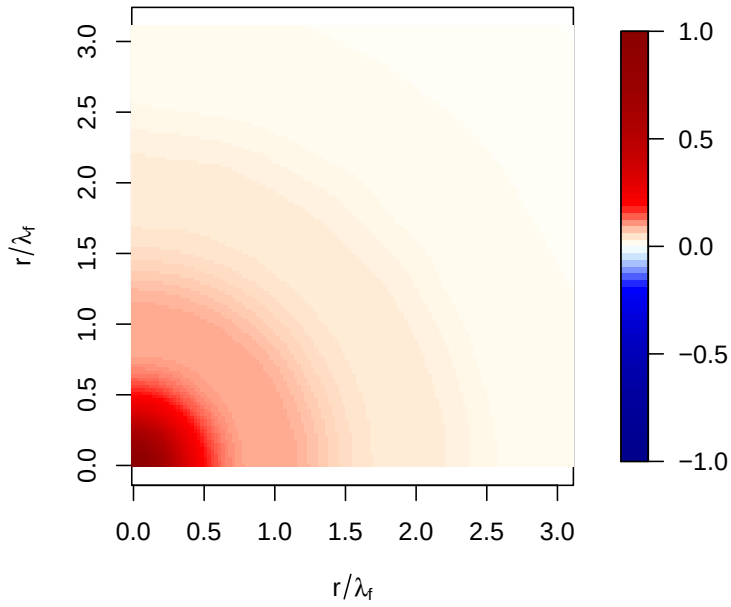
Interpolationsvarianten sieht man eine kleine Aufweitung um die bedingten Punkte herum. Die Tropfenform der Wirbelstrukturen wird bei beiden nahezu identisch wiedergegeben.

Da es im folgenden Kapitel 5 ausschließlich um diese bedingten Mittelwerte geht, wird für diese Arbeit die exakte Rechnung mit höherer Auflösung vorgezogen. Für andere Anwendungsgebiete könnte die Interpolation allerdings auf Grund der teilweise geringen Abweichungen und hohen Simulations- und Analysegeschwindigkeiten erhebliche Vorteile bieten.



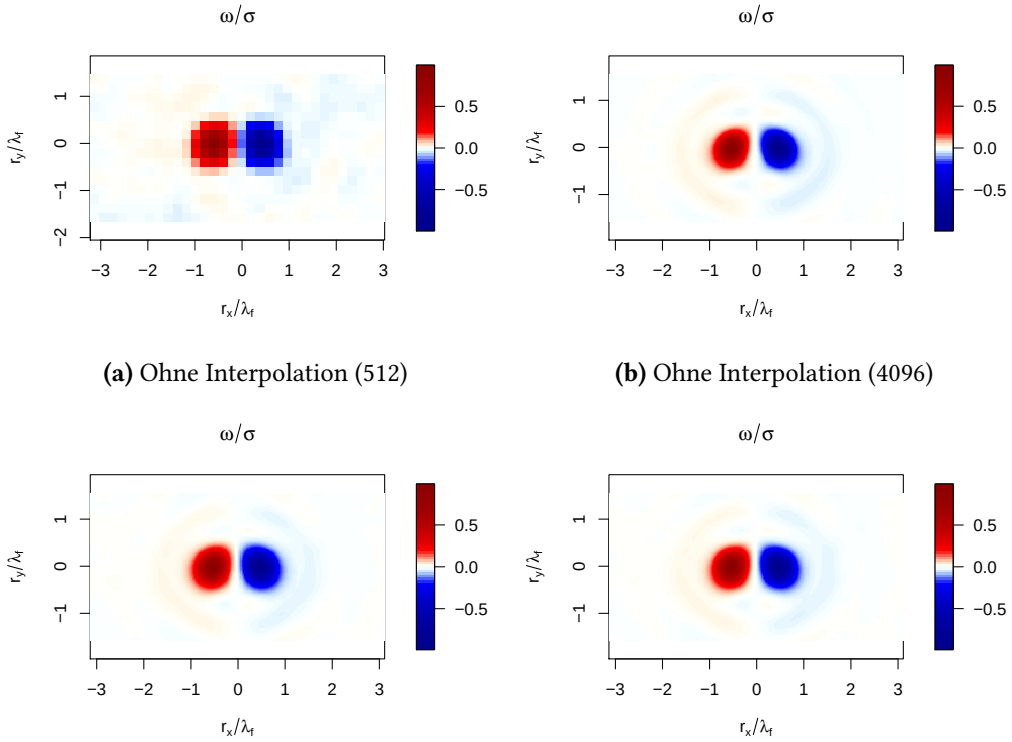
(a) Verschiedene Interpolationen

ω



(b) B-Spline Interpolation

Abbildung 4.14.: Radial gemittelter und normierter auf einen Punkt bedingter Mittelwert für verschiedene Interpolationsmethoden mit $\omega_1 = 2\sigma$ und über 100 Felder. Simulation `resforce`, $f_A = 4\pi^2 \times 10^2$ und deterministischer Kraft (siehe Anhang A).



(a) Ohne Interpolation (512)

(b) Ohne Interpolation (4096)

(c) Lineare Interpolation ($512 \rightarrow 4096$)

(d) Fourier-Interpolation ($512 \rightarrow 4096$)

Abbildung 4.15.: Auf zwei Punkte bedingter Mittelwert für verschiedene Interpolationsmethoden mit $\omega_1 = 2\sigma$, $\omega_2 = -2\sigma$ und $\Delta\omega = 0,05 \cdot 2\sigma$ über 100 Felder. Simulation `resforce`, $f_A = 4\pi^2 \times 10^2$ und deterministischer Kraft (siehe Anhang A).

5. Entwicklungsgleichungen, Wahrscheinlichkeitsverteilungen und bedingte Mittelwerte

In diesem Kapitel soll es darum gehen, ein Modell für die Wahrscheinlichkeitsverteilung des Wirbelstärkefeldes zu finden, das die auf eine verschiedene Anzahl von Punkten bedingten Mittelwerte des Wirbelstärkefeldes gut approximiert.

Um die Bedeutung dieser bedingten Mittelwerte zu kären, wird zunächst ein statistischer Zugang hergeleitet, der im Wesentlichen auf der Lundgren-Monin-Novikov (LMN) Hierarchie basiert ([Lun67], [Mon67] und [Nov68]) und auch schon in [Fri+10] benutzt wurde. Hier werden zeitliche Entwicklungsgleichungen der Wahrscheinlichkeitsdichte hergeleitet, bei denen das Schließungsproblem der Turbulenz auftaucht. Diese Gleichungshierarchie kann man auf bedingte Mittelwerte übertragen und über die Methode der Charakteristiken Bewegungsgleichungen für auf N Punkte bedingte Mittelwerte spezifizieren.

Als mögliche Schließung des Gleichungssystems kann man nun Modelle für die Wahrscheinlichkeitsverteilungen erstellen, das im Idealfall physikalisch begründet werden kann. Hierfür werden die Wahrscheinlichkeitsverteilungen genauer untersucht. Dabei wird festgestellt, dass diese nicht einer gaußschen Verteilung, sondern eher einer stabilen Verteilung ähneln. Bei der Modellbildung wird innerhalb eines allgemeinen Polynomansatzes zunächst einmal der Fall der gaußschen Verteilung untersucht, der als einziger bisher schon untersucht wurde. Dabei werden die Unterschiede zu den numerisch aus der Simulation gewonnenen bedingten Mittelwerten gezeigt. Im Anschluss wird ein erweitertes gaußsches Modell untersucht, bei dem Hermite-Polynome eine zentrale Rolle spielen. Ebenso wird ein Modell betrachtet, in dem die Verteilung durch eine bivariate stabile Verteilung approximiert wird.

Der letzte und beste Ansatz ist der einer Faltung von verschiedenen Verteilungen. Dieser Ansatz kann anders als alle vorherigen die auf einen Punkt bedingten Mittelwerte erstaunlich gut vorhersagen. Innerhalb dieses Ansatzes spielt der Kraftterm offenbar eine besondere Rolle.

5.1. Wirbeltransportgleichung für die Wahrscheinlichkeitsdichte

Entsprechend der in [Lun67] vorgestellten Methode soll nun wie auch in [Fri+10] eine Evolutionsgleichung für die Wahrscheinlichkeitsdichte ausgerechnet werden. Das Prinzip ist recht simpel: Zunächst berechnet man die Ableitung der feinkörnigen Verteilung $\tilde{f}_n$

$$\tilde{f}_n(\boldsymbol{x}_1, \omega_1, ..., \boldsymbol{x}_n, \omega_n, t) = \prod_i \delta(\omega_i - \omega(\boldsymbol{x}_i, t)) \tag{5.1}$$

Die Ableitung dieser feinkörnigen Verteilung ist dann:

$$\begin{aligned}
&\frac{\partial}{\partial t}\tilde{f}_n(\boldsymbol{x}_1, \omega_1, ..., \boldsymbol{x}_n, \omega_n, t) \\
&\qquad = -\sum_i \left[\prod_{j\neq i} \delta(\omega_j - \omega(\boldsymbol{x}_j, t))\right] \delta'(\omega_i \\
&\qquad\qquad - \omega(\boldsymbol{x}_i, t)) \frac{\partial}{\partial t}\omega(\boldsymbol{x}_i, t) \\
&\qquad = -\sum_i \left[\prod_{j\neq i} \delta(\omega_j - \omega(\boldsymbol{x}_j, t))\right] \frac{\partial}{\partial \omega_i}\delta(\omega_i \\
&\qquad\qquad - \omega(\boldsymbol{x}_i, t)) \frac{\partial}{\partial t}\omega(\boldsymbol{x}_i, t) \\
&\qquad = -\sum_i \frac{\partial}{\partial \omega_i}\tilde{f}_n(\boldsymbol{x}_1, \omega_1, ..., \boldsymbol{x}_n, \omega_n, t) \frac{\partial}{\partial t}\omega(\boldsymbol{x}_i, t)
\end{aligned} \tag{5.2}$$

Zum Weiterrechnen benötigt man nun die Wirbeltransportgleichung

$$\frac{\partial}{\partial t}\omega(\boldsymbol{x}, t) = -\boldsymbol{u}(\boldsymbol{x}_i, t) \cdot \nabla_{\boldsymbol{x}_i}\omega(\boldsymbol{x}_i, t) + \mathcal{L}\omega(\boldsymbol{x}_i, t) + F(\boldsymbol{x}_i, t) \tag{5.3}$$

Diese setzt man nun in die Ableitung von $\tilde{f}_n$ ein und erhält

$$\begin{aligned}
&\frac{\partial}{\partial t}\tilde{f}_n(\boldsymbol{x}_1,\omega_1,...,\boldsymbol{x}_n,\omega_n,t) \\
&\qquad = -\sum_i \frac{\partial}{\partial \omega_i}\tilde{f}_n(\boldsymbol{x}_1,\omega_1,...,\boldsymbol{x}_n,\omega_n,t)\,[-\boldsymbol{u}(\boldsymbol{x}_i,t) \\
&\qquad\qquad \cdot \nabla_{\boldsymbol{x}_i}\omega(\boldsymbol{x}_i,t) + \mathcal{L}\omega(\boldsymbol{x}_i,t) + F(\boldsymbol{x}_i,t)] \\
&\qquad = -\sum_i \Bigg[\boldsymbol{u}(\boldsymbol{x}_i,t)\cdot\nabla_{\boldsymbol{x}_i} \\
&\qquad\qquad + (\mathcal{L}\omega(\boldsymbol{x}_i,t) + F(\boldsymbol{x}_i,t))\,\frac{\partial}{\partial\omega_i}\Bigg]\,\tilde{f}_n(\boldsymbol{x}_1,\omega_1,...,\boldsymbol{x}_n,\omega_n,t)
\end{aligned} \tag{5.4}$$

In den Umformungen wurden dabei folgende Identitäten benutzt:

$$\frac{\partial}{\partial t}\delta(\omega_1-\omega(\boldsymbol{x}_1,t)) = -\left(\frac{\partial}{\partial t}\omega(\boldsymbol{x}_1,t)\right)\frac{\partial}{\partial\omega_1}\delta(\omega_1-\omega(\boldsymbol{x}_1,t)) \tag{5.5a}$$

$$\nabla_{\boldsymbol{x}_1}\delta(\omega_1-\omega(\boldsymbol{x}_1,t)) = -\left(\nabla_{\boldsymbol{x}_1}\omega(\boldsymbol{x}_1,t)\right)\frac{\partial}{\partial\omega_1}\delta(\omega_1-\omega(\boldsymbol{x}_1,t)) \tag{5.5b}$$

Für den nächsten Schritt benötigt man die Definition von bedingten Mittelwerten. Für eine beliebige Funktion $g(\omega)$ wäre dies

$$\begin{aligned}
\langle g(\omega(\boldsymbol{x},t))|\omega(\boldsymbol{x}_1,t)=\omega_1\rangle_\omega\, f_1(\omega_1,t) &= \int g(\omega(\boldsymbol{x},t))f_2(\omega,\boldsymbol{x},\omega_1,\boldsymbol{x}_1)\mathrm{d}\omega \\
&= \langle g(\omega(\boldsymbol{x},t))\delta(\omega_1-\omega(\boldsymbol{x}_1,t)\rangle_\omega
\end{aligned} \tag{5.6}$$

Nun führt man eine statistische Ensemblemittelung über alle möglichen Feldkonfigurationen durch. Das Ensemblemittel f_n von $\tilde{f}_n$ ist dabei gegeben durch

$$f_n(\boldsymbol{x}_1,\omega_1,...,\boldsymbol{x}_n,\omega_n,t) = \left\langle \tilde{f}_n(\boldsymbol{x}_1,\omega_1,...,\boldsymbol{x}_n,\omega_n,t)\right\rangle_\omega \tag{5.7}$$

Führt man damit nun die Ensemblemittelung durch und benutzt die Inkompressibilitätsbedingung $\nabla\cdot\boldsymbol{u}(\boldsymbol{x},t)=0$, so bekommt man

$$\begin{aligned}
&\left[\frac{\partial}{\partial t} + \sum_i \nabla_{\boldsymbol{x}_i}\cdot\langle\boldsymbol{u}(\boldsymbol{x}_i,t)|\omega(\boldsymbol{x}_1,t)=\omega_1,...,\omega(\boldsymbol{x}_n,t)=\omega_n\rangle_\omega\right] \\
&\quad f_n(\boldsymbol{x}_1,\omega_1,...,\boldsymbol{x}_n,\omega_n,t) \\
&\qquad = -\sum_i \frac{\partial}{\partial\omega_i}\,[(\langle\mathcal{L}\omega(\boldsymbol{x}_i,t)+F(\boldsymbol{x}_i,t)|\omega(\boldsymbol{x}_1,t)=\omega_1,...,\omega(\boldsymbol{x}_n,t)=\omega_n\rangle_\omega) \\
&\qquad f_n(\boldsymbol{x}_1,\omega_1,...,\boldsymbol{x}_n,\omega_n,t)]
\end{aligned} \tag{5.8}$$

In dieser Gleichung taucht das Schließungsproblem der Turbulenz auf, denn die bedingten Mittelwerte hängen jeweils von der n+1 Punkt Wahrscheinlichkeitsdichte f_{n+1} ab. Dieses lässt sich dann umgehen, wenn man diese bedingten Mittelwerte aus der Numerik bestimmt oder ein geeignetes Modell zur Beschreibung der Wahrscheinlichkeitsdichten findet. Letzteres soll die Methode sein, die im Folgenden weiter verfolgt wird.

5.2. Methode der Charakteristiken

Bevor nun diese bedingten Mittelwerte genauer analysiert werden, soll diese recht unhandliche partielle Differentialgleichung (5.8) durch die Methode der Charakteristiken in mehrere gewöhnliche Differentialgleichungen überführt werden. Dabei bestimmt man zunächst die totale Zeitableitung von f_n. Dabei geht man davon aus, dass sowohl die $\boldsymbol{x}_i$ als auch die ω_i vom Parameter t abhängen.

$$\frac{\mathrm{d}}{\mathrm{d}t} f_n(\boldsymbol{x}_1, \omega_1, ..., \boldsymbol{x}_n, \omega_n, t) = \left[\sum_i \frac{\mathrm{d}\boldsymbol{x}_i}{\mathrm{d}t} \cdot \nabla_{\boldsymbol{x}_i} + \sum_i \frac{\mathrm{d}\omega}{\mathrm{d}t} \frac{\partial}{\partial \omega_i} + \frac{\partial}{\partial t} \right] f_n(\boldsymbol{x}_1, \omega_1, ..., \boldsymbol{x}_n, \omega_n, t) \tag{5.9}$$

Vergleicht man die Koeffizienten mit Gleichung (5.8), so ergeben sich folgende gewöhnliche Differentialgleichungen:

$$\frac{\mathrm{d}\boldsymbol{x}_i}{\mathrm{d}t} = \langle \boldsymbol{u}(\boldsymbol{x}_i, t) | \omega(\boldsymbol{x}_1, t) = \omega_1, ..., \omega(\boldsymbol{x}_n, t) = \omega_n \rangle_\omega \tag{5.10a}$$

$$\frac{\mathrm{d}\omega_i}{\mathrm{d}t} = \langle \mathcal{L}\omega(\boldsymbol{x}_i, t) + F(\boldsymbol{x}_i, t) | \omega(\boldsymbol{x}_1, t) = \omega_1, ..., \omega(\boldsymbol{x}_n, t) = \omega_n \rangle_\omega \tag{5.10b}$$

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}t} f_n(\boldsymbol{x}_1, \omega_1, ..., \boldsymbol{x}_n, \omega_n, t) = \\ &\quad - \sum_i \Big[\nabla_{\boldsymbol{x}_i} \cdot \langle \boldsymbol{u}(\boldsymbol{x}_i, t) | \omega(\boldsymbol{x}_1, t) = \omega_1, ..., \omega(\boldsymbol{x}_n, t) = \omega_n \rangle_\omega \\ &\quad + \frac{\partial}{\partial \omega_i} \langle \mathcal{L}\omega(\boldsymbol{x}_i, t) + F(\boldsymbol{x}_i, t) | \omega(\boldsymbol{x}_1, t) = \omega_1, ..., \omega(\boldsymbol{x}_n, t) = \omega_n \rangle_\omega \Big] f_n(\boldsymbol{x}_1, \omega_1, ..., \boldsymbol{x}_n, \omega_n, t) \end{aligned} \tag{5.10c}$$

Dies ergibt Bewegungsgleichungen für die Punkte, auf die das Feld bedingt wird sowie weitere gewöhnliche Differentialgleichungen für deren Wirbelstärke und Verteilungsfunktion.

5.3. Numerische Bestimmung der bedingten Mittelwerte

Nun sollen zunächst die auf einen sowie auf zwei Punkte bedingten Mittelwerte des Wirbelstärkefeldes numerisch aus Simulationsdaten berechnet werden, um einen genaueren Überblick über deren Form zu gewinnen. Ein Beispiel hierfür findet sich unter anderem auch in [Fri+10].

5.3.1. 1-Punkt bedingte Mittelwerte

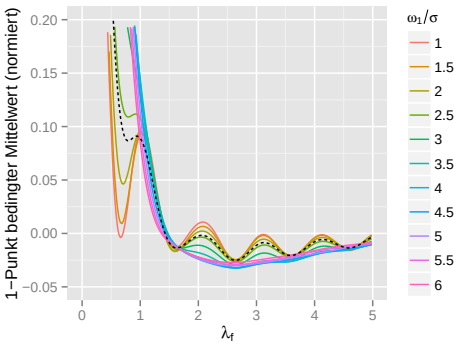
In Abbildungen 5.1 und 5.2 sind die auf einen Punkt bedingten Mittelwerte $\langle \omega(\boldsymbol{x}, t) | \omega(0, t) = \omega_1 \rangle_\omega$ zum Vergleich mit der 2-Punkt Wirbelstärkekorrelation $C(r)$ aufgetragen, jeweils radial gemittelt. Letztere wurde bereits in Kapitel 4.7 eingeführt. Es wurde bei der Berechnung des bedingten Mittelwerts für ω_1 jeweils eine Abweichung von $\pm 0,0005\omega_1$ zugelassen. Für die stochastische Kraft sieht man für fast alle ω_1 starke Abweichungen von $C(r)$. Je größer die Kraft ist, desto weniger stark ausgeprägt sind die Oszillationen. Gleiches gilt für große Wirbelstärken ω_1. Hier nehmen die Kurven eine glatte Form an. Für alle Kräfte ist hier der Wechsel von Oszillation zu glatter Form zu sehen.

Bei der deterministischen Kraft ist es so, dass die Oszillationen bei kleineren Kräften deutlich stärker sind. Hierbei weichen die Kurven kaum von $C(r)$ ab. Bei größeren Kraftstärken sieht man dahingegen ebenso den Übergang von oszillierenden Kurven zu glatten Kurven bei großen ω_1.

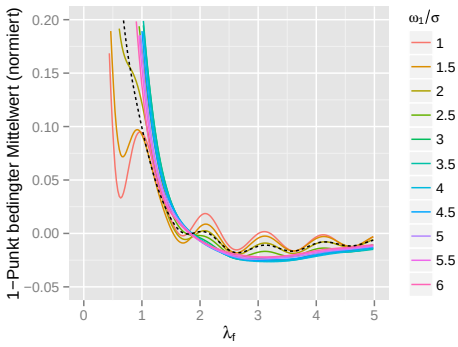
Eine mögliche Deutung des Ganzen ist, dass bei kleinen Wirbelstärken die stark oszillierende Kraft den bedingten Mittelwert dominiert. Je nach Krafttyp und Kraftamplitude befinden sich die größeren Wirbelstärken jenseits des Bereiches, der von der Kraft beeinflusst wird und stärker vom Wirbeltransport abhängt.

5.3.2. 2-Punkt bedingte Mittelwerte

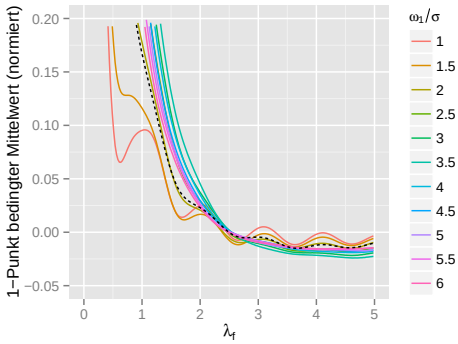
In Abbildung 5.3 sind zwei auf zwei Punkte gleicher Wirbelstärke bedingte Mittelwerte mit verschiedenen Abständen dargestellt. Für entfernte Punkte (z.B. $r = 4\lambda_f$) sieht man, dass die beiden Punkte sich nur kaum beeinflussen und die einzelnen Wirbelstärkefelder nahezu radialsymmetrisch sind. Schaut man sich näher benachbarte Punkte an, so ergibt sich hier eine Punktsymmetrie bezüglich des Mittelpunkts zwischen den beiden Wirbeln. Die Art dieser Symmetrie ist nicht verwunderlich, da zwei Wirbel dieser Art umeinander rotieren. Die Wirbel selbst nehmen eine Tropfenform an, die in erster Näherung eventuell durch eine um 45°nach links gekippte Ellipse beschrieben werden könnte. Die rechteckige Struktur in der Mitte entsteht dadurch, dass die beiden Punkte nicht unbedingt zu zwei verschiedenen Wirbel gehören müssen, sondern auch Teil einer einzelnen Wirbelstruktur sein können.



(a) stochastische Kraft, $f_A = 4\pi^2 \cdot 10^{-6}$



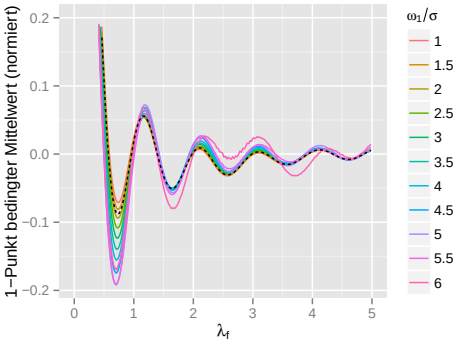
(b) stochastische Kraft, $f_A = 4\pi^2 \cdot 10^{-2}$



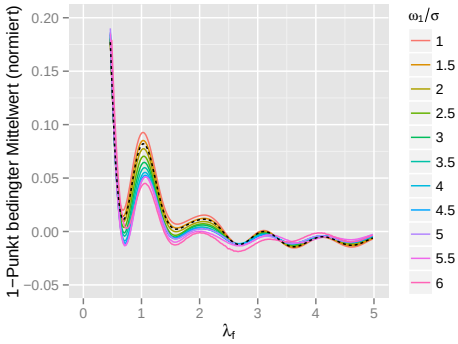
(c) stochastische Kraft, $f_A = 4\pi^2 \cdot 10^{2}$

Abbildung 5.1.: Gegenüberstellung des radial gemittelten bedingten Mittelwerts $\langle \omega(\boldsymbol{x}, t) | \omega(0, t) = \omega_1 \rangle_\omega$ (durchgezogen) zur 2-Punkt-Korrelation $C(r)$ (gestrichelt) bei verschiedenen Kraftparametern. (Simulation: `distcond`, siehe Anhang A)

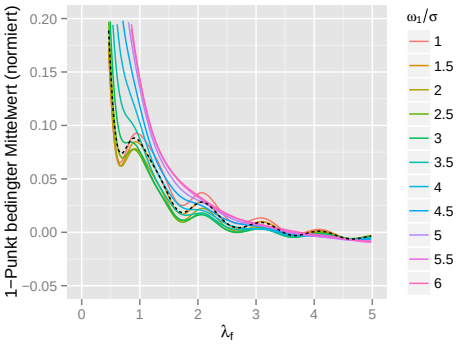
Für Wirbel unterschiedlichen Vorzeichens wie in Abbildung 5.4 ergibt sich dagegen für kleine Abstände eine Achsensymmetrie. Bei der stochastischen Kraft nehmen die Wirbel hier zunächst wieder eine Tropfenform an, die über Ellipsen genähert werden könnte. Bei größeren Abständen sind diese erst um ca. 45°geneigt und stellen sich für kleinere Abstände auf. Für die deterministische Kraft mit kleiner Amplitude ergibt sich keine merkbare Verformung, so dass ungefähr eine Achsensymmtrie entlang des Verbindungsvektors besteht.



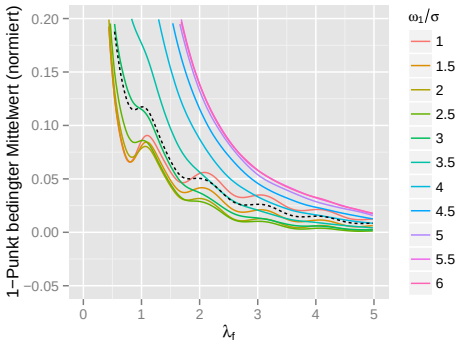
(a) deterministische Kraft, $f_A = 4\pi^2 \cdot 10^{-2}$



(b) deterministische Kraft, $f_A = 4\pi^2 \cdot 10^{-1}$

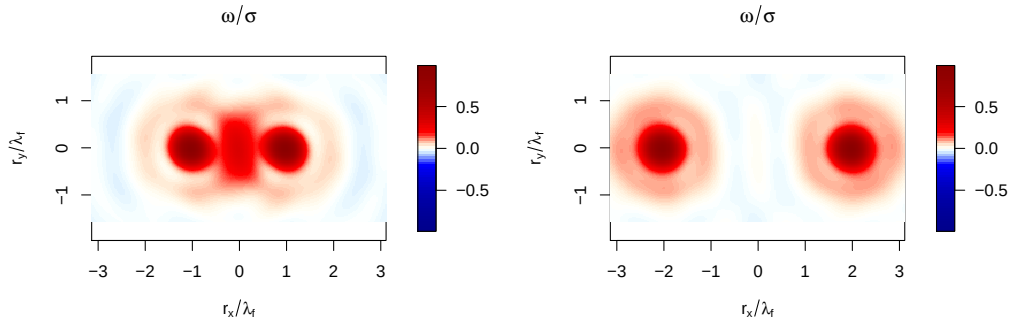


(c) deterministische Kraft, $f_A = 4\pi^2 \cdot 10^{0}$



(d) deterministische Kraft, $f_A = 4\pi^2 \cdot 10^{2}$

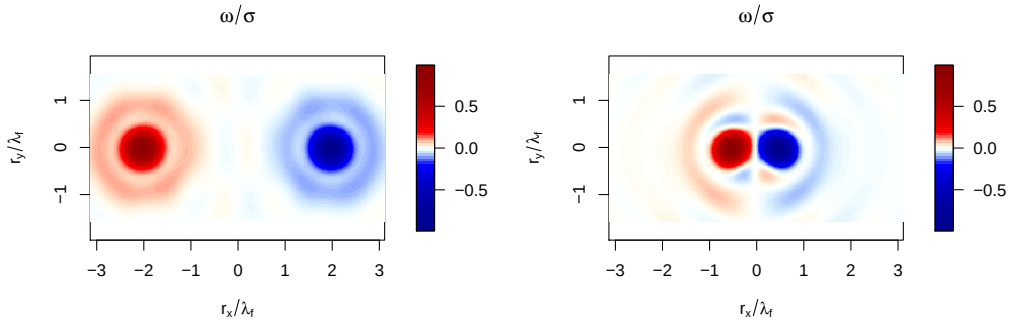
Abbildung 5.2.: Gegenüberstellung des radial gemittelten bedingten Mittelwerts $\langle \omega(\boldsymbol{x}, t) | \omega(0, t) = \omega_1 \rangle_\omega$ (durchgezogen) zur 2-Punkt-Korrelation $C(r)$ (gestrichelt) bei verschiedenen Kraftparametern. (Simulation: `4096inverse`, siehe Anhang A)



(a) stochastische Kraft, $f_A = 4\pi^2 \cdot 10^{-2}$, $r = 2\lambda_f$

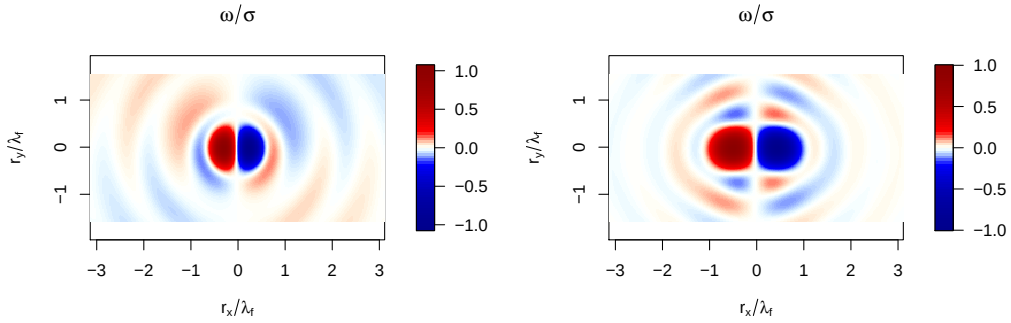
(b) stochastische Kraft, $f_A = 4\pi^2 \cdot 10^{-2}$, $r = 4\lambda_f$

Abbildung 5.3.: Darstellung auf zwei Punkte bedingter Mittelwerte $\langle \omega(\boldsymbol{x}, t) | \omega(\boldsymbol{x}_1, t) = \omega_1, \omega(\boldsymbol{x}_2, t) = \omega_2 \rangle_\omega$ mit $|\boldsymbol{x}_1 - \boldsymbol{x}_2| = r$ und $\omega_1 = \omega_2 = \sigma$. (Simulation: `4096inverse`, siehe Anhang A)



(a) stochastische Kraft, $f_A = 4\pi^2 \cdot 10^{-2}$, $r = 4\lambda_f$

(b) stochastische Kraft, $f_A = 4\pi^2 \cdot 10^{-2}$, $r = 1\lambda_f$



(c) stochastische Kraft, $f_A = 4\pi^2 \cdot 10^{-2}$, $r = 0, 4\lambda_f$

(d) deterministische Kraft, $f_A = 4\pi^2 \cdot 10^{-2}$, $r = 1\lambda_f$

Abbildung 5.4.: Darstellung auf zwei Punkte bedingter Mittelwerte $\langle \omega(\boldsymbol{x}, t) | \omega(\boldsymbol{x}_1, t) = \omega_1, \omega(\boldsymbol{x}_2, t) = \omega_2 \rangle_\omega$ mit $|\boldsymbol{x}_1 - \boldsymbol{x}_2| = r$ und $\omega_1 = -\omega_2 = \sigma$. (Simulation: `4096inverse`, siehe Anhang A)

5.4. Darstellung der bedingten Mittelwerte durch die charakteristische Funktion

In diesem kurzen Kapitel soll gezeigt werden, wie man die auf N Punkte bedingten Mittelwerte durch die charakteristische Funktion (siehe Kapitel 2.4) darstellt. Diese Herleitung findet sich auch in [Fri+10]. Sie ist Kernbestandteil der weiteren Auswertung, da hier beschrieben wird, wie mit den folgenden Modellen das Schließungsproblem behandelt wird.

Erst einmal überlegt man sich die Darstellung der N+1 Punkt Verteilung durch deren charakteristische Funktion im Fourierraum:

$$\begin{aligned} & f_{n+1}(\omega', \boldsymbol{x}', \omega_1, \boldsymbol{x}_1, ..., \omega_n, \boldsymbol{x}_n) \\ & = \frac{1}{(2\pi)^{n+1}} \int \mathrm{e}^{-i\alpha'\omega' - \sum_j i\alpha_j\omega_j} \varphi(\alpha', \boldsymbol{x}', \omega_1, \boldsymbol{x}_1, ..., \omega_n, \boldsymbol{x}_n) \mathrm{d}\alpha' \prod_j \mathrm{d}\alpha_j \\ & = \frac{1}{(2\pi)^{n+1}} \int \mathrm{e}^{-i\alpha'\omega' - \sum_j i\alpha_j\omega_j} \mathrm{e}^{-W_{n+1}(\alpha', \boldsymbol{x}', \omega_1, \boldsymbol{x}_1, ..., \omega_n, \boldsymbol{x}_n)} \mathrm{d}\alpha' \prod_j \mathrm{d}\alpha_j \end{aligned} \tag{5.11}$$

Der bedingte Mittelwert ist dann wie folgt über die Wahrscheinlichkeitsverteilung definiert:

$$\begin{aligned} & \left\langle \omega(\boldsymbol{x}') \middle| \omega(\boldsymbol{x}_1) = \omega_1, ..., \omega(\boldsymbol{x}_n) = \omega_n \right\rangle_\omega f_n(\omega_1, \boldsymbol{x}_1, ..., \omega_n, \boldsymbol{x}_n) \\ & = \int \omega' f_{n+1}(\omega', \boldsymbol{x}', \omega_1, \boldsymbol{x}_1, ..., \omega_n, \boldsymbol{x}_n) \mathrm{d}\omega' \\ & = \frac{1}{(2\pi)^{n+1}} \int \left[i \frac{\partial}{\partial \alpha'} \mathrm{e}^{-i\alpha'\omega' - \sum_j i\alpha_j\omega_j} \right] \mathrm{e}^{-W_{n+1}(\alpha', \boldsymbol{x}', \alpha_1, \boldsymbol{x}_1, ..., \alpha_n, \boldsymbol{x}_n)} \mathrm{d}\omega' \mathrm{d}\alpha' \prod_j \mathrm{d}\alpha_j \\ & = -\frac{1}{(2\pi)^{n+1}} \int \mathrm{e}^{-i\alpha'\omega' - \sum_j i\alpha_j\omega_j} \left[i \frac{\partial}{\partial \alpha'} \mathrm{e}^{-W_{n+1}(\alpha', \boldsymbol{x}', \alpha_1, \boldsymbol{x}_1, ..., \alpha_n, \boldsymbol{x}_n)} \right] \mathrm{d}\omega' \mathrm{d}\alpha' \prod_j \mathrm{d}\alpha_j \\ & = \frac{1}{(2\pi)^n} \int \mathrm{e}^{-\sum_j i\alpha_j\omega_j} \mathrm{e}^{-W_{n+1}(\alpha', \boldsymbol{x}', \alpha_1, \boldsymbol{x}_1, ..., \alpha_n, \boldsymbol{x}_n)} \\ & \left[i \frac{\partial}{\partial \alpha'} W_{n+1}(\alpha', \boldsymbol{x}', \alpha_1, \boldsymbol{x}_1, ..., \alpha_n, \boldsymbol{x}_n) \right] \delta(\alpha') \mathrm{d}\alpha' \prod_j \mathrm{d}\alpha_j \\ & = \frac{1}{(2\pi)^n} \int \mathrm{e}^{-\sum_j i\alpha_j\omega_j} \mathrm{e}^{-W_n(\alpha_1, \boldsymbol{x}_1, ..., \alpha_n, \boldsymbol{x}_n)} \\ & \left[i \frac{\partial}{\partial \alpha'} W_{n+1}(\alpha', \boldsymbol{x}', \alpha_1, \boldsymbol{x}_1, ..., \alpha_n, \boldsymbol{x}_n) \right]_{\alpha'=0} \prod_j \mathrm{d}\alpha_j \\ & = \left[i \frac{\partial}{\partial \alpha'} W_{n+1}(\alpha', \boldsymbol{x}', i\frac{\partial}{\partial \omega_1}, \boldsymbol{x}_1, ..., i\frac{\partial}{\partial \omega_n}, \boldsymbol{x}_n) \right]_{\alpha'=0} f_n(\omega_1, \boldsymbol{x}_1, ..., \omega_n, \boldsymbol{x}_n) \end{aligned} \tag{5.12}$$

Bei den Umformungen wurde zunächst die Definition des bedingten Mittelwerts benutzt, dann ω' durch eine Ableitung ersetzt, partiell integriert und die Ableitung mit Kettenregel ausgeführt. Da die Fouriertransformierte von f_{n+1} existiert, muss die charakteristische Funktion im Unendlichen verschwinden und somit fällt der zusätzliche Term der partiellen Integration weg. Anschließend wurde nach ω' integriert, wobei eine Delta-Distribution auftaucht. Nun wurde über α' ausintegriert und gleichzeitig berücksichtigt, dass $W_{n+1}(0, \boldsymbol{x}', \alpha_1, \boldsymbol{x}_1, ..., \alpha_n, \boldsymbol{x}_n) = W_n(\alpha_1, \boldsymbol{x}_1, ..., \alpha_n, \boldsymbol{x}_n)$ ist. Jedes Auftauchen von α_i wurde hierbei durch die Ableitung nach ω_i ersetzt.

Diese Gleichung, also kurz

$$\begin{aligned} &\left\langle \omega(\boldsymbol{x}') \middle| \omega(\boldsymbol{x}_1) = \omega_1, ..., \omega(\boldsymbol{x}_n) = \omega_n \right\rangle_\omega f_n(\omega_1, \boldsymbol{x}_1, ..., \omega_n, \boldsymbol{x}_n) \\ &= \left[i\frac{\partial}{\partial \alpha'} W_{n+1}(\alpha', \boldsymbol{x}', i\frac{\partial}{\partial \omega_1}, \boldsymbol{x}_1, ..., i\frac{\partial}{\partial \omega_n}, \boldsymbol{x}_n) \right]_{\alpha'=0} f_n(\omega_1, \boldsymbol{x}_1, ..., \omega_n, \boldsymbol{x}_n) \end{aligned} \tag{5.13}$$

ermöglicht es nun, als Modell den Exponenten der charakteristischen Funktion W_{n+1} vorzugeben und damit die bedingten Mittelwerte zu errechnen. In den nächsten Kapiteln soll es darum gehen, ein geeignetes Modell für die Wahrscheinlichkeitsdichte zu finden, über welche die bedingten Mittelwerte beschrieben werden können.

5.5. Wahrscheinlichkeitsverteilungen und charakteristische Funktion

Zur Modelierung der Wahrscheinlichkeitsdichte und der charakteristischen Funktion muss man einige Regeln beachten, die aus dem Prinzip der homogenen isotropen Turbulenz folgen (vgl. auch [Lun67]). Diese Regeln werden hier an der 2-Punkt-Verteilungsfunktion verdeutlicht, gelten aber für beliebige N-Punkt-Verteilungsfunktionen:

- **Homogenität**: Aus der Homogenität folgt, dass die Wahrscheinlichkeitsdichte nicht vom Ort abhängen darf, sondern nur von den Abständen der Punkte.

$$f_2(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2) = f_2(\omega_1, \omega_2, \boldsymbol{x}_1 - \boldsymbol{x}_2) \tag{5.14}$$

- **Isotropie**: Aus der Isotropie folgt, dass die Wahrscheinlichkeitsdichte invariant unter Rotationen sein muss. Nimmt man Isotropie und Homogenität zusammen, erhält man

$$\begin{aligned} f_2(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2) &= f_2(\omega_1, \omega_2, |\boldsymbol{x}_1 - \boldsymbol{x}_2|) \\ &= f_2(\omega_1, \omega_2, r) \end{aligned} \tag{5.15}$$

- **Gleiche Punkte**: Für zwei gleiche Punkte muss die Wahrscheinlichkeitsdichte in eine Delta-Distribution der Wirbelstärken übergehen.

$$\lim_{\boldsymbol{x}_1 \to \boldsymbol{x}_2} f_2(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2) = f_1(\omega_1, \boldsymbol{x}_1)\delta(\omega_1 - \omega_2) \tag{5.16}$$

- **Weit entfernte Punkte**: Zwei weit entfernte Punkte müssen statistisch unabhängig voneinander sein.

$$\lim_{r \to \infty} f_2(\omega_1, \omega_2, r) = f_1(\omega_1) f_1(\omega_2) \tag{5.17}$$

Im Folgenden werden die aus den Simulationen gewonnenen 1-Punkt und 2-Punkt Verteilungen dargestellt, um ein Gefühl für deren Form zu bekommen.

5.5.1. 1-Punkt Verteilungen

Die 1-Punkt Verteilungen wurden in Kapitel 4.6 schon einmal dargestellt und analysiert. Hier soll deren Form noch einmal genau betrachtet werden.

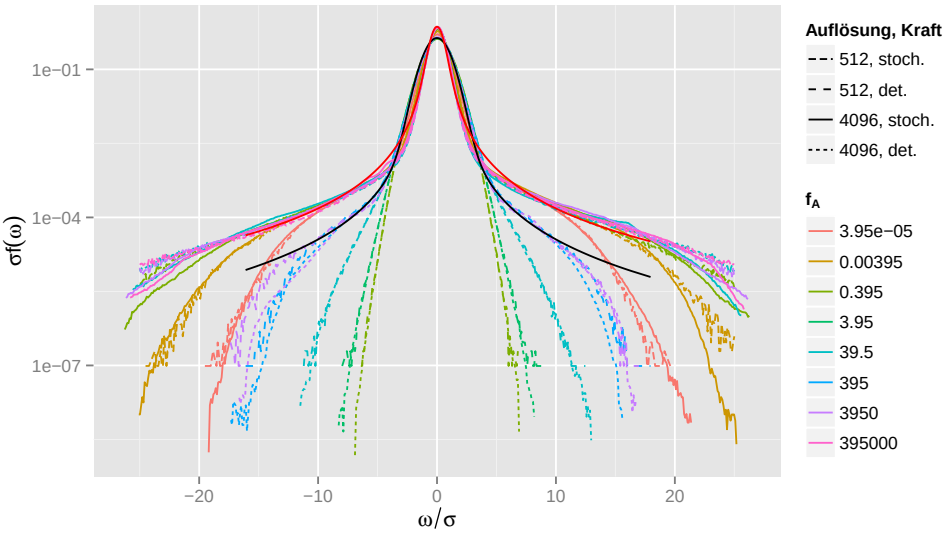


Abbildung 5.5.: Wahrscheinlichkeitsdichte $f_1(\omega)$ berechnet bei verschiedenen Kräften und Auflösungen. Als schwarze und rote Linie sind zwei stabile Verteilungen $f_s(\omega/\sigma)$ eingezeichnet mit den Parametern $\beta = 1,93$, $\gamma = 0,44$ für die schwarze Linie und $\beta = 1,6$, $\gamma = 0,22$ für die rote. (Simulation: `resforce`, siehe Anhang A)

Wie man in Abbildung 5.5 sieht, nehmen die Verteilungsfunktionen für kleine deterministische Kräfte nahezu eine Gaußverteilung an. Höhere Kräfte verursachen weite

Flügel. Charakteristisch für die eingezeichneten stabilen Verteilungen ist, dass die Flügel nicht wieder nach unten abknicken, was aber bei den vorliegenden Verteilungen deutlich der Fall ist. Dies muss mit der Viskosität zusammenhängen, die dafür sorgt, dass Wirbelstärken nicht unendlich anwachsen können. Desweiteren sieht man bei der roten Kurve, also der Näherung durch eine stabile Verteilung, dass diese in der Nähe des Knicks eine größere Abweichung von der echten Verteilung ausweist.

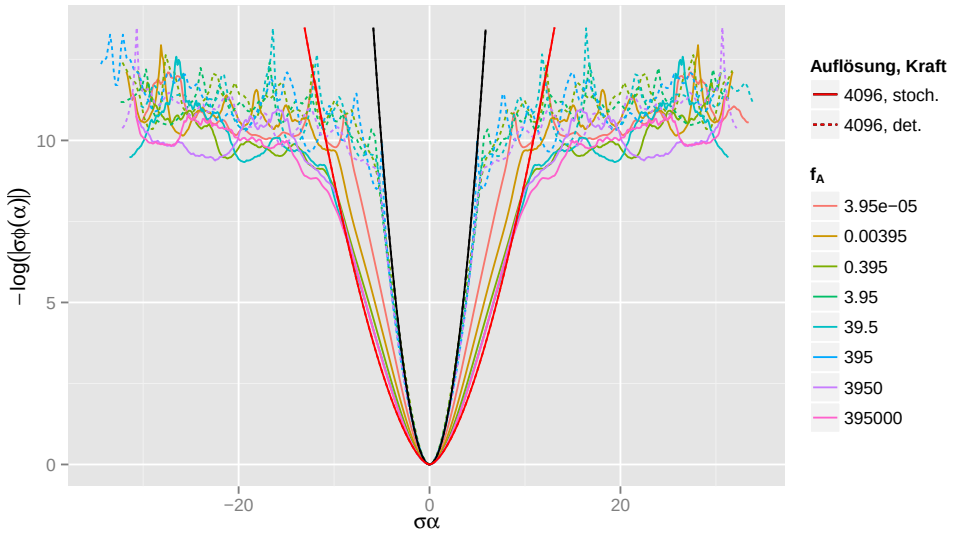


Abbildung 5.6.: Exponent der charakteristischen Funktion $\varphi(\alpha)$ berechnet bei verschiedenen Kräften. Die Parameter der eingezeichneten roten und schwarzen Kurve entsprechen denen aus Abbildung 5.5. (Simulation: `resforce`, siehe Anhang A)

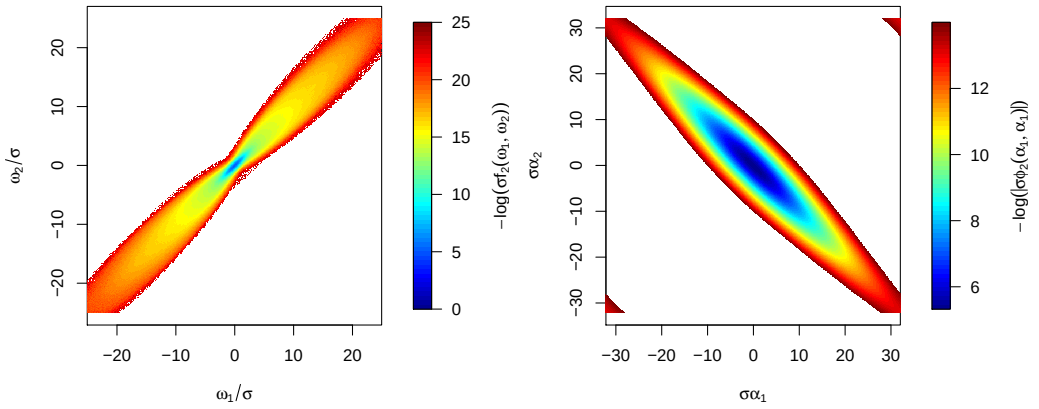
Abbildung 5.5 zeigt die dazugehörigen Exponenten der charakteristischen Funktion. Sie sind die Größen, die man gerne modellieren möchte und aus denen man die bedingten Mittelwerte errechnen kann. Die scheinbar chaotischen „Flügel“ bei $W \approx 10$ stammen vermutlich aus numerischen Ungenauigkeiten der Wahrscheinlichkeitsverteilung an den Rändern, die sich schlecht mit der fouriertransformieren lassen. Eine Rücktransformation einer „ordentlichen“ chrarakteristischen Funktion ergibt einen ähnlichen Effekt im Realraum an den Seiten. Man sieht auch hier wieder, dass die stabilen Funktionen zur Näherung einigermaßen passen, aber nicht komplett. Zum Beispiel ist W bei $\sigma\alpha = 0$ sehr spitz und steigt ab spätestens $2\sigma\alpha$ nahezu geradlinig an.

5.5.2. 2-Punkt Verteilungen

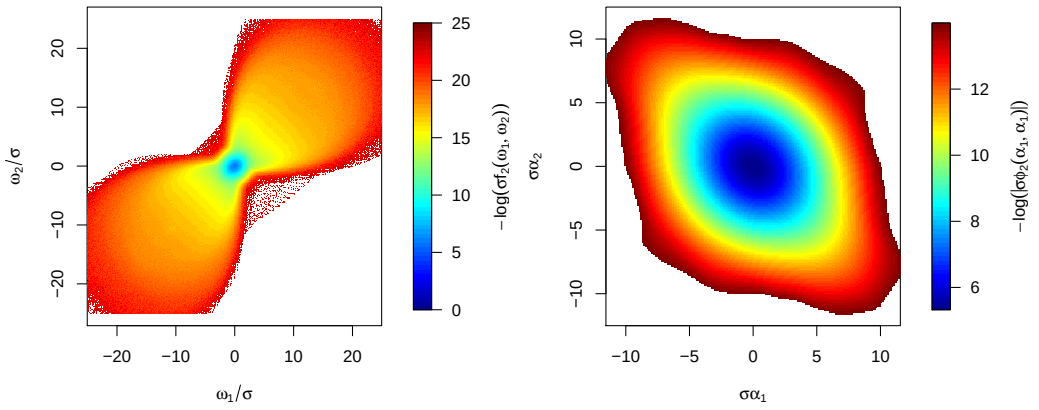
In Abbildungen 5.7 und 5.8 wird eine ganze Serie von Verteilungen und den Exponenten ihrer charakteristischen Funktionen dargestellt. Letztere sind dabei um die störenden Ränder bereinigt worden, um die innere Struktur besser erkennen zu können. Die manchmal etwas zackige Randstruktur hat vermutlich keine größere Bedeutung, da sie sehr nah am Übergang zu den Fluktuationen ist, die schon in Abbildung 5.6 beobachtet wurden.

Bei den Verteilungsfunktionen sieht man beim kleinsten gewählten Abstand den Übergang zur Delta-Distribution in ω_1 und ω_2. Der Exponent der charakteristischen Funktion entspricht hier einer schmalen Ellipse. Vergrößert man den Abstand, so weitet sich die Verteilung auf und W_2 nimmt eine gestauchte Form an, die etwas spitzer als eine Ellipse ist. Im äußeren Bereich nähert sich die Form fast einer flach liegenden Raute. Vergrößert man den Abstand weiter, so bildet sich bei der Verteilung langsam eine Sternform heraus. Dies ist nicht verwunderlich, da diese einer Multiplikation zweier unabhängiger stabiler Verteilungen in ω_1 bzw. ω_2 entspricht. W_2 bekommt hier nahezu die Form eines Kreises. Für sehr große Abstände verstärkt sich die Sternform von f_2 noch weiter und W_2 erhält die Form einer stehenden Raute. Eine exakte stehende Raute entspräche $W_2 \propto |x|+|y|$.

In den nächsten Kapiteln geht es nun darum, diesen Exponenten der charakteristischen Funktion zu modellieren bzw. diese Modelle zu parametrisieren und dadurch Vorhersagen über die bedingten Mittelwerte zu machen.

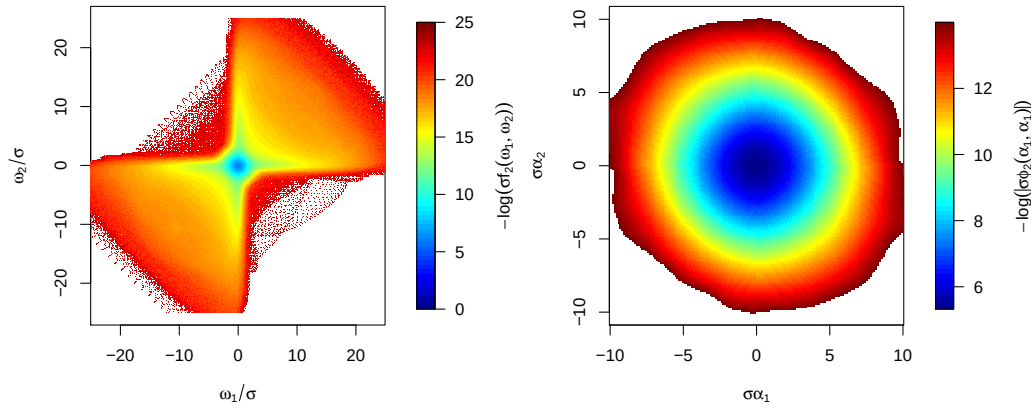


(a) $f_2(\omega_1/\sigma, \omega_2/\sigma, r = 0, 1\lambda_f)$ **(b)** $W_2(\sigma\alpha_1, \sigma\alpha_2, r = 0, 1\lambda_f)$

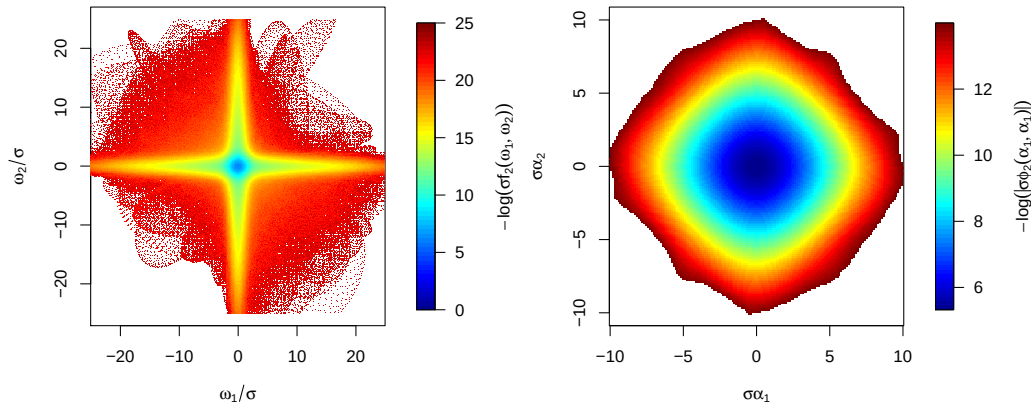


(c) $f_2(\omega_1/\sigma, \omega_2/\sigma, r = 0, 4\lambda_f)$ **(d)** $W_2(\sigma\alpha_1, \sigma\alpha_2, r = 0, 4\lambda_f)$

Abbildung 5.7.: 2-Punkt Verteilungen $f_2(\frac{\omega_1}{\sigma}, \boldsymbol{x}_1, \frac{\omega_2}{\sigma}, \boldsymbol{x}_2)$ und der Exponent ihrer zugehörigen charakteristischen Funktion $W_2(\sigma\alpha_1, \boldsymbol{x}_1, \sigma\alpha_2, \boldsymbol{x}_2)$ in verschiedenen Abständen. (Simulation: `distcond`, siehe Anhang A)



(a) $f_2(\omega_1/\sigma, \omega_2/\sigma, r = 0,7\lambda_f)$ **(b)** $W_2(\sigma\alpha_1, \sigma\alpha_2, r = 0,7\lambda_f)$



(c) $f_2(\omega_1/\sigma, \omega_2/\sigma, r = 4\lambda_f)$ **(d)** $W_2(\sigma\alpha_1, \sigma\alpha_2, r = 4\lambda_f)$

Abbildung 5.8.: 2-Punkt Verteilungen $f_2(\frac{\omega_1}{\sigma}, \boldsymbol{x}_1, \frac{\omega_2}{\sigma}, \boldsymbol{x}_2)$ und der Exponent ihrer zugehörigen charakteristischen Funktion $W_2(\sigma\alpha_1, \boldsymbol{x}_1, \sigma\alpha_2, \boldsymbol{x}_2)$ in verschiedenen Abständen. (Simulation: `distcond`, siehe Anhang A)

5.6. Allgemeiner Polynomansatz für die Wahrscheinlichkeitsdichte

Als erste Näherung für den Exponenten der charakteristischen Funktion könnte man ein allgemeines Polynom einsetzen.

$$W_n(\alpha_1, \boldsymbol{x}_1, ..., \alpha_n, \boldsymbol{x}_n) = \sum_{m_1,...,m_n=0}^{\infty} \alpha_1^{m_1}\alpha_2^{m_2} \cdot ... \cdot \alpha_n^{m_n} C^n_{m_1,...,m_n}(\boldsymbol{x}_1, ..., \boldsymbol{x}_n) \quad (5.18)$$

Dieser Ansatz ist allerdings problematisch, da für Exponenten $\alpha > 2$ die Wahrscheinlichkeitsverteilung an manchen Stellen negativ sein kann (siehe [Luk60]). Dies kann nur im Falle eines unendlich langen Polynoms umgangen werden. Dieses Problem könnte eventuell durch geeignete Wahl von Koeffizienten C^n minimiert werden, sodass man trotzdem eine recht gute Approximation einer Wahrscheinlichkeitsdichte erhalten könnte. Nehmen wir erst einmal den Fall an, dass so eine Näherung möglich ist, so stellen sich bestimmte Bedingungen an die Koeffizienten-Funktionen $C^n_{m_1,...,m_n}$.

Damit die Wahrscheinlichkeitsverteilung reell ist, muss gelten $W(-\boldsymbol{\alpha}) = W(\boldsymbol{\alpha})^*$. Somit ergibt sich als Bedingung an die C^n:

$$\mathrm{Re}(C^n_{m_1,...,m_n}) = 0, \quad \text{für } \sum_{i=1}^{n} m_i \text{ ungerade} \quad (5.19a)$$

$$\mathrm{Im}(C^n_{m_1,...,m_n}) = 0, \quad \text{für } \sum_{i=1}^{n} m_i \text{ gerade} \quad (5.19b)$$

Für den Fall, dass die Wahrscheinlichkeitsverteilung zusätzlich symmetrisch in $\boldsymbol{\omega}$ sein soll, müsste $W(\boldsymbol{\alpha})$ rein reell sein und damit auch die $C^n_{m_1,...,m_n}$.

Außerdem muss für jede Randverteilung gelten

$$\begin{aligned}
&\int f_n(\omega_1, \boldsymbol{x}_1, ..., \omega_n, \boldsymbol{x}_n)\mathrm{d}\omega_n \\
&= \frac{1}{(2\pi)^n} \int \mathrm{e}^{-W_n(\alpha_1,\boldsymbol{x}_1,...,\alpha_n,\boldsymbol{x}_n)} \mathrm{e}^{-i\sum_{i=1}^{n}\alpha_i\omega_i} \mathrm{d}\alpha_1 ... \mathrm{d}\alpha_n \mathrm{d}\omega_n \\
&= \frac{1}{(2\pi)^{n-1}} \int \left[\mathrm{e}^{-W_n(\alpha_1,\boldsymbol{x}_1,...,\alpha_n,\boldsymbol{x}_n)}\right]_{\alpha_n=0} \mathrm{e}^{-i\sum_{i=1}^{n-1}\alpha_i\omega_i} \mathrm{d}\alpha_1 ... \mathrm{d}\alpha_{n-1} \\
&\overset{!}{=} \frac{1}{(2\pi)^{n-1}} \int \mathrm{e}^{-W_{n-1}(\alpha_1,\boldsymbol{x}_1,...,\alpha_{n-1},\boldsymbol{x}_{n-1})} \mathrm{e}^{-i\sum_{i=1}^{n-1}\alpha_i\omega_i} \mathrm{d}\alpha_1 ... \mathrm{d}\alpha_{n-1} \\
&= f_{n-1}(\omega_1, \boldsymbol{x}_1, ..., \omega_{n-1}, \boldsymbol{x}_{n-1})
\end{aligned} \quad (5.20)$$

Hier sieht man sofort, dass die $C^n_{m_1,\ldots,m_{n-1},0}(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n)$ nicht mehr von $\boldsymbol{x}_n$ abhängen können. Da außerdem über jedes ω_i ausintegriert werden kann, muss $C^n_{m_1,\ldots,m_{n-1},0} = C^n_{m_1,\ldots,0,m_n} = \ldots = C^n_{0,m_2,\ldots,m_n}$ gelten. Sie sind durch $C^n_{m_1,\ldots,m_{n-1},0} = C^{n-1}_{m_1,\ldots,m_{n-1}}$ über ihre Ordnungen n hinweg verknüpft.

Aus Gründen der Homogenität hängt die Funktion C außerdem nur von den Abständen $\boldsymbol{x}_i - \boldsymbol{x}_j$ zueinander ab.

Vertauscht man $(\omega_i, \boldsymbol{x}_i)$ mit $(\omega_j, \boldsymbol{x}_j)$, so ändert sich die Wahrscheinlichkeitsdichte nicht. Somit muss auch C^n unter Vertauschung von $(m_i, \boldsymbol{x}_i)$ und $(m_j, \boldsymbol{x}_j)$ identisch sein.

Schaut man sich nun den Fall $n = 2$ an, so erhält man für den bedingten Mittelwert

$$\begin{aligned}\int \omega f_2(\omega, \boldsymbol{x}, \omega_1, \boldsymbol{x}_1)\mathrm{d}\omega &= \langle \omega(\boldsymbol{x}) | \omega(\boldsymbol{x}_1) = \omega_1 \rangle \, f_1(\omega_1, \boldsymbol{x}_1) \\ &= i \int \left[\frac{\partial}{\partial \alpha} W_2(\alpha, \boldsymbol{x}, \alpha_1, \boldsymbol{x}_1) \right]_{\alpha=0} \mathrm{e}^{-W_1(\alpha_1, \boldsymbol{x}_1)} \mathrm{e}^{-i\omega_1\alpha_1} \mathrm{d}\alpha_1 \\ &= i \int \sum_n \alpha_1^n C^2_{1n}(\boldsymbol{x}, \boldsymbol{x}_1) \mathrm{e}^{-W_1(\alpha_1, \boldsymbol{x}_1)} \mathrm{e}^{-i\omega_1\alpha_1} \mathrm{d}\alpha_1 \\ &= i \sum_n C^2_{1n}(\boldsymbol{x}, \boldsymbol{x}_1) \left(i \frac{\partial}{\partial \omega_1} \right)^n f_1(\omega_1, \boldsymbol{x}_1)\end{aligned} \tag{5.21}$$

unter dem allgemein gehaltenen Fall, dass

$$W_1(\alpha_1, \boldsymbol{x}_1) = \sum_n \alpha^n C^1_n(\boldsymbol{x}_1) \tag{5.22a}$$

$$W_2(\alpha, \boldsymbol{x}, \alpha_1, \boldsymbol{x}_1) = \sum_{n,m} \alpha^n \alpha_1^m C^2_{nm}(\boldsymbol{x}, \boldsymbol{x}_1) \tag{5.22b}$$

mit $C^2_{0n}(\boldsymbol{x}, \boldsymbol{x}_1) = C^2_{n0}(\boldsymbol{x}_1, \boldsymbol{x}) = C^1_n(\boldsymbol{x}_1) = C^1_n$. Auf Grund der Homogenität muss dies eine Konstante sein. Somit sind auch alle $C^n_{m_1,\ldots,m_n}$ konstant, falls ein $m_i = 0$ ist.

Außerdem gilt aus Symmetriegründen $f_1(\omega, \boldsymbol{x}) = f_1(-\omega, \boldsymbol{x})$ sowie $f_2(\omega, \boldsymbol{x}, \omega_1, \boldsymbol{x}_1) = f_2(-\omega, \boldsymbol{x}, -\omega_1, \boldsymbol{x}_1)$. Dies führt darauf, dass alle C^1_n und $C^2_{nm}(\boldsymbol{x} - \boldsymbol{x}_1)$ reell sind. Das heißt für ungerade m ist $C^1_n = 0$ und für ungerade $n + m$ ist $C^2_{nm} = 0$. Isotropie sorgt dafür, dass $C^2_{nm}(\boldsymbol{x} - \boldsymbol{x}_1) = C^2_{nm}(|\boldsymbol{x} - \boldsymbol{x}_1|)$ nur vom Betrag des Abstandes der beiden Punkte zueinander abhängt.

Auf diese Art und Weise hat man nun durch einen allgemeinen Polynomansatz eine nichtlineare Abhängigkeit von ω_1 für den bedingten Mittelwert in Gleichung (5.21)

erhalten, wie sie später in Kapitel 5.3.1 beobachtet wird. Allerdings zeigt sich so noch keine einfache Möglichkeit, die Koeffizientenfunktionen C^n zu bestimmen. Ein direktes Weiterrechnen mit dieser Gleichung erfordert sukzessives Ableiten der charakteristischen Funktion im Fourierraum und führt auf komplexe Polynome.

5.7. Näherung mit gaußscher Verteilung

Erlaubt man in dem Polynomansatz aus Kapitel 5.6 nur maximal $m_i = 2$, so erhält man die gaußsche Näherung, wie sie auch schon in [Fri+10] benutzt wurde. Durch die gewählte Beschränkung sind automatisch alle Koeffizientenfunktionen C^n reell. Die Form des Exponenten entspricht nun einer normalen multivariaten Gaußverteilung.

$$\begin{aligned} &W_{n+1}(\alpha', \boldsymbol{x}', \alpha_1, \boldsymbol{x}_1, ..., \alpha_n, \boldsymbol{x}_n) \\ &= \frac{1}{2}\alpha' C(0)\alpha' + \sum_{i,j} \frac{1}{2}\alpha_i C(\boldsymbol{x}_i - \boldsymbol{x}_j)\alpha_j + \sum_i \alpha' C(\boldsymbol{x}' - \boldsymbol{x}_i)\alpha_i \end{aligned} \tag{5.23}$$

In Analogie zur Normalverteilung wurde hier der Faktor $\frac{1}{2}$ aus den Koeffizienten explizit herausgezogen. In Matrixschreibweise erkennt man die gaußsche Natur dieser charakteristischen Funktion, denn sie ist die Fouriertransformierte einer elliptisch gaußschen Verteilung:

$$\mathcal{F}\left[f(\omega_1, \boldsymbol{x}_1, ..., \omega_n, \boldsymbol{x}_n)\right] = \mathcal{F}\left[\frac{1}{(\sqrt{2\pi})^n\sqrt{\det C}} \mathrm{e}^{-\frac{1}{2}\boldsymbol{\omega} C^{-1} \boldsymbol{\omega}}\right] = \frac{1}{(\sqrt{2\pi})^n} \mathrm{e}^{-\frac{1}{2}\boldsymbol{\alpha} C \boldsymbol{\alpha}} \tag{5.24}$$

Dabei ist Matrix C^{-1} die Inverse der Matrix $C_{ij} = C(\boldsymbol{x}_i - \boldsymbol{x}_j)$.

Setzt man sowohl W_{n+1} als auch die gaußsche Wahrscheinlichkeitsdichte in Gleichung (5.13) ein, so erhält man

$$\begin{aligned} &\left\langle \omega(\boldsymbol{x}') \middle| \omega(\boldsymbol{x}_1) = \omega_1, ..., \omega(\boldsymbol{x}_n) = \omega_n \right\rangle_\omega f_n(\omega_1, \boldsymbol{x}_1, ..., \omega_n, \boldsymbol{x}_n) \\ &= \left[i\frac{\partial}{\partial \alpha'} W_{n+1}(\alpha', \boldsymbol{x}', i\frac{\partial}{\partial \omega_1}, \boldsymbol{x}_1, ..., i\frac{\partial}{\partial \omega_n}, \boldsymbol{x}_n) \right]_{\alpha'=0} f_n(\omega_1, \boldsymbol{x}_1, ..., \omega_n, \boldsymbol{x}_n) \\ &= i\left[\alpha' C(0) + \sum_i C(\boldsymbol{x}' - \boldsymbol{x}_i) i\frac{\partial}{\partial \omega_i} \right]_{\alpha'=0} f_n(\omega_1, \boldsymbol{x}_1, ..., \omega_n, \boldsymbol{x}_n) \\ &= \left[-\sum_i C(\boldsymbol{x}' - \boldsymbol{x}_i)\frac{\partial}{\partial \omega_i} \right] f_n(\omega_1, \boldsymbol{x}_1, ..., \omega_n, \boldsymbol{x}_n) \end{aligned} \tag{5.25}$$

Nach dieser Umformung muss man nur noch die Ableitungen auf die Wahrscheinlichkeitsdichte ausführen, wie sie in Gleichung (5.24) steht und erhält

$$\left\langle \omega(\boldsymbol{x}') \middle| \omega(\boldsymbol{x}_1) = \omega_1, ..., \omega(\boldsymbol{x}_n) = \omega_n \right\rangle_\omega = \sum_{im} C(\boldsymbol{x}' - \boldsymbol{x}_i) C^{-1}_{im} \omega_m \tag{5.26}$$

Diese Formel wird als linear gaußsche Näherung bezeichnet, da sie sowohl in ω_m als auch in $C(\boldsymbol{x}' - \boldsymbol{x}_i)$ linear ist.

5.8. Vergleich: Näherung durch gaußsche Verteilung mit Numerik

In Kapitel 5.5 wurde gezeigt, dass bei Verwendung einer deterministischen Kraft und kleinen Amplituden die Wahrscheinlichkeitsdichte der Wirbelstärke in erster Näherung einer gaußschen Verteilung ähnelt. Für größere Kraftamplituden verändert sich der Exponent β der Verteilungsfunktion, sodass die Verteilung immer stärkere Flügel bekommt. In diesem Kapitel soll nun untersucht werden, wie gut die Näherungen sind bzw. wann Abweichungen auftreten und wie diese aussehen.

5.8.1. 2-Punkt Wirbelstärkekorrelation

Die in Kapitel 4.7 vorgestellte 2-Punkt-Wirbelstärkekorrelation lässt sich auch durch bedingte Mittelwerte ausdrücken.

$$\begin{aligned} \langle \omega(\boldsymbol{x}) \omega(\boldsymbol{x} + \boldsymbol{r}) \rangle_\omega &= \iint \omega_1 \omega_2 f_2(\omega_1, \boldsymbol{x}, \omega_2, \boldsymbol{x} + \boldsymbol{r}) \mathrm{d}\omega_1 \mathrm{d}\omega_2 \\ &= \int \omega_1 \left\langle \omega(\boldsymbol{x} + \boldsymbol{r}) | \omega(\boldsymbol{x}) = \omega_1 \right\rangle_\omega f(\omega_1) \mathrm{d}\omega_1 \end{aligned} \tag{5.27}$$

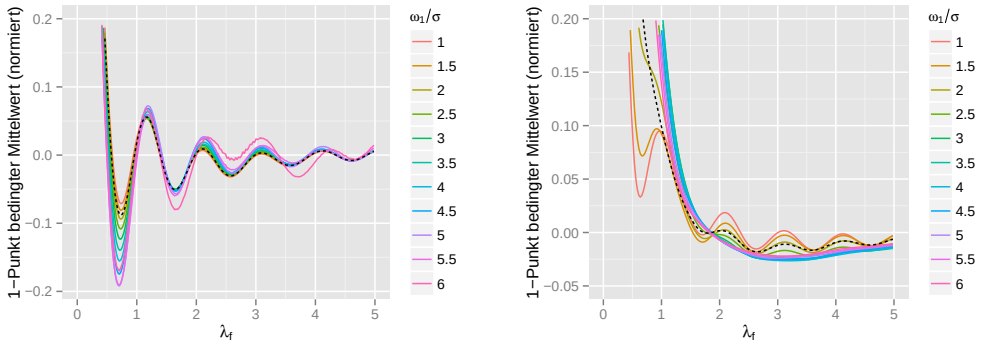
Unter Annahme einer gaußschen Verteilung und mit Hilfe von Gleichung (5.39) folgt dann weiter

$$\langle \omega(\boldsymbol{x}) \omega(\boldsymbol{x} + \boldsymbol{r}) \rangle_\omega = \int \omega_1 \omega_1 \frac{C(\boldsymbol{r})}{C(0)} f(\omega_1) \mathrm{d}\omega_1 = \frac{C(\boldsymbol{r})}{C(0)} \left\langle \omega^2 \right\rangle_\omega \tag{5.28}$$

Dies gibt einem eine einfache Möglichkeit, die Funktion $C(\boldsymbol{r})$ zu berechnen, welche die Kovarianzmatrix der Gaußverteilung bestimmt.

5.8.2. 1-Punkt bedingter Mittelwert

In Kapitel 5.3.1 in Abbildungen 5.1 und 5.2 wurden bereits die auf einen Punkt bedingten Mittelwerte gegen die 2-Punkt Wirbelstärkekorrelation $C(r)$ radial gemittelt aufgetragen. Als Folgerung aus der der gaußschen Näherung (Gleichung (5.26)) sollten die bedingten Mittelwerte forminvariant sein und sich nur in ihrer Amplitude von der 2-Punkt Wirbelstärkekorrelationsfunktion $C(\boldsymbol{r})$ unterscheiden. Hier in Abbildung 5.9 seien noch einmal zur Übersicht die beiden Extremfälle gegenübergestellt.



(a) deterministische Kraft, $f_A = 4\pi^2 \cdot 10^{-2}$ **(b)** stochastische Kraft, $f_A = 4\pi^2 \cdot 10^{-2}$

Abbildung 5.9.: Gegenüberstellung des radial gemittelten bedingten Mittelwerts $\langle \omega(\boldsymbol{x}, t) | \omega(0, t) = \omega_1 \rangle_\omega$ (durchgezogen) zur 2-Punkt-Korrelation $C(r)$ (gestrichelt) bei verschiedenen Kraftparametern. (Simulation: `distcond`, siehe Anhang A)

Im Falle kleiner Kraftstärken und einer deterministischen Kraft entspricht die Wahrscheinlichkeitsdichte $f_1(\omega)$ fast einer gaußschen Verteilung und somit ist die Form für unterschiedliche ω_1 nahezu identisch $C(r)$. Größere Abweichungen treten nur beim ersten Minimum auf. Die Abweichung im Fall $\omega_1 = 6\sigma$ ist darauf zurückzuführen, dass für diese Kurve nur über wenige Felder gemittelt wurde (72 in diesem Fall).

Bei größeren Kraftstärken oder der stochastischen Kraft sieht man größere Abweichungen von $C(r)$. Für diese Parameter besitzt auch die Wahrscheinlichkeitsverteilung keine gaußsche Form, bzw. die Abweichungen von der gaußschen Form werden in diesen Bereichen von ω_1 stärker, wie man in Abbildung 4.9 sieht.

Für höhere Kraftstärken und jeweils höhere ω_1 werden die Oszillationen schwächer bzw. verschwinden komplett. Dies deutet auf einen immer schwächeren Einfluss der Kraft hin, da diese für die räumlichen Oszillationen verantwortlich ist.

5.8.3. 2-Punkt bedingter Mittelwert

Nun soll untersucht werden, inwieweit der auf zwei Punkte bedingte Mittelwert von der gaußschen Näherung abweicht. Unter Annahme einer multivariat gaußschen Verteilung und der daraus resultierenden Näherung (Gleichung (5.26)) ergibt sich für diesen

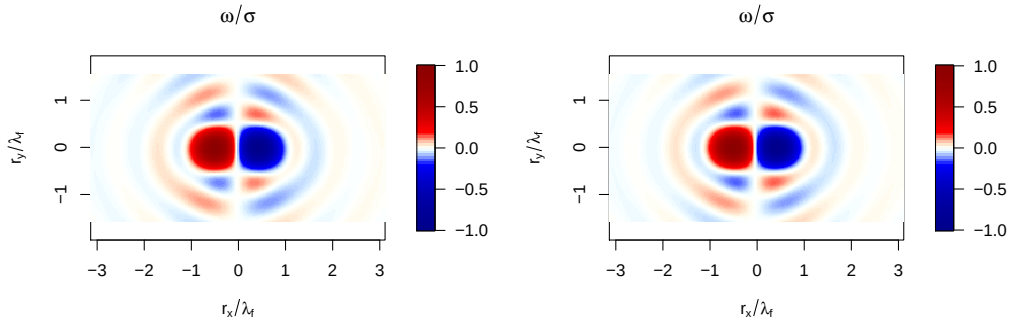
$$\begin{aligned}&\langle\omega(\boldsymbol{x})|\omega(\boldsymbol{x}_1)=\omega_1,\omega(\boldsymbol{x}_2)=\omega_2\rangle\\&=C(\boldsymbol{x}-\boldsymbol{x}_1)\frac{C(0)\omega_1-C(\boldsymbol{x}_1-\boldsymbol{x}_2)\omega_2}{C(0)^2-C(\boldsymbol{x}_1-\boldsymbol{x}_2)^2}\\&+C(\boldsymbol{x}-\boldsymbol{x}_2)\frac{C(0)\omega_2-C(\boldsymbol{x}_1-\boldsymbol{x}_2)\omega_1}{C(0)^2-C(\boldsymbol{x}_1-\boldsymbol{x}_2)^2}\end{aligned}\tag{5.29}$$

unter Benutzung der inversen Matrix von C_{im}

$$C_{im}=\begin{pmatrix}C(0)&C(\boldsymbol{x}_1-\boldsymbol{x}_2)\\C(\boldsymbol{x}_1-\boldsymbol{x}_2)&C(0)\end{pmatrix}\tag{5.30a}$$

$$C_{im}^{-1}=\frac{1}{C(0)^2-C(\boldsymbol{x}_1-\boldsymbol{x}_2)^2}\begin{pmatrix}C(0)&-C(\boldsymbol{x}_1-\boldsymbol{x}_2)\\-C(\boldsymbol{x}_1-\boldsymbol{x}_2)&C(0)\end{pmatrix}\tag{5.30b}$$

Wählt man Kraftparameter und Wirbelstärken $\omega_{1,2}$ bei denen die Wahrscheinlichkeitsverteilung mit der gaußschen Näherung übereinstimmt, so funktioniert diese Approximation sehr gut. Dies ist beispielhaft in Abbildung 5.10 dargestellt. Bei anderen

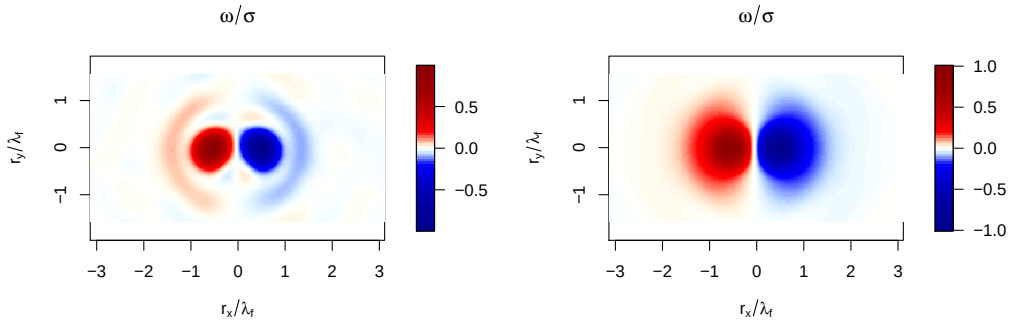


(a) Numerisch berechneter Mittelwert **(b)** Gaußsche Näherung

Abbildung 5.10.: Gegenüberstellung des auf zwei Punkte bedingten Mittelwertes bei $\omega_1 = -\omega_2 = \sigma$ und $d = \lambda_f$ mit der Näherung durch eine multivariat gaußsche Verteilung bei deterministischer Kraft und $f_A = 4\pi^2 \cdot 10^{-2}$. (Simulation: `4096inverse`, siehe Anhang A)

Kraftparametern bekommt man stattdessen wesentlich stärkere Abweichungen von

der Näherung, wie in Abbildung 5.11 zu sehen ist. Zusätzlich sieht man dort den zuvor beschriebenen Symmetriebruch in den numerischen Daten, der durch die gaußsche Näherung nicht beschrieben werden kann. Dieser Symmetriebruch ist abhängig von



(a) Numerisch berechneter Mittelwert **(b)** Gaußsche Näherung

Abbildung 5.11.: Gegenüberstellung des auf zwei Punkte bedingten Mittelwertes bei $\omega_1 = -\omega_2 = 2\sigma$ und $d = \lambda_f$ mit der gaußschen Näherung bei stochastischer Kraft und $f_A = 4\pi^2 \cdot 10^{-2}$. (Simulation: `4096inverse`, siehe Anhang A)

der Entfernung der beiden Punkte $d = |\boldsymbol{x}_1 - \boldsymbol{x}_2|$, den Wirbelstärken ω_1 und ω_2 sowie den Kraftparametern und der dabei auftretenden Abweichung der Wahrscheinlichkeitsdichte von der gaußschen Form.

5.9. Hermitesche Polynome

Wie man in Kapitel 5.8 sieht, weicht die gaußsche Näherung teilweise stark von den numerisch berechneten bedingten Mittelwerten ab. Es besteht nun die Möglichkeit, weitere Potenzen im Potenzansatz zuzulassen, auch wenn diese Methode nicht unbedingt zu gültigen Wahrscheinlichkeitsverteilungen führen muss.

Für diese Näherung soll nun angenommen werden, dass die 1-Punkt-Verteilung $f_1(\omega_1, \boldsymbol{x}_1)$ exakt gaußförmig ist und die 2-Punkt-Verteilung $f_2(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2)$ durch einen allgemeinen Polynomansatz (siehe Kapitel 5.6) beschrieben werden kann. Für die Koeffizienten von f_1 bedeutet dies $C^1_{n\neq 2} = 0$ und $C^1_2 = 2\sigma^2$. Die C^2_{nm} seien außer den zuvor genannten Einschränkungen (wie z.B. $C^1_{1n} = 0$, falls n gerade) ansonsten beliebig und über die folgende Rechnung zu bestimmen.

Ausgehend von Gleichung (5.21) ergibt sich damit

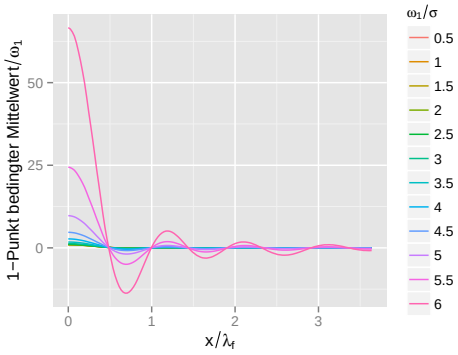
$$\begin{aligned}
&\langle \omega(\boldsymbol{x}) | \omega(\boldsymbol{x}_1) = \omega_1 \rangle \, f_1(\omega_1, \boldsymbol{x}_1) \\
&= i \sum_n C_{1n}^2(\boldsymbol{x}, \boldsymbol{x}_1) \left(i \frac{\partial}{\partial \omega_1} \right)^n f_1(\omega_1, \boldsymbol{x}_1) \\
&= i \sum_n C_{1n}^2(\boldsymbol{x}, \boldsymbol{x}_1) \, (i)^n \, \frac{\partial^n}{\partial \omega_1^n} \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-\frac{\omega_1^2}{2\sigma^2}} \\
&= i \sum_n C_{1n}^2(\boldsymbol{x}, \boldsymbol{x}_1) \, (i)^n \, \mathrm{e}^{-y^2} \mathrm{e}^{y^2} \left(\frac{1}{\sqrt{2\sigma^2}} \right)^n \frac{\partial^n}{\partial y^n} \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-y^2} \\
&= i \sum_n C_{1n}^2(\boldsymbol{x}, \boldsymbol{x}_1) \, (i)^n \left(\frac{1}{\sqrt{2\sigma^2}} \right)^n H_n(y) \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-y^2} \\
&= \sum_n b_n(\boldsymbol{x}, \boldsymbol{x}_1) H_n \left(\frac{\omega_1}{\sqrt{2\sigma^2}} \right) f_1(\omega_1, \boldsymbol{x}_1)
\end{aligned} \tag{5.31}$$

Hier wurde mit $y = \frac{1}{\sqrt{2\sigma^2}} \omega_1$ substituiert, die Definition der Hermite-Polynome $H_n(y)$ benutzt und die Koeffizientenfunktion für weitere Rechnungen mit $b_n(\boldsymbol{x}, \boldsymbol{x}_1)$ abgekürzt. Um letztgenannte Funktionen bestimmen zu können, berechnet man nun bestimmte Linearkombinationen höherer Korrelationsfunktionen in der Form

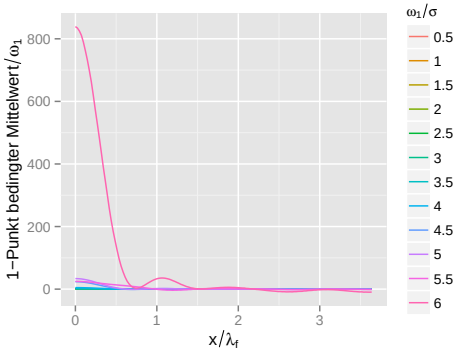
$$\begin{aligned}
&\left\langle H_m \left(\frac{\omega(\boldsymbol{x}_1)}{\sqrt{2\sigma^2}} \right) \omega(\boldsymbol{x}) \right\rangle_\omega \\
&= \iint H_m \left(\frac{\omega_1}{\sqrt{2\sigma^2}} \right) \omega_2 f_2(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}) \mathrm{d}\omega_2 \mathrm{d}\omega_1 \\
&= \int H_m \left(\frac{\omega_1}{\sqrt{2\sigma^2}} \right) \langle \omega(\boldsymbol{x}) | \omega(\boldsymbol{x}_1) = \omega_1 \rangle \, f_1(\omega_1, \boldsymbol{x}_1) \mathrm{d}\omega_1 \\
&= \int H_m \left(\frac{\omega_1}{\sqrt{2\sigma^2}} \right) \sum_n b_n(\boldsymbol{x}, \boldsymbol{x}_1) H_n \left(\frac{\omega}{\sqrt{2\sigma^2}} \right) \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-\frac{\omega_1^2}{2\sigma^2}} \mathrm{d}\omega_1 \\
&= \sum_n b_n(\boldsymbol{x}, \boldsymbol{x}_1) \int H_m(y) H_n(y) \frac{1}{\sqrt{\pi}} \mathrm{e}^{-y^2} \mathrm{d}y \\
&= \sum_n b_n(\boldsymbol{x}, \boldsymbol{x}_1) 2^n n! \, \delta_{mn} \\
&= b_m(\boldsymbol{x}, \boldsymbol{x}_1) 2^m m!
\end{aligned} \tag{5.32}$$

Dies liefert den sehr einfachen Ausdruck zur Bestimmung von b_n

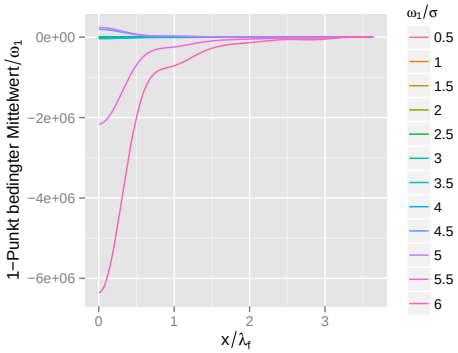
$$b_n(\boldsymbol{x}, \boldsymbol{x}_1) = \left\langle \frac{1}{2^n n!} H_n \left(\frac{\omega(\boldsymbol{x}_1)}{\sqrt{2\sigma^2}} \right) \omega(\boldsymbol{x}) \right\rangle_\omega \tag{5.33}$$



(a) deterministische Kraft, $f_A = 4\pi^2 \cdot 10^{-2}$



(b) deterministische Kraft, $f_A = 4\pi^2 \cdot 10^{-1}$



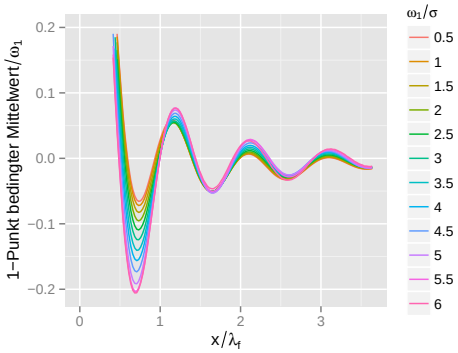
(c) deterministische Kraft, $f_A = 4\pi^2 \cdot 10^{0}$

Abbildung 5.12.: Über die Hermite-Näherung berechneter bedingter Mittelwert $\langle \omega(\boldsymbol{x}, t) | \omega(\boldsymbol{x}_1, t) = \omega_1 \rangle_\omega$ normiert auf ω_1 bei verschiedenen Kraftparametern. Es wurde hierfür bis zur Ordnung $n = 15$ genähert. (Simulation: `4096inverse`, siehe Anhang A)

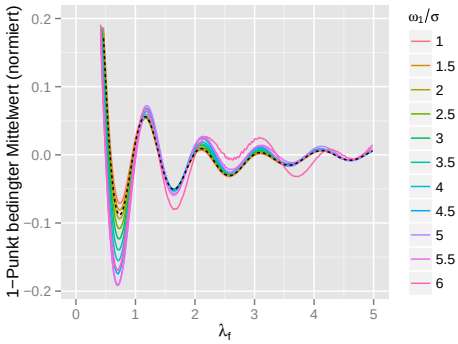
In Abbildung 5.12 sieht man für einige Simulationen die entsprechenden bedingten Mittelwerte, wie man sie aus den Korrelationsfunktionen errechnen kann. Zu erwarten wäre gewesen, dass an der Stelle $\boldsymbol{x} = \boldsymbol{x}_1$ die Identität $\langle \omega(\boldsymbol{x}_1, t) | \omega(\boldsymbol{x}_1, t) = \omega_1 \rangle_\omega = \omega_1$ gelten muss. Deswegen wurden die dargestellten Kurven auf ω_1 normiert. Der bedingte Mittelwert müsste bei $r = 0$ in den Graphen immer bei 1 anfangen. Bei der kleinsten Kraft und kleinen ω_1 scheint dies noch einigermaßen zu stimmen. Bei höheren Kraftstärken und ω_1 versagt die Näherung zunehmend. Diese starke Abweichung lässt sich natürlich dadurch begründen, dass die Wahrscheinlichkeitsdichte $f_1(\omega_1, \boldsymbol{x}_1)$ nicht gaußsch ist, wie eigentlich vom Ansatz vorausgesetzt wird und damit die Hermite-Polynome auch keine Eigenwerte zum Ableitungsoperator darstellen.

Diese Näherung macht allerdings deutlich, dass man durch Hinzunahme von höheren Potenzen in der charakteristischen Funktion eine nichtlineare Abhängigkeit von ω_1

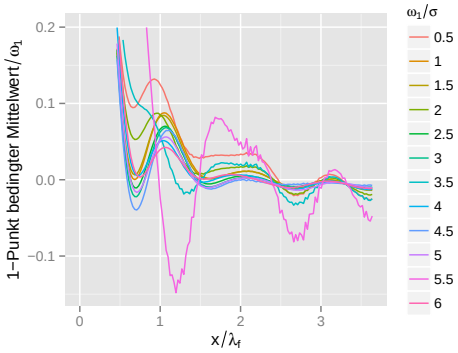
des bedingten Mittelwertes erreichen kann. Zum besseren Vergleich der Form wurde in Abbildung 5.13 der bedingte Mittelwert auf $\langle \omega(\boldsymbol{x}_1, t) | \omega(\boldsymbol{x}_1, t) = \omega_1 \rangle_\omega = \omega_1$ normiert, so dass erzwungen wird, dass die gezeigten bedingten Mittelwerte alle bei 1 anfangen. Dies hat so keinen mathematischen Hintergrund und dient nur der Überprüfung, ob man die Näherung durch Multiplikation mit einer noch zu bestimmenden Funktion $g(\omega_1)$ verbessern kann. Bei kleiner Kraftstärke funktioniert dieses Vorgehen noch relativ gut. Leider verschwindet die Aufspaltung bei höheren Kraftstärken, sodass man auch hier sagen muss, dass die Näherung nur bei nahezu gaußförmigen Wahrscheinlichkeitsdichten einigermaßen gut funktioniert. Es wäre eventuell auszuprobieren, ob die Mitnahme von höheren Ordnungen in der 1-Punkt Verteilung bessere Ergebnisse liefert.



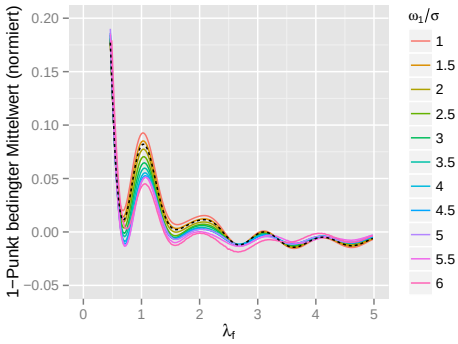
(a) Näherung: deterministische Kraft, $f_A = 4\pi^2 \cdot 10^{-2}$



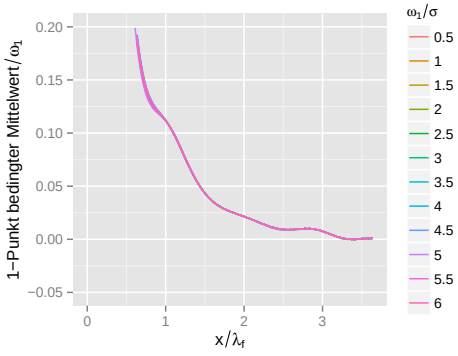
(b) Numerik: deterministische Kraft, $f_A = 4\pi^2 \cdot 10^{-2}$



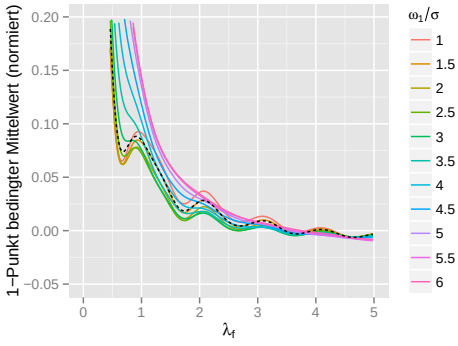
(c) Näherung: deterministische Kraft, $f_A = 4\pi^2 \cdot 10^{-1}$



(d) Numerik: deterministische Kraft, $f_A = 4\pi^2 \cdot 10^{-1}$



(e) Näherung: deterministische Kraft, $f_A = 4\pi^2 \cdot 10^{0}$



(f) Numerik: deterministische Kraft, $f_A = 4\pi^2 \cdot 10^{0}$

Abbildung 5.13.: Bedingter Mittelwert auf $\langle \omega(\boldsymbol{x}_1, t) | \omega(\boldsymbol{x}_1, t) = \omega_1 \rangle_\omega = \omega_1$ normiert bei verschiedenen Kraftparametern. Links die bis zur Ordnung $n = 15$ durchgeführte Hermite-Näherung und rechts die originalen numerischen Daten. (Simulation: `4096inverse`, siehe Anhang A)

5.10. Näherung des 1-Punkt bedingten Mittelwertes durch eine stabile Verteilung

In Kapitel 5.5 wurde gezeigt, dass die 1-Punkt Wahrscheinlichkeitsdichte von der Gaußform abweicht und eher einer allgemeineren stabilen Verteilung ähnelt. Deshalb soll nun hier der gaußförmige Ansatz von Kapitel 5.7 auf stabile Verteilungen erweitert werden. Stabile Verteilungen besitzen unendlich breite Flügel und damit existieren die höheren Momente nicht. Dies aber soll vorerst für die Näherung vernachlässigt werden.

Da es nicht ganz einfach ist, multivariat stabile Funktionen analytisch so zu definieren, dass alle Bedingungen aus Kapitel 5.5 erfüllt sind, soll nun erst einmal die 2-Punkt Verteilung betrachtet werden. Oft werden diese Verteilungen nur über ihre konvexe Kontur der charakteristischen Funktion definiert, die bei einem Exponenten von $\beta \in [1, 2]$ vorliegen muss. Für eine detaillierte Betrachtung von multivariat stabilen Verteilungen siehe [Mol09].

Nimmt man zunächst eine elliptisch konturierte multivariat stabile Verteilung, so ist

$$W_n(\alpha_1, \boldsymbol{x}_1, ..., \alpha_n, \boldsymbol{x}_n) = \left(\sum_i (\gamma \alpha_j)^2 + \sum_{ij} 2\hat{C}(r)\gamma^2 \alpha_1 \alpha_2 \right)^{\frac{\beta}{2}} \tag{5.34}$$

Diese Form erfüllt allerdings noch nicht die Bedingung, dass weit entfernte Punkte in der Wirbelstärke statistisch unabhängig werden. Hier soll nun eine Interpolation zwischen Ellipse und unabhängigen Variablen auf folgende Art und Weise definiert werden, sodass W_2 die folgende Form annimmt:

$$\begin{aligned} W_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2) = (1 - \tilde{C}(r)) \left((\gamma\alpha_1)^\beta + (\gamma\alpha_2)^\beta \right) \\ + \tilde{C}(r) \left((\gamma\alpha_1)^2 + (\gamma\alpha_2)^2 + \hat{C}(r) 2\gamma^2 \alpha_1 \alpha_2 \right)^{\frac{\beta}{2}} \end{aligned} \tag{5.35}$$

Dabei sei r der Abstand der beiden Wirbelpositionen $r = |\boldsymbol{x}_1 - \boldsymbol{x}_2|$. Die Gaußverteilung bekommt man, wenn man $\beta = 2$ wählt. Dabei sind die Funktionen so gewählt, dass $\hat{C}(0) = 1$, $\hat{C}(\infty) = 0$ und $\tilde{C}(0) = 1$, $\tilde{C}(\infty) = 0$ gilt, um zwischen statistischer Unabhängigkeit und vollständiger Abhängigkeit interpolieren zu können. Diese Definition besitzt auch identische stabile Marginalverteilungen, wie man sieht, wenn man

$\alpha_1 = 0$ oder $\alpha_2 = 0$ setzt.

$$\begin{aligned} W_2(\alpha_1, \boldsymbol{x}_1, 0, \boldsymbol{x}_2) &= (1 - \tilde{C}(r))(\gamma\alpha_1)^\beta + \tilde{C}(r) \left((\gamma\alpha_1)^2\right)^{\frac{\beta}{2}} \\ &= (1 - \tilde{C}(r))(\gamma\alpha_1)^\beta + \tilde{C}(r)(\gamma\alpha_1)^\beta \\ &= (\gamma\alpha_1)^\beta \\ &= W_1(\alpha_1, \boldsymbol{x}_1) \end{aligned} \tag{5.36}$$

Damit berechnet man wie gewohnt den bedingten Mittelwert über

$$\begin{aligned} \frac{\partial}{\partial\alpha_2} W_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2)\Big|_{\alpha_2=0} &= \tilde{C}(r)\frac{\beta}{2}\left((\gamma\alpha_1)^2\right)^{\frac{\beta}{2}-1} 2\gamma^2\alpha_1\hat{C}(r) \\ &= \tilde{C}(r)\beta\gamma^\beta\alpha_1^{\beta-1}\hat{C}(r) \\ &= \tilde{C}(r)\hat{C}(r)\frac{\partial}{\partial\alpha_1}W_1(\alpha_1, \boldsymbol{x}_1) \end{aligned} \tag{5.37}$$

Setzt man dies in Gleichung (5.12) ein, so erhält man

$$\begin{aligned} \langle\omega(\boldsymbol{x})|\omega(\boldsymbol{x}_1) = \omega_1\rangle &= \frac{1}{2\pi}\int \mathrm{e}^{-i\alpha_1\omega_1}\mathrm{e}^{-W_1(\alpha_1,\boldsymbol{x}_1)} i\left[W_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2)\right]_{\alpha_2=0}\mathrm{d}\alpha_1 \\ &= \frac{1}{2\pi}\int \mathrm{e}^{-i\alpha_1\omega_1}\mathrm{e}^{-W_1(\alpha_1,\boldsymbol{x}_1)} i\tilde{C}(r)\hat{C}(r)\frac{\partial}{\partial\alpha_1}W_1\mathrm{d}\alpha_1 \\ &= -\tilde{C}(r)\hat{C}(r)\frac{1}{2\pi}\int i\mathrm{e}^{-i\alpha_1\omega_1}\frac{\partial}{\partial\alpha_1}\mathrm{e}^{-W_1(\alpha_1,\boldsymbol{x}_1)}\mathrm{d}\alpha_1 \\ &= \tilde{C}(r)\hat{C}(r)\frac{1}{2\pi}\int i\left(\frac{\partial}{\partial\alpha_1}\mathrm{e}^{-i\alpha_1\omega_1}\right)\mathrm{e}^{-W_1(\alpha_1,\boldsymbol{x}_1)}\mathrm{d}\alpha_1 \\ &= \tilde{C}(r)\hat{C}(r)\omega_1\frac{1}{2\pi}\int \mathrm{e}^{-i\alpha_1\omega_1}\mathrm{e}^{-W_1(\alpha_1,\boldsymbol{x}_1)}\mathrm{d}\alpha_1 \\ &= \tilde{C}(r)\hat{C}(r)\omega_1 f(\omega_1, \boldsymbol{x}_1) \end{aligned} \tag{5.38}$$

Das Ergebnis ist in diesem Fall genau das gleiche, wie im Falle der gaußschen Näherung (Gleichung (5.26)) und kann damit die Formänderung bei unterschiedlichen ω_1 auch nicht erklären.

$$\langle\omega(\boldsymbol{x})|\omega(\boldsymbol{x}_1) = \omega_1\rangle = C(r)\omega_1 f(\omega_1, \boldsymbol{x}_1) \tag{5.39}$$

Der einzige Unterschied besteht darin, dass hier nur das Produkt aus $\tilde{C}(r)$ und $C(r)$ vorkommt und seine Faktoren über den 1-Punkt bedingten Mittelwert nicht zu bestimmen sind. Diese könnte man eventuell über den zweiten bedingten Moment $\left\langle\omega(\boldsymbol{x})^2\middle|\omega(\boldsymbol{x}_1) = \omega_1\right\rangle$ betrachten. Welche Form diese haben wird in Kapitel 5.11.4 genauer untersucht. Ab hier soll die Abkürzung $C(r) = \tilde{C}(r)\hat{C}(r)$ verwendet werden.

5.11. Näherung durch Faltung von stabilen Verteilungen

Da nun einfache stabile Verteilungen keinen direkten Vorteil bieten, soll hier nun eine weitere Methode getestet werden, nämlich die Faltung zweier stabiler Verteilungen. Eine Faltung zweier Verteilungen entspricht der Addition zweier Zufallsvariablen. Eventuell könnte es dabei gelingen, einer der beiden Verteilungen die Oszillationen der Kraft zuzuordnen. Da eine Faltung nur einer Multiplikation im Fourierraum entspricht, müssen dort nur die Exponenten W_2^i der einzelnen Verteilungen addiert werden. Beide Verteilungen sollen jeweils so definiert sein, wie es in Kapitel 5.10 beschrieben wurde, dabei aber unterschiedliche Exponenten besitzen.

5.11.1. Faltung zweier stabiler Verteilungen

Für die Faltung zweier verschiedener symmetrischer, stabiler und elliptisch konturierter 1-Punkt-Verteilungen $g_1(\omega_1, \boldsymbol{x}_1)$ und $h_1(\omega_1, \boldsymbol{x}_1)$ gilt

$$\begin{aligned} f_1(\omega_1, \boldsymbol{x}_1) &= g_1(\omega_1, \boldsymbol{x}_1) * h_1(\omega_1, \boldsymbol{x}_1) \\ &= \frac{1}{4\pi^2}\mathcal{F}\left[\phi_1(\alpha_1, \boldsymbol{x}_1)\psi_1(\alpha_1, \boldsymbol{x}_1)\right] \\ &= \frac{1}{4\pi^2}\mathcal{F}\left[\varphi(\alpha_1, \boldsymbol{x}_1)\right] \\ &= \frac{1}{4\pi^2}\mathcal{F}\left[\mathrm{e}^{-W_1(\alpha_1, \boldsymbol{x}_1)}\right] \end{aligned} \tag{5.40}$$

Dabei sind $\phi_1 = \mathrm{e}^{-\frac{1}{2}(\gamma\alpha_1^2)^{\frac{\beta_1}{2}}}$, $\psi_1 = \mathrm{e}^{-\frac{1}{2}(\gamma\alpha_1^2)^{\frac{\beta_2}{2}}}$ und φ_1 die charakteristischen Funktionen von g_1, h_1 und f_1. Der Exponent der charakteristischen Funktion W_1 ist gegeben als

$$\begin{aligned} W_1(\alpha_1, \boldsymbol{x}_1) &= W_1^1(\alpha_1, \boldsymbol{x}_1) + W_1^2(\alpha_1, \boldsymbol{x}_1) \\ &= \frac{1}{2}(\gamma\alpha_1^2)^{\frac{\beta_1}{2}} + \frac{1}{2}(\gamma\alpha_1^2)^{\frac{\beta_2}{2}} \end{aligned} \tag{5.41}$$

Für 2-Punkt-Verteilungen gilt ganz analog

$$\begin{aligned} f_2(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2) &= g_2(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2) * h_2(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2) \\ &= \frac{1}{4\pi^2}\mathcal{F}\left[\phi_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2)\psi_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2)\right] \\ &= \frac{1}{4\pi^2}\mathcal{F}\left[\varphi(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2)\right] \\ &= \frac{1}{4\pi^2}\mathcal{F}\left[\mathrm{e}^{-W_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2)}\right] \end{aligned} \tag{5.42}$$

Ebenso sind ϕ_2, ψ_2 und φ die charakteristischen Funktionen von g_2, h_2 und f_2.
Mit zwei Verteilungen wie in Kapitel 5.10 ist dann

$$\begin{aligned} W_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2)) = (1 - \tilde{C}_1(r)) \left((\gamma\alpha_1)^{\beta_1} + (\gamma\alpha_2)^{\beta_1} \right) \\ + \tilde{C}_1(r) \left((\gamma\alpha_2)^2 + (\gamma\alpha_2)^2 + \hat{C}_1(r) 2\gamma^2\alpha_1\alpha_2 \right)^{\frac{\beta_1}{2}} \\ + (1 - \tilde{C}_2(r)) \left((\gamma\alpha_1)^{\beta_2} + (\gamma\alpha_2)^{\beta_2} \right) \\ + \tilde{C}_2(r) \left((\gamma\alpha_2)^2 + (\gamma\alpha_2)^2 + \hat{C}_2(r) 2\gamma^2\alpha_1\alpha_2 \right)^{\frac{\beta_2}{2}} \end{aligned} \tag{5.43}$$

Auch dieses W_2 reduziert sich bei Bildung der Marginalverteilung auf ein richtig gefaltetes W_1, also bei $\alpha_1 = 0$ oder $\alpha_2 = 0$:

$$\begin{aligned} W_2(\alpha_1, \boldsymbol{x}_1, 0, \boldsymbol{x}_2) &= (1 - \tilde{C}_1(r))(\gamma\alpha_1)^{\beta_1} + \tilde{C}_1(r) \left((\gamma\alpha_1)^2 \right)^{\frac{\beta_1}{2}} \\ &\quad + (1 - \tilde{C}_2(r))(\gamma\alpha_1)^{\beta_2} + \tilde{C}_2(r) \left((\gamma\alpha_1)^2 \right)^{\frac{\beta_2}{2}} \\ &= (\gamma\alpha_1)_1^{\beta} + (\gamma\alpha_1)^{\beta_2} \\ &= W_1(\alpha_1, \boldsymbol{x}_1) \end{aligned} \tag{5.44}$$

Die anderen Eigenschaften für $r \to \infty$ und $r \to 0$ bleiben ebenfalls erhalten.

Beispielhaft ist in Abbildung 5.15 gezeigt, wie gut man mit solch einer Faltung die 1-Punkt Verteilung annähern kann. Für diese Approximation wurde eine Gaußverteilung mit einer Cauchy-Verteilung gefaltet. Man muss dazu sagen, dass die Wahl der Parameter hierbei nicht eindeutig ist und eventuell auch noch mehrere Verteilungen dazugefaltet werden müssten, um einen noch besseren Fit zu erzielen. Die abfallenden Ränder der Verteilung kann man über die Faltung weiterer stabiler Verteilungen allerdings nicht erreichen.

5.11.2. 1-Punkt bedingter Mittelwert

Mit diesem W_2 soll nun der auf einen Punkt bedingte Mittelwert des Wirbelstärkefeldes nach Gleichung (5.13) berechnet werden. Zunächst benötigt man dafür die Ableitung nach α_2

$$\begin{aligned} \frac{\partial}{\partial\alpha_2} W_2 \Big|_{\alpha_2=0} &= \frac{\beta_1}{2} ((\gamma\alpha_1)^2)^{\frac{\beta_1}{2}-1} 2C_1(r)\gamma^2\alpha_1 \\ &\quad + \frac{\beta_2}{2} ((\gamma\alpha_1)^2)^{\frac{\beta_2}{2}-1} 2C_2(r)\gamma^2\alpha_1 \\ &= C_1(r) \frac{\partial}{\partial\alpha_1} W_1^1 + C_2(r) \frac{\partial}{\partial\alpha_1} W_1^2 \end{aligned} \tag{5.45}$$

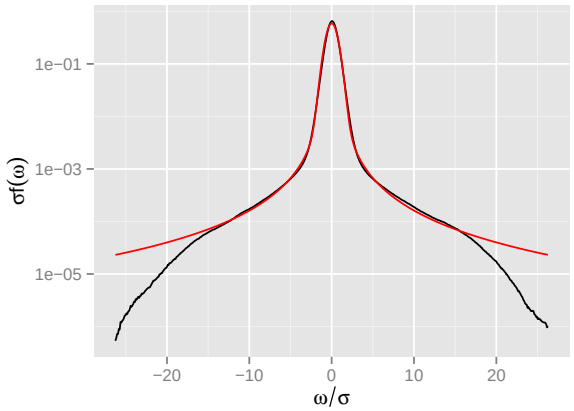


Abbildung 5.15.: Verteilungsfunktion der Wirbelstärke eines Wirbelstärkefeldes (schwarze Kurve) und dessen Näherung durch die Faltung zweier stabiler Verteilungen (rote Kurve). Parameter der Verteilungen: $\beta_1 = 1$, $\gamma_1 = 0,05$, $\beta_2 = 2$, $\gamma_2 = 0,45$. (Simulation: `4096inverse`, stochastische Kraft, $f_A = 4\pi^2 \cdot 10^{-2}$, siehe Anhang A)

Man beachte wieder die Abkürzung $C_i(r) = \tilde{C}_i(r)\hat{C}_i(r)$

Zum weiteren Vorgehen stellt man nun zunächst die Verteilung f_1 im Fourierraum dar

$$\begin{aligned} f_1(\omega_1, \boldsymbol{x}_1) &= \frac{1}{2\pi} \int \phi_1(\alpha_1, \boldsymbol{x}_1)\psi_1(\alpha_1, \boldsymbol{x}_1)\mathrm{e}^{i\alpha_1\omega_1}\mathrm{d}\alpha_1 \\ &= \frac{1}{2\pi} \iint \delta(j - \alpha_1)\phi_1(j, \boldsymbol{x}_1)\psi_1(\alpha_1, \boldsymbol{x}_1)\mathrm{e}^{i\alpha_1\omega_1}\mathrm{d}j\mathrm{d}\alpha_1 \end{aligned} \tag{5.46}$$

Setzt man dies nun in Gleichung (5.12) ein, so erhält man

$$
\begin{aligned}
&\langle\omega(\boldsymbol{x}_2)|\omega(\boldsymbol{x}_1)=\omega_1\rangle_\omega\, f_1(\omega_1,\boldsymbol{x}_1)\\
&\quad=\frac{1}{2\pi}\int \mathrm{e}^{-i\alpha_1\omega_1}\mathrm{e}^{-W_1(\alpha_1,\boldsymbol{x}_1)} i\left[W_2(\alpha_1,\boldsymbol{x}_1,\alpha_2,\boldsymbol{x}_2)\right]_{\alpha_2=0}\mathrm{d}\alpha_1\\
&\quad=i\frac{1}{2\pi}\iint \mathrm{e}^{-i\alpha_1\omega_1}\delta(j-\alpha_1)\left[-C_1(r)\frac{\partial}{\partial j}-C_2(r)\frac{\partial}{\partial\alpha_1}\right]\phi_1(j,\boldsymbol{x}_1)\psi_1(\alpha_1,\boldsymbol{x}_1)\mathrm{d}j\mathrm{d}\alpha_1\\
&\quad=i\frac{1}{4\pi^2}\iiint \mathrm{e}^{-i\alpha_1\omega_1}\mathrm{e}^{il(j-\alpha_1)}\left[-C_1(r)\frac{\partial}{\partial j}\right.\\
&\qquad\qquad\left.-C_2(r)\frac{\partial}{\partial\alpha_1}\right]\phi_1(j,\boldsymbol{x}_1)\psi_1(\alpha_1,\boldsymbol{x}_1)\mathrm{d}l\mathrm{d}j\mathrm{d}\alpha_1\\
&\quad=\frac{1}{4\pi^2}\iiint \mathrm{e}^{ilj-il\alpha_1-i\alpha_1\omega_1}\left[-lC_1(r)+(l+\omega_1)C_2(r)\right]\phi_1(j,\boldsymbol{x}_1)\psi_1(\alpha_1,\boldsymbol{x}_1)\mathrm{d}l\mathrm{d}j\mathrm{d}\alpha_1\\
&\quad=\omega_1 C_2(r) f_1(\omega_1,\boldsymbol{x}_1)\\
&\qquad+\frac{1}{4\pi^2}\left[C_2(r)-C_1(r)\right]\iiint l\mathrm{e}^{il(j-\alpha_1)}\mathrm{e}^{-i\alpha_1\omega_1}\phi_1(j,\boldsymbol{x}_1)\psi_1(\alpha_1,\boldsymbol{x}_1)\mathrm{d}l\mathrm{d}j\mathrm{d}\alpha_1
\end{aligned}
\tag{5.47}
$$

Hier wurde zunächst die Multiplikation der beiden charakteristischen Funktionen durch ein Integral über eine Delta-Distribution erweitert und diese anschließend durch die Exponentialfunktion dargestellt. Anschließend wurde partiell integriert.

Zur Berechnung des zweiten Terms benötigt man nun folgende hilfreiche Umformung

$$
\begin{aligned}
\frac{1}{4\pi^2}\int l\mathrm{e}^{il(j-\alpha_1)}\mathrm{d}l&=\frac{-i}{4\pi^2}\int\frac{\partial}{\partial(j-\alpha_1)}\mathrm{e}^{il(j-\alpha_1)}\mathrm{d}l\\
&=\frac{-i}{2\pi}\delta'(j-\alpha_1)
\end{aligned}
\tag{5.48}
$$

Über Verschiebung von $j\to j+\alpha_1$, partielle Integration und Rückverschiebung erhält man dann

$$
\begin{aligned}
&\frac{-i}{2\pi}\iint\delta'(j-\alpha_1)\mathrm{e}^{-i\alpha_1\omega_1}\phi_1(j,\boldsymbol{x}_1)\psi_1(\alpha_1,\boldsymbol{x}_1)\mathrm{d}j\mathrm{d}\alpha_1\\
&\quad=\frac{i}{2\pi}\iint\delta(j)\mathrm{e}^{-i\alpha_1\omega_1}\frac{\partial}{\partial j}\phi_1(j+\alpha_1,\boldsymbol{x}_1)\psi_1(\alpha_1,\boldsymbol{x}_1)\mathrm{d}j\mathrm{d}\alpha_1\\
&\quad=\frac{i}{2\pi}\iint\delta(j-\alpha_1)\mathrm{e}^{-i\alpha_1\omega_1}\frac{\partial}{\partial j}\phi_1(j,\boldsymbol{x}_1)\psi_1(\alpha_1,\boldsymbol{x}_1)\mathrm{d}j\mathrm{d}\alpha_1\\
&\quad=\frac{i}{2\pi}\int\mathrm{e}^{-i\alpha_1\omega_1}\psi_1(\alpha_1,\boldsymbol{x}_1)\frac{\partial}{\partial\alpha_1}\phi_1(\alpha_1,\boldsymbol{x}_1)\mathrm{d}\alpha_1
\end{aligned}
\tag{5.49}
$$

Die Bedeutung dieser Gleichung erschließt sich, wenn man die beiden charakteristischen Funktionen durch ihre Verteilungen darstellt.

$$\begin{aligned}
&\frac{i}{2\pi}\int \mathrm{e}^{-i\alpha_1\omega_1}\psi_1(\alpha_1,\boldsymbol{x}_1)\frac{\partial}{\partial\alpha_1}\phi_1(\alpha_1,\boldsymbol{x}_1)\mathrm{d}\alpha_1\\
&\quad=\frac{i}{2\pi}\int \mathrm{e}^{-i\alpha_1\omega_1}\mathrm{e}^{i\alpha_1\omega'}h(\omega')\frac{\partial}{\partial\alpha_1}\mathrm{e}^{i\alpha_1\tilde{\omega}}g(\tilde{\omega})\mathrm{d}\alpha_1\mathrm{d}\omega'\mathrm{d}\tilde{\omega}\\
&\quad=-\frac{1}{2\pi}\int \mathrm{e}^{-i\alpha_1(\omega_1-\omega'-\tilde{\omega})}h(\omega')\tilde{\omega}g(\tilde{\omega})\mathrm{d}\alpha_1\mathrm{d}\omega'\mathrm{d}\tilde{\omega}\\
&\quad=-\int \delta(\omega_1-\omega'-\tilde{\omega})h(\omega')\tilde{\omega}g(\tilde{\omega})\mathrm{d}\omega'\mathrm{d}\tilde{\omega}\\
&\quad=-\int h(\omega_1-\tilde{\omega})\tilde{\omega}g(\tilde{\omega})\mathrm{d}\omega'\mathrm{d}\tilde{\omega}
\end{aligned} \tag{5.50}$$

Dieses Ergebnis wiederum liest sich einfacher, wenn man es durch bedingte Wahrscheinlichkeiten darstellt. Wählt man zwei Zufallsgrößen X_g und Y_h mit den jeweiligen Verteilungen g und h, so sieht man

$$\langle X_g|X_g+Y_h=\omega_1\rangle=\int\delta(\omega_1-(\omega'+\tilde{\omega}))\tilde{\omega}g(\tilde{\omega})h(\omega')\mathrm{d}\omega'\mathrm{d}\tilde{\omega} \tag{5.51}$$

Dies ist das gleiche Ergebnis wie zuvor und damit lässt sich das Gesamtergebnis wie folgt schreiben:

$$\begin{aligned}
\langle\omega(\boldsymbol{x}_2)|\omega(\boldsymbol{x}_1)=\omega_1\rangle_\omega &= \omega_1C_2(r)-[C_2(r)-C_1(r)]\,\frac{\langle X_g|X_g+Y_h=\omega_1\rangle}{f(\omega_1,\boldsymbol{x}_1)}\\
&= \omega_1C_2(r)-[C_2(r)-C_1(r)]\,\frac{\langle \omega_1-Y_h|X_g+Y_h=\omega_1\rangle}{f(\omega_1,\boldsymbol{x}_1)}\\
&= \omega_1C_1(r)+[C_2(r)-C_1(r)]\,\frac{\langle Y_h|X_g+Y_h=\omega_1\rangle}{f(\omega_1,\boldsymbol{x}_1)}\\
&= \langle C_1(r)(\omega_1-Y_h)+C_2(r)Y_g|X_g+Y_h=\omega_1\rangle\,\frac{1}{f(\omega_1,\boldsymbol{x}_1)}\\
&= \langle C_1(r)X_g+C_2(r)Y_h|X_g+Y_h=\omega_1\rangle\,\frac{1}{f(\omega_1,\boldsymbol{x}_1)}
\end{aligned} \tag{5.52}$$

Die Identität zu Gleichung (5.47) ist in dieser Schreibweise leicht erkennbar. Reichen zwei gefaltete Verteilungen nicht aus, so lässt sich dieser Ausdruck auf beliebig viele Verteilungen erweitern:

$$\langle\omega(\boldsymbol{x}_2)|\omega(\boldsymbol{x}_1)=\omega_1\rangle_\omega=\left\langle\sum_i C_i(r)X_{g_i}\middle|\sum_i X_{g_i}=\omega_1\right\rangle\frac{1}{f(\omega_1,\boldsymbol{x}_1)} \tag{5.53}$$

Der bedingte Mittelwert ist hier in Abhängigkeit von ω_1 eine nicht-lineare Überlagerung mehrerer Funktionen $C_i(r)$.

5.11.3. Berechnung von C_1 und C_2

Im vorherigen Kapitel wurde nun gezeigt, wie der bedingte Mittelwert mit ω_1 skaliert. Für zwei gefaltete Verteilungen hängt dieser allerdings noch von den Funktionen C_1 und C_2 ab, die aus der Numerik zu bestimmen sind. Dafür kann man zunächst Gleichung (5.52) etwas übersichtlicher darstellen, um die Skalierung zwischen den beiden C_i besser zu sehen

$$b(\omega_1, r) := \frac{\langle \omega(\boldsymbol{x}_2) | \omega(\boldsymbol{x}_1) = \omega_1 \rangle_\omega}{\omega_1} = C_1(r) s(\omega_1) + C_2(r) \left(1 - s(\omega_1)\right) \quad (5.54)$$

Der dabei auftretende Skalierungsfaktor ist $s(\omega_1) = \frac{\langle X_g | X_g + Y_h = \omega_1 \rangle}{\omega_1 f(\omega_1, \boldsymbol{x}_1)}$.

Nun betrachtet man diese Gleichung für zwei verschiedene $\omega_{1,2}$ und formt nach C_1 um.

$$C_1(r) = \frac{1}{s(\omega_2)} \left(b(\omega_2, r) - C_2(r)(1 - s(\omega_2)\right) \quad (5.55)$$

Dies setzt man in die Gleichung für ω_1 ein und erhält

$$C_2(r) \left[(1 - s(\omega_1) - \frac{s(\omega_1)}{s(\omega_2)} (1 - s(\omega_2)) \right] = C_2(r) \left[1 - \frac{s(\omega_1)}{s(\omega_2} \right] = \frac{s(\omega_1)}{s(\omega_2)} b(\omega_2, r) \quad (5.56a)$$

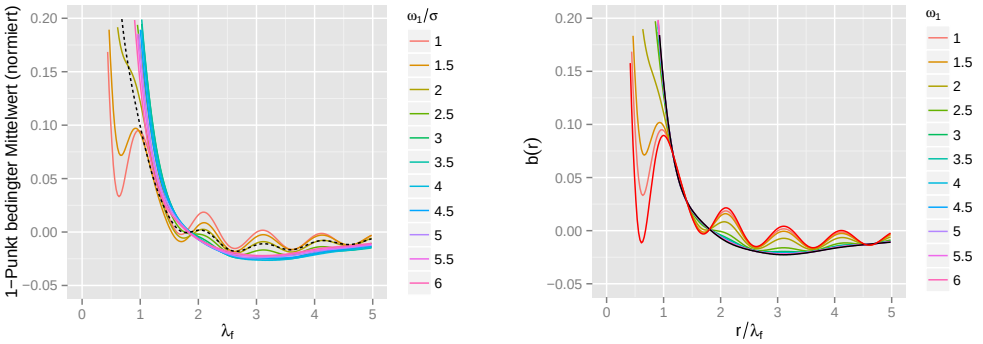
$$C_2(r) = \frac{b(\omega_1, r) - \frac{s(\omega_1)}{s(\omega_2)}}{1 - \frac{s(\omega_1)}{s(\omega_2)}} = \frac{s(\omega_2) b(\omega_1, r) - s(\omega_1) b(\omega_2, r)}{s(\omega_2) - s(\omega_1)} \quad (5.56b)$$

Dies kann man benutzen, um Gleichung (5.55) für ω_1 zu lösen und erhält

$$\begin{aligned} C_1(r) &= \frac{1}{s(\omega_2)} \left(b(\omega_2, r) - \frac{s(\omega_2) b(\omega_1, r) - s(\omega_1) b(\omega_2, r)}{s(\omega_2) - s(\omega_1)} (1 - s(\omega_2)) \right) \\ &= \frac{\frac{b(\omega_2, r)}{s(\omega_2)} (s(\omega_2) - s(\omega_1)) + (\frac{1}{s(\omega_2)} - 1) s(\omega_1) b(\omega_2, r)}{s(\omega_2) - s(\omega_1)} - \frac{b(\omega_1, r)(1 - s(\omega_2))}{s(\omega_2) - s(\omega_1)} \\ &= \frac{b(\omega_2, r)(1 - s(\omega_1)) - b(\omega_1, r)(1 - s(\omega_2))}{s(\omega_2) - s(\omega_1)} \end{aligned} \quad (5.57)$$

Über diese beiden Bestimmungsgleichungen kann man nun bei gegebenen Wahrscheinlichkeitsdichten g, h und f und den bedingten Mittelwerten $b(\omega_1, r) := \frac{\langle \omega(\boldsymbol{x}_2) | \omega(\boldsymbol{x}_1) = \omega_1 \rangle_\omega}{\omega_1}$ die Funktionen $C_1(r)$ und $C_2(r)$ numerisch berechnen.

In Abbildung 5.16 sieht man die Gegenüberstellung der Näherung über die Faltung zu den numerischen Ergebnissen. Man sieht, dass die Änderung der Form durch diese Näherung sehr gut wiedergegeben wird.



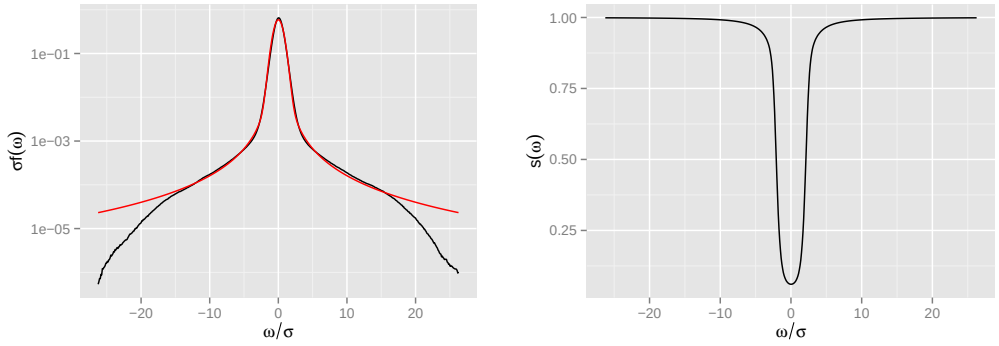
(a) Numerisches Ergebnis **(b)** Näherung über stabile Verteilungen

Abbildung 5.16.: Vergleich der Näherung des 1-Punkt bedingten Mittelwertes $b(r)$ über Faltung zweier stabiler Verteilungen mit den numerisch berechneten Daten. Die schwarze und rote durchgezogene Linie geben jeweils die Funktionen $C_1(r)$ bzw. $C_2(r)$ an. Die Näherung wurde bei den Wirbelstärken $\omega_1 = 1\sigma$ und $\omega_2 = 6\sigma$ berechnet. (Simulation: `4096inverse`, stochastische Kraft, $f_A = 4\pi^2 \cdot 10^{-2}$, siehe Anhang A)

Die zugehörige Wirbelstärkeverteilung und ihre Näherung sind in Abbildung 5.17a dargestellt. Hier sieht man, bis zu welchen Wirbelstärken beide miteinander übereinstimmen und damit auch bis wohin die Näherung ihre Gültigkeit besitzt. Die Güte der gesamten Näherung hängt natürlich von den gewählten β_i und γ_i ab. Diese Wahl ist erst einmal nicht eindeutig und es ergeben sich verschiedene Kombinationen mit ähnlichem Ergebnis. Durch genauere Untersuchungen im Bereich $\omega_1 < 3\sigma$ könnte man hier vielleicht eine genauere Aussage treffen. Über die Wahl von $\beta_1 = 1$ und $\beta_2 = 2$ erhält man ein sogenanntes Voigt-Profil, wie man es auch bei Spektrallinien vorfindet [Arm67]. Es entsteht dort durch zwei unterschiedliche Prozesse, die zur Linienaufweitung führen und dabei unterschiedliche Verteilungen besitzen.

Auf Grund der starken Oszillationen in $C_2(r)$ kann man die Gaußverteilung $h(\omega', \boldsymbol{x}_1)$ der Kraft zuordnen und die Cauchy-Verteilung $g(\tilde{\omega}, \boldsymbol{x}_1)$ mit glattem $C_1(r)$ dem Wirbeltransport. Dabei ist zu beachten, dass es sich bei $C_1(r)$ bzw. $C_2(r)$ immer noch um das Produkt $C_i(r) = \tilde{C}_i(r)\hat{C}_i(r)$ handelt. Der Skalierungsfaktor (Abbildung 5.17b) gibt dabei an, bei welchen Wirbelstärken welcher Einfluss dominiert. In diesem Fall

wären Wirbel mit $\omega < 2\sigma$ dominiert durch die Kraft und Wirbel mit $\omega > 5\sigma$ deutlich durch die andere Verteilung domiert. Als weiterer Effekt müsste noch berücksichtigt werden, dass durch die Viskosität große Wirbelstärken gedämpft werden und dadurch die Flügel schmaler werden, als bei einer Lorentz-Verteilung zu erwarten wäre.



(a) Verteilungsfunktion **(b)** Skalierungsfaktor

Abbildung 5.17.: Links: Verteilungsfunktion der Wirbelstärke eines Wirbelstärkefeldes (schwarze Kurve) und dessen Näherung durch die Faltung zweier stabiler Verteilungen (rote Kurve). Rechts: Skalierungsfaktor $s(\omega)$ Parameter der Verteilungen: $\beta_1 = 1$, $\gamma_1 = 0,05$, $\beta_2 = 2$, $\gamma_2 = 0,45$. (Simulation: `4096inverse`, stochastische Kraft, $f_A = 4\pi^2 \cdot 10^{-2}$, siehe Anhang A)

5.11.4. Numerische Bestimmung des auf einen Punkt bedingten zweiten Moments

Nun sind allerdings noch nicht die Parameter $\tilde{C}_i(r)$ bzw. $\hat{C}_i(r)$ eindeutig bestimmt. Diese könnte man vermutlich durch den auf einen Punkt bedingten zweiten Moment ($\left\langle \omega^2 \middle| \omega(\boldsymbol{x}) = \omega_1 \right\rangle_\omega$) bestimmen. Dieser ist in Abbildung 5.18 für verschiedene ω_1 dargestellt.

Zunächst sieht man, dass alle Kurven wie gewollt bei $r = 0$ beim vorgegeben ω_1^2 beginnen. Bei $r \to \infty$ streben die bedingten zweiten Momente auf eine Konstante hin. Dies ist klar, da weit entfernte Punkte statistisch unabhängig zum bedingten Punkt sein müssen und deren Varianz damit konstant der Varianz der 1-Punkt Verteilung ist, also

$$\lim_{|\boldsymbol{x}-\boldsymbol{x}_1|\to\infty} \left\langle \omega^2(\boldsymbol{x}) \middle| \omega(\boldsymbol{x}_1) = \omega_1 \right\rangle = \left\langle \omega^2 \right\rangle \tag{5.58}$$

Dazwischen treten abhängig vom gewählten ω_1 verschieden starke ortsabhängige Oszillationen auf, die sehr stark schon nach einer halben Periode sehr stark gedämpft

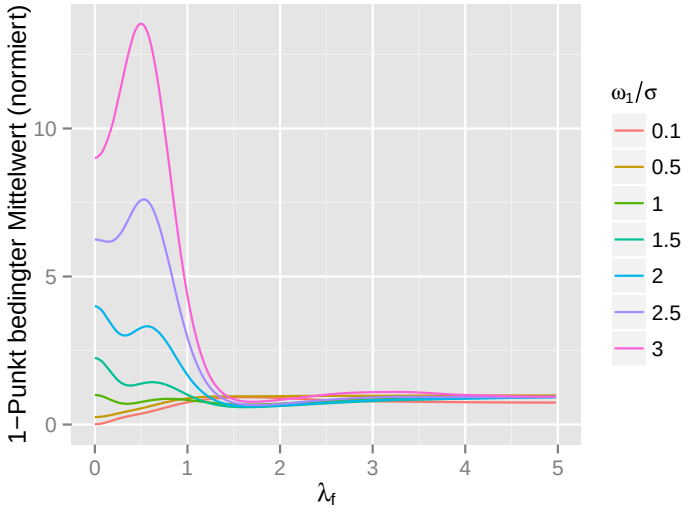


Abbildung 5.18.: Auf einen Punkt bedingter zweiter Moment $\left\langle \omega^2/\sigma^2 \middle| \omega(\boldsymbol{x}) = \omega_1 \right\rangle_\omega$ bei stochastische Kraft und $f_A = 4\pi^2 \cdot 10^{-2}$. (Simulation: `distcond`, siehe Anhang A)

sind. Diese Dämpfung ist wesentlich stärker, als bei der auf einen Punkt bedingten mittleren Wirbelstärke. Es ist daher wahrscheinlich, dass man über eine weitere Analyse dieses zweiten bedingten Moments auf die Parameter $\tilde{C}_i(r)$ bzw. $\hat{C}_i(r)$ schließen kann. Hierfür müsste es allerdings gelingen, bei der Verteilung die stark abfallenden Flügel zu beschreiben, da diese sonst dafür sorgen würden, dass der zweite Moment der Modellverteilung nicht existiert.

Zur Berechnung des auf einen Punkt bedingten zweiten Moments würde man dann ähnlich vorgehen, wie man dies bereits für den einfachen bedingten Mittelwert zuvor gemacht hat. Aus der Definition des bedingten zweiten Moments folgt direkt:

$$
\begin{aligned}
\left\langle \omega^2(\boldsymbol{x}) \middle| \omega(\boldsymbol{x}_1) = \omega_1 \right\rangle &= \int \omega_2^2 f(\omega_1, \omega_2) \mathrm{d}\omega_2 \\
&= \frac{1}{4\pi^2} \int \omega_2^2 \mathrm{e}^{-i(\alpha_1\omega_1 + \alpha_2\omega_2)} \mathrm{e}^{-W_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2)} \mathrm{d}\alpha_1 \mathrm{d}\alpha_2 \mathrm{d}\omega_2 \\
&= -\frac{1}{4\pi^2} \int \frac{\partial^2}{\partial \alpha_2^2} \mathrm{e}^{-i(\alpha_1\omega_1 + \alpha_2\omega_2)} \mathrm{e}^{-W_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2)} \mathrm{d}\alpha_1 \mathrm{d}\alpha_2 \mathrm{d}\omega_2 \\
&= -\frac{1}{4\pi^2} \int \mathrm{e}^{-i(\alpha_1\omega_1 + \alpha_2\omega_2)} \frac{\partial^2}{\partial \alpha_2^2} \mathrm{e}^{-W_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2)} \mathrm{d}\alpha_1 \mathrm{d}\alpha_2 \mathrm{d}\omega_2 \\
&= -\frac{1}{2\pi} \int \mathrm{e}^{-i\alpha_1\omega_1} \left[\frac{\partial^2}{\partial \alpha_2^2} \mathrm{e}^{-W_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2)} \right]_{\alpha_2 = 0} \mathrm{d}\omega_1 \\
&= -\frac{1}{2\pi} \int \mathrm{e}^{-i\alpha_1\omega_1} \left[\left(\frac{\partial}{\partial \alpha_2} W_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2) \right)^2 \right. \\
&\qquad \left. - \frac{\partial^2}{\partial \alpha_2^2} W_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2) \right]_{\alpha_2 = 0} \mathrm{e}^{-W_1(\alpha_1, \boldsymbol{x}_1)} \mathrm{d}\omega_1
\end{aligned}
\tag{5.59}
$$

Dies müsste man anschließend wieder über Ableitungen von e^{-W_1} darstellen, um eine geschlossene Form zu erhalten, die man numerisch auswerten könnte, so dass man hierüber wie in Kapitel 5.11.3 die Parameter $\tilde{C}_i(r)$ bzw. $\hat{C}_i(r)$ bestimmen kann.

5.11.5. 2-Punkt bedingter Mittelwert

Der auf 2 Punkte bedingte Mittelwert berechnet sich bei Faltung zweier Verteilungsfunktionen g_3 und h_3 analog zu den letzten Kapiteln. Es ist bisher unklar, wie genau diese multivariate Wahrscheinlichkeitsverteilung aussehen muss, damit sie alle in Kapitel 5.5 genannten Eigenschaften erfüllen kann. Hierfür wäre es vermutlich hilfreicht, die Wahrscheinlichkeitsverteilung für 3 Punkte genauer zu analysieren. Zunächst einmal sei hier angenommen, dass alle Punkte sehr nah beieinander liegen und dadurch die Verteilung nahezu ellipisch konturiert ist, denn für diesen Fall lässt sich die Rechnung relativ einfach durchführen. Der Exponent W_3 der charakteristischen Funktion von f_3 besitzt dann die Form

$$
\begin{aligned}
&W_3(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2, \alpha_3, \boldsymbol{x}_3) = \\
&\frac{1}{2} \left[\alpha_1 C_1(0) \alpha_1 + \alpha_2 C_1(0) \alpha_2 + \alpha_3 C_1(0) \alpha_3 \right. \\
&\left. + 2\alpha_1 C_1(\boldsymbol{x}_1 - \boldsymbol{x}_2) \alpha_2 + 2\alpha_1 C_1(\boldsymbol{x}_1 - \boldsymbol{x}_3) \alpha_3 + 2\alpha_2 C_1(\boldsymbol{x}_2 - \boldsymbol{x}_3) \alpha_3 \right]^{\frac{\beta_1}{2}} \\
&+ \frac{1}{2} \left[\alpha_1 C_2(0) \alpha_1 + \alpha_2 C_2(0) \alpha_2 + \alpha_3 C_2(0) \alpha_3 \right. \\
&\left. + 2\alpha_1 C_2(\boldsymbol{x}_1 - \boldsymbol{x}_2) \alpha_2 + 2\alpha_1 C_2(\boldsymbol{x}_1 - \boldsymbol{x}_3) \alpha_3 + 2\alpha_2 C_2(\boldsymbol{x}_2 - \boldsymbol{x}_3) \alpha_3 \right]^{\frac{\beta_2}{2}}
\end{aligned}
\tag{5.60}
$$

Die Wahl der Korrelationsfunktionen $C_{1,2}(\boldsymbol{x}_i - \boldsymbol{x}_j)$ bzw. $C_{1,2}(0)$ resultiert aus den Eigenschaften, dass die Verteilung an zwei gleichen Punkten $\boldsymbol{x}_i = \boldsymbol{x}_j$ zur 2-Punkt Verteilung degenerieren muss (z.B. $f_3(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2, \omega_3, \boldsymbol{x}_2) = f_2(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2)\delta(\omega_2 - \omega_3)$. Außerdem muss bei Integration über ω_i gelten $\int f_3(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2, \omega_3, \boldsymbol{x}_2 \mathrm{d}\omega_3 = f_2(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2)$.

Die Ableitung von W_3 nach α_3 ist dann

$$\begin{aligned}\frac{\partial}{\partial\alpha_3} W_3(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2, \alpha_3, \boldsymbol{x}_3)\Big|_{\alpha_3=0} = \frac{\beta_1}{4} &[\alpha_1 C_1(0)\alpha_1 + \alpha_2 C_1(0)\alpha_2 \\ &+ 2\alpha_1 C_1(\boldsymbol{x}_1 - \boldsymbol{x}_2)\alpha_2]^{\frac{\beta_1}{2}-1} (2\alpha_1 C_1(\boldsymbol{x}_1 - \boldsymbol{x}_3) \\ &+ 2\alpha_2 C_1(\boldsymbol{x}_2 - \boldsymbol{x}_3)) \\ + \frac{\beta_2}{4} &[\alpha_1 C_2(0)\alpha_1 + \alpha_2 C_2(0)\alpha_2 \\ &+ 2\alpha_1 C_2(\boldsymbol{x}_1 - \boldsymbol{x}_2)\alpha_2]^{\frac{\beta_2}{2}-1} (2\alpha_1 C_2(\boldsymbol{x}_1 - \boldsymbol{x}_3) \\ &+ 2\alpha_2 C_2(\boldsymbol{x}_2 - \boldsymbol{x}_3))\end{aligned} \tag{5.61}$$

Leitet man nun zum Vergleich die beiden charakteristischen Funktionen ϕ_2 von g_2 und ψ_2 von h_2 nach α_1 bzw. α_2 ab, so erhält man

$$\begin{aligned}\frac{\partial}{\partial\alpha_1}\phi_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2) = -\frac{\beta_1}{4} &[\alpha_1 C_1(0)\alpha_1 + \alpha_2 C_1(0)\alpha_2 \\ &+ 2\alpha_1 C_1(\boldsymbol{x}_1 - \boldsymbol{x}_2)\alpha_2]^{\frac{\beta_1}{2}-1} (2\alpha_1 C_1(0) \\ &+ 2\alpha_2 C_1(\boldsymbol{x}_2 - \boldsymbol{x}_3))\, \phi_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2)\end{aligned} \tag{5.62a}$$

$$\begin{aligned}\frac{\partial}{\partial\alpha_2}\phi_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2) = -\frac{\beta_1}{4} &[\alpha_1 C_1(0)\alpha_1 + \alpha_2 C_1(0)\alpha_2 \\ &+ 2\alpha_1 C_1(\boldsymbol{x}_1 - \boldsymbol{x}_2)\alpha_2]^{\frac{\beta_1}{2}-1} (2\alpha_2 C_1(0) \\ &+ 2\alpha_1 C_1(\boldsymbol{x}_2 - \boldsymbol{x}_3))\, \phi_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2)\end{aligned} \tag{5.62b}$$

$$\begin{aligned}\frac{\partial}{\partial\alpha_2}\psi_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2) = -\frac{\beta_2}{4} &[\alpha_1 C_1(0)\alpha_1 + \alpha_2 C_1(0)\alpha_2 \\ &+ 2\alpha_1 C_1(\boldsymbol{x}_1 - \boldsymbol{x}_2)\alpha_2]^{\frac{\beta_2}{2}-1} (2\alpha_1 C_2(0) \\ &+ 2\alpha_2 C_1(\boldsymbol{x}_2 - \boldsymbol{x}_3))\, \psi_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2)\end{aligned} \tag{5.62c}$$

$$\begin{aligned}\frac{\partial}{\partial\alpha_2}\psi_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2) = -\frac{\beta_2}{4} &[\alpha_1 C_1(0)\alpha_1 + \alpha_2 C_1(0)\alpha_2 \\ &+ 2\alpha_1 C_1(\boldsymbol{x}_1 - \boldsymbol{x}_2)\alpha_2]^{\frac{\beta_2}{2}-1} (2\alpha_2 C_2(0) \\ &+ 2\alpha_1 C_1(\boldsymbol{x}_2 - \boldsymbol{x}_3))\, \psi_2(\alpha_1, \boldsymbol{x}_1, \alpha_2, \boldsymbol{x}_2)\end{aligned} \tag{5.62d}$$

Möchte man mit diesen Funktionen nun den auf zwei Punkte bedingten Mittelwert darstellen, so erhält man

$$\begin{aligned}
&\langle\omega(\boldsymbol{x}_3)|\omega(\boldsymbol{x}_1)=\omega_1,\omega(\boldsymbol{x}_2)=\omega_2\rangle_\omega \\
&= i\frac{\partial}{\partial\alpha_3}W_3(\alpha_1,\boldsymbol{x}_1,\alpha_2,\boldsymbol{x}_2,\alpha_3,\boldsymbol{x}_3)\Big|_{\alpha_3=0} \\
&\frac{1}{4\pi^2}\int \mathrm{e}^{-i(\alpha_1\omega_1+\alpha_2\omega_2)}\phi_2(\alpha_1,\boldsymbol{x}_1,\alpha_2,\boldsymbol{x}_2)\psi_2(\alpha_1,\boldsymbol{x}_1,\alpha_2,\boldsymbol{x}_2)\mathrm{d}\alpha_1\mathrm{d}\alpha_2 \\
&= i\frac{1}{4\pi^2}\int \mathrm{e}^{-i(\alpha_1\omega_1+\alpha_2\omega_2)}\Bigg[\psi_2(\alpha_1,\boldsymbol{x}_1,\alpha_2,\boldsymbol{x}_2)\left(a_1\frac{\partial}{\partial\alpha_1}\right. \\
&\qquad\qquad\qquad\qquad\left. + b_1\frac{\partial}{\partial\alpha_2}\right)\phi_2(\alpha_1,\boldsymbol{x}_1,\alpha_2,\boldsymbol{x}_2) \\
&\quad + \phi_2(\alpha_1,\boldsymbol{x}_1,\alpha_2,\boldsymbol{x}_2)\left(a_2\frac{\partial}{\partial\alpha_1}+b_2\frac{\partial}{\partial\alpha_2}\right)\psi_2(\alpha_1,\boldsymbol{x}_1,\alpha_2,\boldsymbol{x}_2)\Bigg]\mathrm{d}\alpha_1\mathrm{d}\alpha_2
\end{aligned} \tag{5.63}$$

Hieraus ergeben sich zwei Bestimmungsgleichungen für die von $\alpha_{1,2}$ unabhängigen Funktionen $a_{1,2}$ und $b_{1,2}$.

$$\begin{aligned}
&a_i\alpha_1 C_i(0) + a\alpha_2 C_i(\boldsymbol{x}_1-\boldsymbol{x}_2) + b\alpha_2 C_i(0) + b\alpha_1 C_i(\boldsymbol{x}_1-\boldsymbol{x}_2) \\
&\quad = -\alpha_1 C_i(\boldsymbol{x}_1-\boldsymbol{x}_3) - \alpha_2 C_i(\boldsymbol{x}_2-\boldsymbol{x}_3)
\end{aligned} \tag{5.64}$$

Trennt man diese Terme nach den Faktoren α_i und löst dann nach a_i und b_i auf, so erhält man

$$a_i = \frac{C_i(\boldsymbol{x}_1-\boldsymbol{x}_2)C_i(\boldsymbol{x}_2-\boldsymbol{x}_3) - C_i(0)C_i(\boldsymbol{x}_1-\boldsymbol{x}_3)}{C_i(0)C_i(0) - C_i(\boldsymbol{x}_1-\boldsymbol{x}_2)C_i(\boldsymbol{x}_1-\boldsymbol{x}_2)} \tag{5.65a}$$

$$b_i = \frac{C_i(\boldsymbol{x}_1-\boldsymbol{x}_2)C_i(\boldsymbol{x}_1-\boldsymbol{x}_3) - C_i(0)C_i(\boldsymbol{x}_2-\boldsymbol{x}_3)}{C_i(0)C_i(0) - C_i(\boldsymbol{x}_1-\boldsymbol{x}_2)C_i(\boldsymbol{x}_1-\boldsymbol{x}_2)} \tag{5.65b}$$

Rechnet man nun analog zu Kapitel 5.11.2 den bedingten Mittelwert weiter aus, so ergibt sich für die Variablen $X_g^1 + Y_h^1 = \omega_1$ und $X_g^2 + Y_h^2 = \omega_2$

$$\begin{aligned}
\langle \omega(\boldsymbol{x}_3) | \omega(\boldsymbol{x}_1) = \omega_1, \omega(\boldsymbol{x}_2) = \omega_2 \rangle_\omega & \\
&= \frac{1}{f_2(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2)} \big(\big\langle a_1 X_g^1 + a_2 Y_h^1 \big| X_g^1 \\
&\qquad + Y_h^1 = \omega_1, X_g^2 + Y_h^2 = \omega_2 \big\rangle \\
&\qquad + \big\langle b_1 X_g^2 + b_2 Y_h^2 \big| X_g^1 + Y_h^1 = \omega_1, X_g^2 \\
&\qquad + Y_h^2 = \omega_2 \big\rangle \big) \\
&= a_1 \omega_1 - (a_2 \\
&\qquad - a_1) \frac{\big\langle Y_h^1 \big| X_g^1 + Y_h^1 = \omega_1, X_g^2 + Y_h^2 = \omega_2 \big\rangle}{f_2(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2)} \\
&\quad + b_1 \omega_2 - (b_2 \\
&\qquad - b_1) \frac{\big\langle Y_h^2 \big| X_g^1 + Y_h^1 = \omega_1, X_g^2 + Y_h^2 = \omega_2 \big\rangle}{f_2(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2)}
\end{aligned} \tag{5.66}$$

Definiert man nun die beiden Skalierungsfaktoren

$$s_i(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2) = \frac{\big\langle Y_h^i \big| X_g^1 + Y_h^1 = \omega_1, X_g^2 + Y_h^2 = \omega_2 \big\rangle}{\omega_i f_2(\omega_1, \boldsymbol{x}_1, \omega_2, \boldsymbol{x}_2)} \tag{5.67}$$

so erhält man für den auf zwei Punkte bedingten Mittelwert die folgende Form:

$$\begin{aligned}
&\langle \omega(\boldsymbol{x}_3) | \omega(\boldsymbol{x}_1) = \omega_1, \omega(\boldsymbol{x}_2) = \omega_2 \rangle_\omega \\
&\quad = \omega_1 (a_1(1 - s_1) + a_2 s_1) + \omega_2 (b_1(1 - s_1) + b_2 s_1) \\
&\quad = C_1(\boldsymbol{x}_2 - \boldsymbol{x}_3) \frac{(1 - s_1) C_1(\boldsymbol{x}_1 - \boldsymbol{x}_2) \omega_1 - (1 - s_2) C_1(0) \omega_2}{C_1(0) C_1(0) - C_1(\boldsymbol{x}_1 - \boldsymbol{x}_2) C_1(\boldsymbol{x}_1 - \boldsymbol{x}_2)} \\
&\qquad + C_2(\boldsymbol{x}_2 - \boldsymbol{x}_3) \frac{s_1 C_2(\boldsymbol{x}_1 - \boldsymbol{x}_2) \omega_1 - s_2 C_2(0) \omega_2}{C_2(0) C_2(0) - C_2(\boldsymbol{x}_1 - \boldsymbol{x}_2) C_2(\boldsymbol{x}_1 - \boldsymbol{x}_2)} \\
&\qquad + C_1(\boldsymbol{x}_2 - \boldsymbol{x}_3) \frac{(1 - s_2) C_1(\boldsymbol{x}_1 - \boldsymbol{x}_2) \omega_1 - (1 - s_1) C_1(0) \omega_2}{C_1(0) C_1(0) - C_1(\boldsymbol{x}_1 - \boldsymbol{x}_2) C_1(\boldsymbol{x}_1 - \boldsymbol{x}_2)} \\
&\qquad + C_2(\boldsymbol{x}_1 - \boldsymbol{x}_3) \frac{s_2 C_2(\boldsymbol{x}_1 - \boldsymbol{x}_2) \omega_1 - s_1 C_2(0) \omega_2}{C_2(0) C_2(0) - C_2(\boldsymbol{x}_1 - \boldsymbol{x}_2) C_2(\boldsymbol{x}_1 - \boldsymbol{x}_2)}
\end{aligned} \tag{5.68}$$

Vergleicht man diese Gleichung mit der bisherigen gaußschen Näherung für zwei Punkte (Gleichung (5.29)) so sieht man hier eine neue stark nicht-lineare Abhängigkeit von ω_i und eine damit verbundene Skalierung zwischen zwei Korrelationsfunktionen C_i. Dies dürfte vermutlich einen großen Teil der Abweichungen, wie sie in Abbildung

5.11 für nah benachbarte Punkte zu sehen sind, beheben. Dies wäre allerdings noch durch Auswertung der Näherung zu zeigen.

Da die C_i radialsymmetrisch sind, kann die Schrägstellung der Wirbel nicht wiedergegeben werden. Wenn ein analytisches Modell für W_3 existiert, dann käme für die Beschreibung dieser Asymmetrie vermutlich nur ein Term $\propto \alpha_1\alpha_2\alpha_3$ in Frage, da die Marginalverteilungen radialsymmetrisch sein müssen und solch ein Term bei Bildung der Marginalverteilung wegfällt.

6. Elliptisch-gaußförmige Wirbel

In Kapitel 5.3.2 wurde gezeigt, dass das auf 2 Punkte bedingte mittlere Wirbelstärkefeld in erster Näherung so aussieht wie zwei schrägstehende elliptische Wirbel. Dies soll hier als Ansatzpunkt dafür genommen werden, ein Modell für die Bewegung elliptischer Wirbel aufzustellen. Grundlage hierfür ist die Idee aus [FF11], zunächst elliptische Wirbel in die Wirbeltransportgleichung einzusetzen und die Gleichung im Fourierraum so zu vereinfachen, bis man Bewegungsgleichungen für Ausrichtung und Form erhält. Dabei wurde dort allerdings die Ausdehnung der Ellipsen entlang der Nebenachse vernachlässigt. Hier soll dieser Ansatz noch einmal Schritt für Schritt für vollständige Ellipsen nachvollzogen werden. Die dabei gemachten Näherungen werden hierbei auf etwas andere Art und Weise durchgeführt, als bisher. Als Resultat dieser Rechnung erhält man Bewegungsgleichungen für die Positionen und Formen der Ellipse. Anschließend soll noch genauer untersucht werden, wie die Bewegungsgleichungen für nur zwei Wirbel aussehen, insbesondere bei gleicher Wirbelstärke und identischem bzw. entgegengesetztem Vorzeichen.

Danach wird gezeigt, dass sich zwei elliptische Wirbel bei einer bestimmten Ausrichtung anziehen. Dieses Phänomen lässt sich in einem Punktwirbelmodell nicht beschreiben, ist aber in Übereinstimmung mit Experimenten und Simulationen [MZM88]. Außerdem wird die Stabilität dieser Ausrichtungen zueinander in Abhängigkeit vom Abstand der beiden Wirbel untersucht.

6.1. Wirbeltransportgleichung im Fourierraum

Zunächst einmal soll die Wirbeltransportgleichung im Fourierraum dargestellt werden, um dort später den Ansatz von elliptischen Wirbeln einsetzen zu können.

Über Anwendung des Biot-Savartschen Gesetzes auf die Wirbelstärke (siehe z.B. [Arg+10]) gelangt man von dieser über Integration zurück zum Geschwindigkeitsfeld

$$\boldsymbol{u}(\boldsymbol{x},t) = \int \boldsymbol{\omega}(\boldsymbol{x}',t) \times \boldsymbol{K}(\boldsymbol{x}-\boldsymbol{x}')\mathrm{d}x' \tag{6.1}$$

Hierbei ist $\boldsymbol{K} = -\nabla_x G(\boldsymbol{x}-\boldsymbol{x}')$ der Gradient der Greenfunktion des Laplace-Operators

und damit abhängig von der betrachteten Dimensionalität des Systems.

$$\boldsymbol{K}(\boldsymbol{x}-\boldsymbol{x}') = \begin{cases} \frac{1}{2\pi}\frac{\boldsymbol{x}-\boldsymbol{x}'}{|\boldsymbol{x}-\boldsymbol{x}'|^2}, & \text{für } \boldsymbol{x} \in \mathbb{R}^2 \\ \frac{1}{4\pi}\frac{\boldsymbol{x}-\boldsymbol{x}'}{|\boldsymbol{x}-\boldsymbol{x}'|^3}, & \text{für } \boldsymbol{x} \in \mathbb{R}^3 \end{cases} \tag{6.2}$$

Die Fouriertransformierte von $\boldsymbol{K}$ ist

$$\begin{aligned} \mathcal{F}\left[\boldsymbol{K}(\boldsymbol{x}-\boldsymbol{x}')\right] &= \int \boldsymbol{K}(\boldsymbol{x}-\boldsymbol{x}')\mathrm{e}^{-i\boldsymbol{k}\boldsymbol{x}}\mathrm{d}x \\ &= \int \boldsymbol{K}(\boldsymbol{x})\mathrm{e}^{-i\boldsymbol{k}(\boldsymbol{x}+\boldsymbol{x}')}\mathrm{d}x \\ &= \mathrm{e}^{-i\boldsymbol{k}\boldsymbol{x}'}\boldsymbol{K}(\boldsymbol{k}) \end{aligned} \tag{6.3}$$

Dabei ist $K(\boldsymbol{k}) = -i\frac{\boldsymbol{k}}{k^2}$ für $k \in \mathbb{R}^2$.

Daraus ergibt sich nun die Fouriertransformierte von $\boldsymbol{u}(\boldsymbol{x},t)$

$$\begin{aligned} \boldsymbol{u}(\boldsymbol{k},t) &= \int \boldsymbol{\omega}(\boldsymbol{x}',t) \times \mathrm{e}^{-i\boldsymbol{k}\boldsymbol{x}'}\boldsymbol{K}(\boldsymbol{k})\mathrm{d}x' \\ &= -\boldsymbol{K}(\boldsymbol{k}) \times \int \boldsymbol{\omega}(\boldsymbol{x}',t)\mathrm{e}^{-i\boldsymbol{k}\boldsymbol{x}'}\mathrm{d}x' \\ &= -\boldsymbol{K}(\boldsymbol{k}) \times \iint \frac{1}{(2\pi)^n}\boldsymbol{\omega}(\boldsymbol{k}',t)\mathrm{e}^{i\boldsymbol{k}'\boldsymbol{x}'}\mathrm{e}^{-i\boldsymbol{k}\boldsymbol{x}'}\mathrm{d}k'\mathrm{d}x' \\ &= -\boldsymbol{K}(\boldsymbol{k}) \times \int \boldsymbol{\omega}(\boldsymbol{k}',t)\delta(\boldsymbol{k}'-\boldsymbol{k})\mathrm{d}k' \\ &= -\boldsymbol{K}(\boldsymbol{k}) \times \boldsymbol{\omega}(\boldsymbol{k},t) \end{aligned} \tag{6.4}$$

Dies benutzt man nun, um den advektiven Term der substantiellen Ableitung zu transformieren

$$\mathcal{F}\left[(\boldsymbol{u}(\boldsymbol{x},t)\cdot\nabla)\boldsymbol{\omega}(\boldsymbol{x},t)\right] = -\frac{1}{2\pi}\int \left(\left[\boldsymbol{K}(\boldsymbol{k}')\times\boldsymbol{\omega}(\boldsymbol{k}',t)\right]\cdot i\left(\boldsymbol{k}-\boldsymbol{k}'\right)\right)\boldsymbol{\omega}(\boldsymbol{k}-\boldsymbol{k}',t)\mathrm{d}k' \tag{6.5}$$

Der Wirbelstreckungsterm wird gleichermaßen berechnet

$$\mathcal{F}\left[(\boldsymbol{\omega}(\boldsymbol{x},t)\cdot\nabla)\boldsymbol{u}(\boldsymbol{x},t)\right] = -\frac{1}{2\pi}\int \left[\boldsymbol{K}(\boldsymbol{k}')\times\boldsymbol{\omega}(\boldsymbol{k}',t)\right] i\boldsymbol{k}'\cdot\boldsymbol{\omega}(\boldsymbol{k}-\boldsymbol{k}',t)\mathrm{d}k' \tag{6.6}$$

Für inkompressible newtonische Flüssigkeiten lautet damit die Wirbeltransportgleichung (2.23) im Fourierraum

$$\begin{aligned}&\frac{\partial}{\partial t}\boldsymbol{\omega}(\boldsymbol{k},t) - \frac{1}{2\pi}\int \left(\left[\boldsymbol{K}(\boldsymbol{k}')\times\boldsymbol{\omega}(\boldsymbol{k}',t)\right]\cdot i\left(\boldsymbol{k}-\boldsymbol{k}'\right)\right)\boldsymbol{\omega}(\boldsymbol{k}-\boldsymbol{k}',t)\mathrm{d}k' \\ &= -\frac{1}{2\pi}\int \left[\boldsymbol{K}(\boldsymbol{k}')\times\boldsymbol{\omega}(\boldsymbol{k}',t)\right] i\boldsymbol{k}'\cdot\boldsymbol{\omega}(\boldsymbol{k}-\boldsymbol{k}',t)\mathrm{d}k' - \nu\boldsymbol{k}^2\boldsymbol{\omega}(\boldsymbol{k},t) + \nabla\times\tilde{f}(\boldsymbol{k},t)\end{aligned} \tag{6.7}$$

Im Spezialfall von zwei Dimensionen vereinfacht sich diese Gleichung dann auf

$$\begin{aligned}\frac{\partial}{\partial t}\omega(\boldsymbol{k},t) &= -\frac{1}{2\pi}\int \left(\left[\boldsymbol{\omega}(\boldsymbol{k}',t)\times\boldsymbol{K}(\boldsymbol{k}')\right]\cdot i\left(\boldsymbol{k}-\boldsymbol{k}'\right)\right)\omega(\boldsymbol{k}-\boldsymbol{k}',t)\mathrm{d}k' \\ &\quad - \nu\boldsymbol{k}^2\omega(\boldsymbol{k},t) + F(\boldsymbol{k},t) \\ &= -i\int \left(\boldsymbol{k}-\boldsymbol{k}'\right)\cdot\omega(\boldsymbol{k}',t)\boldsymbol{u}(\boldsymbol{k}')\omega(\boldsymbol{k}-\boldsymbol{k}',t)\mathrm{d}k' - \nu\boldsymbol{k}^2\omega(\boldsymbol{k},t) + F(\boldsymbol{k},t)\end{aligned} \tag{6.8}$$

Dabei wurde die Abkürzung $\boldsymbol{u}(\boldsymbol{k}) = \frac{-i}{2\pi}\boldsymbol{e}_z \times \frac{\boldsymbol{k}}{|\boldsymbol{k}|^2}$ verwendet, was dem Geschwindigkeitsfeld eines Punktwirbels $\boldsymbol{u}(\boldsymbol{x}) = \frac{1}{4\pi^2}\boldsymbol{e}_z \times \frac{\boldsymbol{x}}{|\boldsymbol{x}|^2}$ im Fourierraum entspricht.

6.2. Elliptische Wirbel

Ein zweidimensionaler elliptisch-gaußförmiger Wirbel lässt sich folgendermaßen definieren

$$\omega(\boldsymbol{x},t) = \Gamma\frac{1}{2\pi\sqrt{\boldsymbol{r}^2\boldsymbol{s}^2}}\mathrm{e}^{-\frac{1}{2}(\boldsymbol{r}^{-1}\cdot\boldsymbol{x})^2-\frac{1}{2}(\boldsymbol{s}^{-1}\cdot\boldsymbol{x})^2} \tag{6.9}$$

$\boldsymbol{r}$ und $\boldsymbol{s}$ zeigen in die Richtung der beiden Hauptachsen und stehen senkrecht zueinander $\boldsymbol{r}\cdot\boldsymbol{s} = 0$. Sie sind in Polarkoordinaten folgendermaßen definiert

$$\boldsymbol{r} = r\begin{pmatrix}\sin\varphi\\ \cos\varphi\end{pmatrix} \quad \boldsymbol{r}^{-1} = r^{-1}\begin{pmatrix}\sin\varphi\\ \cos\varphi\end{pmatrix} \tag{6.10a}$$

$$\boldsymbol{s} = s\begin{pmatrix}-\cos\varphi\\ \sin\varphi\end{pmatrix} \quad \boldsymbol{s}^{-1} = s^{-1}\begin{pmatrix}-\cos\varphi\\ \sin\varphi\end{pmatrix} \tag{6.10b}$$

Bei der Ellipse, bei der die Wirbelstärke auf das $\frac{1}{\mathrm{e}}$-fache abgefallen ist, sind die Längen der Hauptachsen $a = \sqrt{2}r$ und $b = \sqrt{2}s$, denn

$$\Gamma\mathrm{e}^{-\frac{x^2}{2r^2}} = \frac{\Gamma}{\mathrm{e}} = \Gamma\mathrm{e}^{-1} \tag{6.11a}$$

$$-\frac{x^2}{2r^2} = -1 \tag{6.11b}$$

$$x = \pm\sqrt{2}r \tag{6.11c}$$

Die Fläche der Ellipse bei dieser Wirbelstärke ist

$$\begin{aligned} F &= \pi xy \\ &= 2rs \end{aligned} \tag{6.12}$$

Die Analogie zum Lamb-Oseen-Wirbel [Saf93] sieht man dann, wenn man einen runden Wirbel der Größe $r = s$ betrachtet. Dann muss gelten $r^2 = 2\nu t$. Dieser hätte nun die Form

$$\omega_{\mathrm{LO}} = \frac{\Gamma}{4\pi\nu t}\mathrm{e}^{-\frac{x^2+y^2}{4\nu t}} = \frac{\Gamma}{2\pi\sqrt{r^4}}\mathrm{e}^{-\frac{1}{2}\frac{x^2+y^2}{r^2}} = \omega(\boldsymbol{x}, t) \tag{6.13}$$

Wie bei diesen Wirbeln soll auch in der elliptischen Wirbeldefinition das Γ konstant sein und die Formänderung nur über die Parameter $\boldsymbol{r}(t)$ und $\boldsymbol{s}(t)$ einfließen.

Die Definition (6.9) sorgt dafür, dass ω im Fourieraum etwas einfacher aussieht

$$\begin{aligned} \omega(\boldsymbol{k}, t) &= \Gamma\mathrm{e}^{-\frac{1}{2}(\boldsymbol{r}\cdot\boldsymbol{k})^2 - \frac{1}{2}(\boldsymbol{s}\cdot\boldsymbol{k})^2} \\ &= \Gamma\mathrm{e}^{-\frac{1}{2}\boldsymbol{k}\cdot(\boldsymbol{r}\boldsymbol{r}+\boldsymbol{s}\boldsymbol{s})\cdot\boldsymbol{k}} \\ &= \Gamma\mathrm{e}^{-\frac{1}{2}\boldsymbol{k}\cdot C\cdot\boldsymbol{k}} \end{aligned} \tag{6.14}$$

Dabei wurde im letzten Schritt die Summe der dyadischen Produkte durch die symmetrische Matrix C abgekürzt.

6.3. Bewegungsgleichungen für elliptische Wirbel

Als Ansatz wählt man nun eine Überlagerung von elliptisch-gaußförmigen Wirbeln (Gleichung (6.14))

$$\omega(\boldsymbol{k}, t) = \sum_j \omega_j(\boldsymbol{k}, t) = \sum_j \Gamma_j \mathrm{e}^{i\boldsymbol{k}\cdot\boldsymbol{x}_j - \frac{1}{2}\boldsymbol{k}\cdot C_j\cdot\boldsymbol{k}} \tag{6.15}$$

Die Wirbel sollen dabei hinreichend gut voneinander getrennt sein, so dass sie sich im Ortsraum kaum überlappen. Nun setzt man (6.15) in die Wirbeltransportgleichung (6.8) im Fourierraum ein. Dabei wird zunächst der Fall ohne treibende Kraft betrachtet:

$$\begin{aligned} &\sum_j \left(i\boldsymbol{k}\cdot\dot{\boldsymbol{x}}_j - \frac{1}{2}\boldsymbol{k}\dot{C}_j\boldsymbol{k} \right) \omega_j(\boldsymbol{k}, t) \\ &= -i\sum_{j,l} \Gamma_l\Gamma_j \int \left(\boldsymbol{k} - \boldsymbol{k}'\right) \\ &\quad \cdot \boldsymbol{u}(\boldsymbol{k}')\mathrm{e}^{i\boldsymbol{k}'\cdot\boldsymbol{x}_l - \frac{1}{2}\boldsymbol{k}'\cdot C_l\cdot\boldsymbol{k}'} \mathrm{e}^{i(\boldsymbol{k}-\boldsymbol{k}')\cdot\boldsymbol{x}_j - \frac{1}{2}(\boldsymbol{k}-\boldsymbol{k}')\cdot C_j\cdot(\boldsymbol{k}-\boldsymbol{k}')} - \nu\boldsymbol{k}^2\omega_j(\boldsymbol{k}, t) \end{aligned} \tag{6.16}$$

Auf Grund der Lokalität der einzelnen Wirbel gilt diese Gleichung nun näherungsweise für alle Summanden der Summation über j einzeln. Diese kann man dann weglassen und anschließend durch ω_j kürzen:

$$\begin{aligned} i\boldsymbol{k}\cdot\dot{\boldsymbol{x}}_j - \frac{1}{2}\boldsymbol{k}\dot{C}_j\boldsymbol{k} = -i\sum_l \Gamma_l \int \left(\boldsymbol{k}-\boldsymbol{k}'\right) \\ \cdot \boldsymbol{u}(\boldsymbol{k}')\mathrm{e}^{i\boldsymbol{k}'\cdot(\boldsymbol{x}_l-\boldsymbol{x}_j)-\frac{1}{2}\boldsymbol{k}'\cdot(C_l+C_j)\cdot\boldsymbol{k}'}\mathrm{e}^{-\boldsymbol{k}\cdot C_j\cdot\boldsymbol{k}'}\mathrm{d}k' - \nu\boldsymbol{k}^2 \end{aligned} \tag{6.17}$$

Um diese Gleichung weiter zu vereinfachen betrachtet man die Reihenentwicklung der Exponentialfunktion und ersetzt $\boldsymbol{k}'$ durch die Ableitung $-i\nabla_{lj} = -i\nabla_{\boldsymbol{R}_{lj}}$. Dabei ist $\boldsymbol{R}_{lj} = \boldsymbol{x}_l - \boldsymbol{x}_j$ und es folgt

$$\begin{aligned} i\boldsymbol{k}\cdot\dot{\boldsymbol{x}}_j - \frac{1}{2}\boldsymbol{k}\dot{C}_j\boldsymbol{k} &= -\sum_l \Gamma_l \mathrm{e}^{\frac{1}{2}\nabla_{lj}\cdot(C_l+C_j)\cdot\nabla_{lj}}\mathrm{e}^{i\boldsymbol{k}\cdot C_j\cdot\nabla_{lj}} i\left(\boldsymbol{k}-\nabla_{lj}\right) \\ &\quad\cdot\int \boldsymbol{u}(\boldsymbol{k}')\mathrm{e}^{i\boldsymbol{k}'\cdot\boldsymbol{R}_{lj}}\mathrm{d}k' - \nu\boldsymbol{k}^2 \\ &= -\sum_l \Gamma_l \mathrm{e}^{\frac{1}{2}\nabla_{lj}\cdot(C_l+C_j)\cdot\nabla_{lj}}\mathrm{e}^{i\boldsymbol{k}\cdot C_j\cdot\nabla_{lj}} i\left(\boldsymbol{k}-\nabla_{lj}\right)\cdot\boldsymbol{u}(\boldsymbol{R}_{lj}) - \nu\boldsymbol{k}^2 \end{aligned} \tag{6.18}$$

Diese Gleichung kann man nun nach Real- und Imaginärteil aufspalten und bekommt jeweils eine Entwicklungsgleichung für die Ortskoordinate x_j und eine für die Form des Wirbels C_j. Der Term $\nabla_{lj}\cdot\boldsymbol{u}(\boldsymbol{R}_{lj})$ entfällt dabei, da das Punktwirbelfeld divergenzfrei ist:

$$\boldsymbol{k}\cdot\dot{\boldsymbol{x}}_j = -\sum_l \Gamma_l \mathrm{e}^{\frac{1}{2}\nabla_{lj}\cdot(C_l+C_j)\cdot\nabla_{lj}}\cos\left(\boldsymbol{k}\cdot C_j\cdot\nabla_{lj}\right)\boldsymbol{k}\cdot\boldsymbol{u}(\boldsymbol{R}_{lj}) \tag{6.19a}$$

$$\begin{aligned} \frac{1}{2}\boldsymbol{k}\dot{C}_j\boldsymbol{k} = &-\sum_l \Gamma_l \mathrm{e}^{\frac{1}{2}\nabla_{lj}\cdot(C_l+C_j)\cdot\nabla_{lj}}\sin\left(\boldsymbol{k}\cdot C_j\cdot\nabla_{lj}\right)\boldsymbol{k}\cdot\boldsymbol{u}(\boldsymbol{R}_{lj}) \\ &+\nu\boldsymbol{k}^2 \end{aligned} \tag{6.19b}$$

Das Problem mit den Gleichungen (6.19) ist offensichtlich: Sie sind nicht eindeutig für verschiedene Wahl von $\boldsymbol{k}$. Dies ist aber auch nicht verwunderlich, da der Ansatz (6.15) nicht allgemeingültig ist, sondern nur für Wirbel mit großem Abstand R_{lj} gelten soll. Für große Abstände allerdings werden die Ableitungen von $\boldsymbol{u}(\boldsymbol{R}_{lj})$ klein und man kann somit die Exponential- und Winkelfunktionen in ihre Taylorreihen entwickeln.

Anschließend sammelt man alle Terme mit der gleichen Potenz in $\boldsymbol{k}$ zusammen. Bis zur ersten Ordnung ergibt sich dann

$$\boldsymbol{k} \cdot \dot{\boldsymbol{x}}_j = -\sum_l \Gamma_l \left(1 + \frac{1}{2}\nabla_{lj} \cdot (C_l + C_j) \cdot \nabla_{lj}\right) \boldsymbol{k} \cdot \boldsymbol{u}(\boldsymbol{R}_{lj}) \tag{6.20a}$$

$$\begin{aligned}\frac{1}{2}\boldsymbol{k}\dot{C}_j\boldsymbol{k} = -\sum_l \Gamma_l \left(1 + \frac{1}{2}\nabla_{lj} \cdot (C_l + C_j) \cdot \nabla_{lj}\right) \boldsymbol{k} \\ \cdot C_j \cdot \nabla_{lj}\boldsymbol{u}(\boldsymbol{R}_{lj}) \cdot \boldsymbol{k} + \nu\boldsymbol{k}^2\end{aligned} \tag{6.20b}$$

Möchte man noch weitere Potenzen von $\boldsymbol{k}$ mitnehmen, so müsste man zunächst den Ansatz (6.15) auf höhere Potenzen in $\boldsymbol{k}$ erweitern. Dies würde auch die Beschreibung nicht-elliptischer Wirbel ermöglichen.

6.4. Selbstwechselwirkung eines Wirbels

Nun soll zunächst diskutiert werden, welche Selbstwechselwirkung ein elliptischer Wirbel in erster Näherung erzeugt. Wir wissen aus Experimenten und Simulationen [MZM88], dass diese Wirbel an ihren Spitzen Wirbelfäden erzeugen und eine Spiralstruktur bekommen würden. Da diese Form nur sehr schwer zu beschreiben ist, soll hier erst einmal angenommen werden, dass der Wirbel in erster Näherung rund sei, wie ein Lamb-Oseen-Wirbel. Die Analogie hierzu wurde in Abschnitt 6.2 schon einmal gezeigt. Hierfür lässt sich leicht ausrechnen, wie dessen Drehgeschwindigkeit ist und welche Geschwindigkeit an seiner Mitte vorherrscht.

Für einen Wirbel mit rotationssymetrischer Form gilt

$$\boldsymbol{\omega} = \omega(\rho)\boldsymbol{e}_z = \frac{\Gamma}{2\pi r^2}\mathrm{e}^{-\frac{\rho^2}{2r^2}}\boldsymbol{e}_z \tag{6.21}$$

Man geht nun davon aus, dass das Geschwindigkeitsfeld ebenfalls radialsymmetrisch mit $\boldsymbol{u}(\boldsymbol{r}) = u(\rho)\boldsymbol{e}_\varphi$ ist. Dieses errechnet sich dann leicht in Zylinderkoordinaten

$$\omega(\rho) = \nabla \times \boldsymbol{u} = \frac{1}{\rho}\frac{\partial}{\partial\rho}\rho u(\rho) = \frac{\Gamma}{2\pi r^2}\mathrm{e}^{-\frac{\rho^2}{2r^2}} \tag{6.22a}$$

$$\frac{\partial}{\partial\rho}\rho u(\rho) = \frac{\Gamma}{2\pi r^2}\rho\mathrm{e}^{-\frac{\rho^2}{2r^2}} = -\frac{\partial}{\partial\rho}\frac{\Gamma}{2\pi}\mathrm{e}^{-\frac{\rho^2}{2r^2}} \tag{6.22b}$$

$$u(\rho) = -\frac{\Gamma}{2\pi\rho}\mathrm{e}^{-\frac{\rho^2}{2r^2}} + \frac{C}{\rho} \tag{6.22c}$$

Da $u(r)$ im Ursprung stetig sein muss, folgt direkt $u(0) = 0$ und damit

$$\frac{C}{\rho} = \frac{\Gamma}{2\pi\rho} \tag{6.23a}$$

$$u(\rho) = \frac{\Gamma}{2\pi\rho}\left(1 - \mathrm{e}^{-\frac{\rho^2}{2r^2}}\right) \tag{6.23b}$$

Das Geschwindigkeitsprofil ist bei kleinen Abständen in erster Näherung linear, da

$$u(\rho)\boldsymbol{e}_\varphi \approx \frac{\Gamma}{4\pi r^2}\rho\boldsymbol{e}_\varphi \tag{6.24}$$

Die Winkelgeschwindigkeit der Flüssigkeitsrotation ist daher im Nahbereich nahezu konstant $\frac{\Gamma}{4\pi r^2}$. Da der Wirbel keine Geschwindigkeit an der Wirbelmitte erzeugt, beeinflusst er seine Bewegungsrichtung nicht selbst. Seine Selbstwechselwirkung beschränkt sich somit allein auf seine Form. In der gewählten Näherung ist dieser Beitrag ausschließlich eine Drehung des elliptischen Wirbels mit konstanter Winkelgeschwindigkeit. Statt dem r^2 im Nenner soll im Weiteren der Mittelwert $\frac{1}{2}(r^2 + s^2)$ stehen, da natürlich kein exakter Kreis vorliegt, sondern eine Ellipse.

6.5. Parametergleichungen im Vielwirbelsystem

In diesem Abschnitt sollen die Gleichungen (6.20a) und (6.20b) so umgeformt werden, dass man ein Differentialgleichungssystem für die zeitliche Evolution der Ellipsenparameter erhält. Dazu werden spezielle Werte für $\boldsymbol{k}$ gewählt, um die jeweils gewünschten Zeitableitungen auf der linken Seite der Gleichungen zu selektieren. Dies ist nur deshalb möglich, da die Gleichungen nun nicht mehr von der Amplitude von $\boldsymbol{k}$ abhängig sind.

6.5.1. Bewegungsrichtung

Wählt man $\boldsymbol{k} = \boldsymbol{e}_x$ bzw. $\boldsymbol{k} = \boldsymbol{e}_y$ in Gleichung (6.20a), so entkoppeln die Gleichungen für x- und y-Komponente. Wir gehen davon aus, dass auch ein elliptisch-gaußförmiger Wirbel wie der radialsymmetrische Wirbel in Kapitel 6.4 seine eigene Position nicht

beeinflusst. Ohne Selbstwechselwirkung sieht die Gleichung somit wie folgt aus:

$$\begin{aligned}\dot{\boldsymbol{x}}_j &= -\sum_{l\neq j}\Gamma_l\left(1+\frac{1}{2}\nabla\cdot(C_l+C_j)\cdot\nabla\right)\boldsymbol{u}(\boldsymbol{R}_{lj})\\ &= -\sum_{l\neq j}\Gamma_l\left(1+\frac{1}{2}\left(\boldsymbol{r}_l\cdot\nabla\right)^2+\frac{1}{2}\left(\boldsymbol{s}_l\cdot\nabla\right)^2+\frac{1}{2}\left(\boldsymbol{r}_j\cdot\nabla\right)^2+\frac{1}{2}\left(\boldsymbol{s}_j\cdot\nabla\right)^2\right)\boldsymbol{u}(\boldsymbol{R}_{lj})\\ &= -\sum_{l\neq j}\Gamma_l\boldsymbol{U}_{lj}(\boldsymbol{R}_{lj})\end{aligned} \tag{6.25}$$

Um weiterrechnen zu können, benötigen wir zunächst einige Ableitungen des Geschwindigkeitsfeldes $\boldsymbol{u}(\boldsymbol{R}) = \frac{1}{4\pi^2}\boldsymbol{e}_z \times \frac{\boldsymbol{R}}{|\boldsymbol{R}|^2}$. Dies ist hier exemplarisch für die x-Komponente gerechnet. Die y-Komponente folgt analog dazu.

$$\frac{\partial}{\partial x}\left(\frac{x}{x^2+y^2}\right) = \frac{y^2-x^2}{(x^2+y^2)^2} \quad \frac{\partial}{\partial y}\left(\frac{x}{x^2+y^2}\right) = -\frac{2xy}{(x^2+y^2)^2} \tag{6.26a}$$

$$\frac{\partial^2}{\partial x^2}\left(\frac{x}{x^2+y^2}\right) = \frac{2x^3-6xy^2}{(x^2+y^2)^3} \quad \frac{\partial^2}{\partial y^2}\left(\frac{x}{x^2+y^2}\right) = -\frac{2x^3-6xy^2}{(x^2+y^2)^3} \tag{6.26b}$$

$$\frac{\partial}{\partial y}\frac{\partial}{\partial x}\left(\frac{x}{x^2+y^2}\right) = -\frac{2y^3-6yx^2}{(x^2+y^2)^3} \tag{6.26c}$$

Für einen Term ist dann

$$\begin{aligned}&\left[\nabla\cdot\boldsymbol{r}\boldsymbol{r}\cdot\nabla\right]\boldsymbol{u}(\boldsymbol{R})\\ &= \left[r_x^2\partial_x^2+2r_xr_y\partial_x\partial_y+r_y^2\partial_y^2\right]\boldsymbol{u}(\boldsymbol{R})\\ &= \boldsymbol{e}_z\times\frac{1}{4\pi^2|\boldsymbol{R}|^6}\left[\left(r_x^2-r_y^2\right)\begin{pmatrix}2x^3-6xy^2\\-2y^3+6yx^2\end{pmatrix}-2r_xr_y\begin{pmatrix}2y^3-6yx^2\\2x^3-6xy^2\end{pmatrix}\right]\\ &= -\boldsymbol{e}_z\times\frac{r^2}{4\pi^2|\boldsymbol{R}|^6}\left[\sin(2\varphi)-\cos(2\varphi)\boldsymbol{e}_z\times\right]\begin{pmatrix}2y^3-6yx^2\\2x^3-6xy^2\end{pmatrix}\\ &= -\frac{r^2}{4\pi^2|\boldsymbol{R}|^6}\left[\cos(2\varphi)+\sin(2\varphi)\boldsymbol{e}_z\times\right]\begin{pmatrix}2y^3-6yx^2\\2x^3-6xy^2\end{pmatrix}\end{aligned} \tag{6.27}$$

Dabei wurde im dritten Schritt die Parametrisierung (6.10) benutzt, die im Wesentlichen auf folgende benötigte Relationen führt:

$$r_x^2-r_y^2 = -r^2\cos(2\varphi) \quad s_x^2-s_y^2 = s^2\cos(2\varphi) \tag{6.28a}$$

$$2r_xr_y = r^2\sin(2\varphi) \quad 2s_xs_y = -s^2\sin(2\varphi) \tag{6.28b}$$

Insgesamt ergibt sich nun folgende Gleichung für das Geschwindigkeitsfeld des l-ten Wirbels:

$$\begin{aligned}\boldsymbol{U}_{lj}(x,y) = &\frac{1}{4\pi^2|\boldsymbol{R}|^2}\begin{pmatrix}-y\\x\end{pmatrix}\\ &-\frac{r_l^2-s_l^2}{4\pi^2|\boldsymbol{R}|^6}\left[\cos(2\varphi_l)+\sin(2\varphi_l)\boldsymbol{e}_z\times\right]\begin{pmatrix}2y^3-6yx^2\\2x^3-6xy^2\end{pmatrix}\\ &-\frac{r_j^2-s_j^2}{4\pi^2|\boldsymbol{R}|^6}\left[\cos(2\varphi_j)+\sin(2\varphi_j)\boldsymbol{e}_z\times\right]\begin{pmatrix}2y^3-6yx^2\\2x^3-6xy^2\end{pmatrix}\end{aligned} \tag{6.29}$$

Der erste Term tritt auch bei Punktwirbeln auf und ist unabhängig von den Ellipsenparametern. Die zusätzlichen Terme erzeugen eine zusätzliche Bewegung, die nun nicht immer senkrecht zum Abstandsvektor stehen muss und daher auch anziehend oder abstoßend wirken kann. Für ein System bestehend aus zwei Wirbeln wird dies in Kapitel 6.6 weiter analysiert werden.

6.5.2. Formänderung

Um auf die Evolutionsgleichungen der Ellipsenparameter zu kommen, muss die Gleichung (6.20b) auf die Ellipsenparameter umgeschrieben werden. Da der Wirbel unter geringer Verformung in erster Näherung ein Lamb-Oseen-Wirbel ist, wird angenommen, dass der Selbstwechselwirkungsterm nur einen Einfluss auf die Drehgeschwindigkeit $\dot{\varphi}_j$ hat. Daher wird dieser zunächst in der Rechnung weggelassen und an passender Stelle die Eigenrotation hinzuaddiert.

$$\begin{aligned}\frac{1}{2}\boldsymbol{k}\dot{C}_j\boldsymbol{k} &= -\sum_{l\neq j}\Gamma_l\left(1+\frac{1}{2}\nabla_{lj}\cdot(C_l+C_j)\cdot\nabla_{lj}\right)\boldsymbol{k}\cdot C_j\cdot\nabla_{lj}\boldsymbol{u}(\boldsymbol{R}_{lj})\cdot\boldsymbol{k}+\nu\boldsymbol{k}^2\\ &= -\sum_{l\neq j}\Gamma_l\boldsymbol{k}\cdot C_j\cdot\nabla_{lj}\boldsymbol{U}_{lj}(\boldsymbol{R}_{lj})\cdot\boldsymbol{k}+\nu\boldsymbol{k}^2\end{aligned} \tag{6.30}$$

Dabei wurde benutzt, dass

$$\left(1+\frac{1}{2}\nabla\cdot(C_l+C_j)\cdot\nabla\right)\nabla\boldsymbol{u}(\boldsymbol{R}_{lj}) = \nabla\boldsymbol{U}_{lj}(\boldsymbol{R}_{lj}) = \begin{pmatrix}\partial_x U_{lj,x} & \partial_x U_{lj,y}\\ \partial_y U_{lj,x} & \partial_y U_{lj,y}\end{pmatrix} \tag{6.31}$$

Um nun $\nabla_{lj}\boldsymbol{U}_{lj}$ auszurechnen, benötigt man die Ableitungen

$$\frac{\partial}{\partial x}\frac{2y^3-6yx^2}{(x^2+y^2)^3} = \frac{24xy(x^2-y^2)}{(x^2+y^2)^4} \tag{6.32a}$$

$$\frac{\partial}{\partial y}\frac{2y^3-6yx^2}{(x^2+y^2)^3} = -\frac{6x^4-36x^2y^2+6y^4}{(x^2+y^2)^4} \tag{6.32b}$$

sowie

$$\cdot\nabla\left(\boldsymbol{e}_z \times \boldsymbol{f}(x,y)\right)\cdot = -\cdot\nabla\boldsymbol{f}(x,y)\cdot\boldsymbol{e}_z\times \tag{6.33}$$

Dann erhält man durch Einsetzen von Gleichung (6.29)

$$\begin{aligned}
&\nabla\boldsymbol{U}_{lj}(\boldsymbol{R}_{lj}) \\
&\quad= \frac{1}{4\pi^2|\boldsymbol{R}|^4}\begin{pmatrix} 2xy & y^2-x^2 \\ y^2-x^2 & -2xy \end{pmatrix} \\
&\qquad - \frac{r_l^2-s_l^2}{4\pi^2|\boldsymbol{R}|^8}\begin{pmatrix} 24xy(x^2-y^2) & -(6x^4-36x^2y^2+6y^4) \\ -(6x^4-36x^2y^2+6y^4) & -24xy(x^2-y^2) \end{pmatrix}[\cos(2\varphi_l) - \\
&\sin(2\varphi_l)\boldsymbol{e}_z\times] \\
&\qquad - \frac{r_j^2-s_j^2}{4\pi^2|\boldsymbol{R}|^8}\begin{pmatrix} 24xy(x^2-y^2) & -(6x^4-36x^2y^2+6y^4) \\ -(6x^4-36x^2y^2+6y^4) & -24xy(x^2-y^2) \end{pmatrix}[\cos(2\varphi_j) - \\
&\sin(2\varphi_j)\boldsymbol{e}_z\times]
\end{aligned} \tag{6.34}$$

Um nun herauszufinden, wie sich die Ellipsenparameter r_j, s_j und φ_j zeitlich entwickeln, müssen nun in Gleichung (6.30) verschiedene $\boldsymbol{k}$ eingesetzt werden. Zunächst formt man die linke Seite, also die Ableitung von C, unter Benutzung der Definition (6.10) um:

$$\dot{\boldsymbol{r}} = \dot{r}\boldsymbol{e}_r + r\dot{\varphi}\boldsymbol{e}_s \qquad \dot{\boldsymbol{s}} = \dot{s}\boldsymbol{e}_s - s\dot{\varphi}\boldsymbol{e}_r \tag{6.35a}$$

$$\dot{C} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\boldsymbol{r}\boldsymbol{r} + \boldsymbol{s}\boldsymbol{s}\right) = \boldsymbol{r}\dot{\boldsymbol{r}} + \dot{\boldsymbol{r}}\boldsymbol{r} + \boldsymbol{s}\dot{\boldsymbol{s}} + \dot{\boldsymbol{s}}\boldsymbol{s} \tag{6.35b}$$

Nun benötigt man die Wahl von $\boldsymbol{k} = \boldsymbol{e}_r$, $\boldsymbol{k} = \boldsymbol{e}_s$ bzw. $\boldsymbol{k} = \boldsymbol{e}_r + \boldsymbol{e}_s$ um die Ableitungen der jeweiligen Ellipsenparameter r, s bzw. φ zu selektieren.

$$\boldsymbol{e}_r\cdot\dot{C}\cdot\boldsymbol{e}_r = 2r\dot{r} = \frac{\mathrm{d}}{\mathrm{d}t}\left(r^2\right) \tag{6.36a}$$

$$\boldsymbol{e}_s\cdot\dot{C}\cdot\boldsymbol{e}_s = 2s\dot{r} = \frac{\mathrm{d}}{\mathrm{d}t}\left(s^2\right) \tag{6.36b}$$

$$\begin{aligned}
(\boldsymbol{e}_r+\boldsymbol{e}_s)\cdot\dot{C}\cdot(\boldsymbol{e}_r+\boldsymbol{e}_s) &= 2r\dot{r} + 2s\dot{s} + 2r^2\dot{\varphi} - 2s^2\dot{\varphi} \\
&= \frac{\mathrm{d}}{\mathrm{d}t}\left(r^2\right) + \frac{\mathrm{d}}{\mathrm{d}t}\left(s^2\right) + 2\left(r^2-s^2\right)\dot{\varphi}
\end{aligned} \tag{6.36c}$$

Damit man die Gleichung für den Winkel erhält, muss man noch einmal Gleichungen (6.36a) und (6.36b) von Gleichung (6.36c) abziehen. Damit erhält mal insgesamt folgende Gleichungen:

$$\begin{aligned}\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(r_j^2\right) = -\sum_{l\neq j}\Gamma_l r_j^2 \Bigg[& \frac{1}{4\pi^2|\boldsymbol{R}|^4}\boldsymbol{e}_{r_j}\cdot A\cdot\boldsymbol{e}_{r_j} \\ & - \frac{r_l^2-s_l^2}{4\pi^2|\boldsymbol{R}|^8}\boldsymbol{e}_{r_j}\cdot B\cdot\left(\cos(2\varphi_l)\boldsymbol{e}_{r_j}-\sin(2\varphi_l)\boldsymbol{e}_{s_j}\right) \\ & - \frac{r_j^2-s_j^2}{4\pi^2|\boldsymbol{R}|^8}\boldsymbol{e}_{r_j}\cdot B\cdot\left(\cos(2\varphi_j)\boldsymbol{e}_{r_j}-\sin(2\varphi_j)\boldsymbol{e}_{s_j}\right)\Bigg] + \nu\end{aligned} \tag{6.37}$$

$$\begin{aligned}\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(s_j^2\right) = -\sum_{l\neq j}\Gamma_l s_j^2 \Bigg[& \frac{1}{4\pi^2|\boldsymbol{R}|^4}\boldsymbol{e}_{s_j}\cdot A\cdot\boldsymbol{e}_{s_j} \\ & - \frac{r_l^2-s_l^2}{4\pi^2|\boldsymbol{R}|^8}\boldsymbol{e}_{s_j}\cdot B\cdot\left(\cos(2\varphi_l)\boldsymbol{e}_{s_j}+\sin(2\varphi_l)\boldsymbol{e}_{r_j}\right) \\ & - \frac{r_j^2-s_j^2}{4\pi^2|\boldsymbol{R}|^8}\boldsymbol{e}_{s_j}\cdot B\cdot\left(\cos(2\varphi_j)\boldsymbol{e}_{s_j}+\sin(2\varphi_j)\boldsymbol{e}_{r_j}\right)\Bigg] + \nu\end{aligned} \tag{6.38}$$

$$\begin{aligned}\dot{\varphi}_j = & -\frac{\Gamma_j}{2\pi\left(r^2+s^2\right)} - \sum_{l\neq j}\frac{\Gamma_l}{r_j^2-s_j^2}\Bigg[\frac{1}{4\pi^2|\boldsymbol{R}|^4}\left[r_j^2\boldsymbol{e}_{r_j}\cdot A\cdot\boldsymbol{e}_{s_j}+s_j^2\boldsymbol{e}_{s_j}\cdot A\cdot\boldsymbol{e}_{r_j}\right] \\ & - \frac{r_l^2-s_l^2}{4\pi^2|\boldsymbol{R}|^8}\left[r_j^2\cos(2\varphi_l)\boldsymbol{e}_{r_j}\cdot B\cdot\boldsymbol{e}_{s_j}+s_j^2\cos(2\varphi_l)\boldsymbol{e}_{s_j}\cdot B\cdot\boldsymbol{e}_{r_j}\right. \\ & \left. + r_j^2\sin(2\varphi_l)\boldsymbol{e}_{r_j}\cdot B\cdot\boldsymbol{e}_{r_j}-s_j^2\sin(2\varphi_l)\boldsymbol{e}_{s_j}\cdot B\cdot\boldsymbol{e}_{s_j}\right] \\ & - \frac{r_j^2-s_j^2}{4\pi^2|\boldsymbol{R}|^8}\left[r_j^2\cos(2\varphi_j)\boldsymbol{e}_{r_j}\cdot B\cdot\boldsymbol{e}_{s_j}+s_j^2\cos(2\varphi_j)\boldsymbol{e}_{s_j}\cdot B\cdot\boldsymbol{e}_{r_j}\right. \\ & \left. + r_j^2\sin(2\varphi_j)\boldsymbol{e}_{r_j}\cdot B\cdot\boldsymbol{e}_{r_j}-s_j^2\sin(2\varphi_j)\boldsymbol{e}_{s_j}\cdot B\cdot\boldsymbol{e}_{s_j}\right]\Bigg]\end{aligned} \tag{6.39}$$

Dabei sind A und B die beiden Matrizen aus Gleichung (6.34). In der Gleichung (6.39) ist nun auch der Selbstwechselwirkungsterm näherungsweise berücksichtigt worden. Auf den ersten Blick sehen die drei Gleichungen ziemlich unhandlich aus und es lässt sich nicht direkt erschließen, wie die Matrizen A und B auf die Einheitsvektoren wirken. In Kapitel 6.6 wird dies für zwei einzelne Wirbel explizit ausgewertet und nimmt dort eine wesentlich einfachere Form an.

6.6. Wechselwirkung zweier Wirbel

In diesem Abschnitt soll die Bewegung und zeitliche Verformung zweier Wirbel betrachtet werden. Dafür betrachtet man ein ruhendes, nicht-rotierendes Koordinatensystem, in dem beide Wirbel momentan horizontal zueinander stehen, also $R_{21} = \boldsymbol{x}_2 - \boldsymbol{x}_1 = x\boldsymbol{e}_x$. Dies hat den Vorteil, dass man sich nicht zusätzlich überlegen muss, wie sich die Gleichungen für ein zeitabhängiges $\boldsymbol{k}$ ändern müssten.

6.6.1. Bewegungsrichtung

Setzt man nun $R_{21} = x\boldsymbol{e}_x$ in Gleichung (6.29) ein und wechselt damit ins Koordinatensystem, in dem der Abstandsvektor beider Wirbel momentan horizontal ist, so erhält man

$$\begin{aligned}\boldsymbol{U}_{21}(x,0) = &\frac{1}{4\pi^2 x}\begin{pmatrix}0\\1\end{pmatrix} \\ &- \frac{1}{4\pi^2 x^3}\left[\left(r_2^2 - s_2^2\right)\begin{pmatrix}-\sin(2\varphi_2)\\ \cos(2\varphi_2)\end{pmatrix} + \left(r_1^2 - s_1^2\right)\begin{pmatrix}-\sin(2\varphi_1)\\ \cos(2\varphi_1)\end{pmatrix}\right]\end{aligned} \tag{6.40}$$

Der erste Term ist identisch zur Bewegung von Punktwirbeln. Dazu ist nun ein Term gekommen, der abhängig von der Verformung und der Ausrichtung der beiden Ellipsen eine anziehende oder abstoßende Bewegung verursacht.

Betrachtet man nun ausschließlich die x-Komponente, so gilt:

$$\begin{aligned}\dot{x} &= \dot{x}_2 - \dot{x}_1 \\ &= -\Gamma_1 \boldsymbol{U}_{12}(x,0)\cdot\boldsymbol{e}_x + \Gamma_2 \boldsymbol{U}_{21}(x,0)\cdot\boldsymbol{e}_x \\ &= \left(\Gamma_1 + \Gamma_2\right)\boldsymbol{U}_{21}(x,0)\cdot\boldsymbol{e}_x \\ &= \frac{\Gamma_1+\Gamma_2}{4\pi^2 x^3}\left[\left(r_2^2 - s_2^2\right)\sin(2\varphi_2) + \left(r_1^2 - s_1^2\right)\sin(2\varphi_1)\right]\end{aligned} \tag{6.41}$$

Im zweiten Schritt wurde dabei die Punktsymmetrie des Geschwindigkeitsfeldes ausgenutzt $\boldsymbol{U}_{21} = -\boldsymbol{U}_{12}$. Für eine Auswahl an Orientierungen sind die entsprechenden Bewegungsrichtungen in Tabelle 6.1 angegeben.

Dieses Ergebnis ist in guter Übereinstimmung im Vergleich zu [MZM88]. Verschmelzende Wirbel nehmen dort einen Winkel von etwa 45°bezüglich ihres Abstandsvektors ein. Nach diesem neuen Modell ergibt sich bei diesem Winkel die stärkste Anziehung. Sollte dieser Winkel stabil sein, so könnten sich in diesem Modell zwei Wirbel einfangen und miteinander verschmelzen.

φ_1	φ_2	$\dot{\boldsymbol{x}}$
$\uparrow$	$\uparrow$	$\downarrow$
$\leftarrow$	$\leftarrow$	$\uparrow$
$\nwarrow$	$\nwarrow$	$\leftarrow$
$\nearrow$	$\nearrow$	$\rightarrow$

φ_1	φ_2	$\dot{\boldsymbol{x}}$
$\uparrow$	$\leftarrow$	$\cdot$
$\leftarrow$	$\uparrow$	$\cdot$
$\nearrow$	$\nwarrow$	$\cdot$
$\nwarrow$	$\nearrow$	$\cdot$

Tabelle 6.1.: Bewegungsrichtung eines elliptisch-gaußförmigen Wirbels mit $\Gamma > 0$ im Geschwindigkeitsfeld eines zweiten solchen Wirbels. Die Pfeile geben die Ausrichtung der Wirbel bezüglich ihres momentanen Abstandsvektors an sowie dessen Änderung.

Die Drehgeschwindigkeit des Abstandsvektors ω_x hängt ausschließlich von den y-Komponenten der Geschwindigkeiten der beiden Ellipsen ab.

$$\begin{aligned} x\omega_x = U_y &= \Gamma_1 U_{21}(x,0)\cdot\boldsymbol{e}_y - \Gamma_2 U_{12}(-x,0)\cdot\boldsymbol{e}_y \\ &= (\Gamma_1+\Gamma_2)\, U_{21}(x,0)\cdot\boldsymbol{e}_y \end{aligned} \tag{6.42a}$$

$$\begin{aligned} \omega_x = (\Gamma_1+\Gamma_2)\Bigg[&\frac{1}{4\pi^2x^2} \\ &- \frac{1}{4\pi^2x^4}\left[\left(r_2^2-s_2^2\right)\cos(2\varphi_2) + \left(r_1^2-s_1^2\right)\cos(2\varphi_1)\right]\Bigg] \end{aligned} \tag{6.42b}$$

Der erste Term entspricht hierbei wieder der Drehgeschwindigkeit von zwei Punktwirbeln umeinander.

6.6.2. Formänderung

Wie zuvor betrachtet man nun den Spezialfall von Wirbeln im Abstand $R_{21} = x\boldsymbol{e}_x$ und berechnet die Produkte der Einheitsvektoren mit den Matrizen A und B in den Gleichungen (6.37) bis (6.39). Zunächst sieht man, dass

$$A = \begin{pmatrix} 0 & -x^2 \\ -x^2 & 0 \end{pmatrix} = -x^2 \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = -x^2 T \tag{6.43a}$$

$$B = \begin{pmatrix} 0 & -6x^4 \\ -6x^4 & 0 \end{pmatrix} = -6x^4 T \tag{6.43b}$$

Nun erkennt man sofort, dass die Mischterme mit B verschwinden ($\boldsymbol{e}_r \cdot B \cdot \boldsymbol{e}_s = \boldsymbol{e}_s \cdot B \cdot \boldsymbol{e}_r = 0$). Für die restlichen Terme mit T gilt

$$\boldsymbol{e}_r \cdot T \cdot \boldsymbol{e}_r = \sin(2\varphi) \tag{6.44a}$$

$$\boldsymbol{e}_s \cdot T \cdot \boldsymbol{e}_s = -\sin(2\varphi) \tag{6.44b}$$

$$\boldsymbol{e}_r \cdot T \cdot \boldsymbol{e}_s = \boldsymbol{e}_s \cdot T \cdot \boldsymbol{e}_r = -\cos(2\varphi) \tag{6.44c}$$

Eingesetzt in Gleichung (6.37) bis (6.39) ergeben sich so die Evolutionsgleichungen für die Ausdehnung der Ellipsen und deren Orientierung:

$$\begin{aligned}
\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(r_1^2\right) = -\Gamma_2 r_1^2 \Bigg(& \frac{1}{4\pi^2 x^2}(r_1^2 - s_1^2)\sin(2\varphi_1) \\
& - \frac{6}{4\pi^2 x^4}(r_1^2 - s_1^2)\left[\cos(2\varphi_1)\sin(2\varphi_1) + \sin(2\varphi_1)\cos(2\varphi_1)\right] \\
& - \frac{6}{4\pi^2 x^4}(r_2^2 - s_2^2)\left[\cos(2\varphi_2)\sin(2\varphi_1) + \sin(2\varphi_2)\cos(2\varphi_1)\right]\Bigg) + \nu \\
= -\Gamma_2 r_1^2 \Bigg(& + \frac{1}{4\pi^2 x^2}(r_1^2 - s_1^2)\sin(2\varphi_1) - \frac{6}{4\pi^2 x^4}(r_1^2 - s_1^2)\sin(4\varphi_1) \\
& - \frac{6}{4\pi^2 x^4}(r_2^2 - s_2^2)\sin(2\varphi_1 + 2\varphi_2)\Bigg) + \nu
\end{aligned} \tag{6.45}$$

$$\begin{aligned}
\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(s_1^2\right) = -\Gamma_2 s_1^2 \Bigg(& -\frac{1}{4\pi^2 x^2}(r_1^2 - s_1^2)\sin(2\varphi_1) \\
& - \frac{6}{4\pi^2 x^4}(r_1^2 - s_1^2)\left[-\cos(2\varphi_1)\sin(2\varphi_1) - \sin(2\varphi_1)\cos(2\varphi_1)\right] \\
& - \frac{6}{4\pi^2 x^4}(r_2^2 - s_2^2)\left[-\cos(2\varphi_2)\sin(2\varphi_1) - \sin(2\varphi_2)\cos(2\varphi_1)\right]\Bigg) + \nu \\
= -\Gamma_2 s_1^2 \Bigg(& -\frac{1}{4\pi^2 x^2}(r_1^2 - s_1^2)\sin(2\varphi_1) + \frac{6}{4\pi^2 x^4}(r_1^2 - s_1^2)\sin(4\varphi_1) \\
& + \frac{6}{4\pi^2 x^4}(r_2^2 - s_2^2)\sin(2\varphi_1 + 2\varphi_2)\Bigg) + \nu
\end{aligned} \tag{6.46}$$

$$\begin{aligned}
\dot{\varphi_1} = -\frac{\Gamma_1}{2\pi\left(r^2 + s^2\right)} - \Gamma_2 \frac{r_1^2 + s_1^2}{r_1^2 - s_1^2}\Bigg[& \frac{1}{4\pi^2 x^2}\cos(2\varphi_1) \\
& - \frac{6}{4\pi^2 x^4}\left[(r_1^2 - s_1^2)\cos(4\varphi_1) + (r_2^2 - s_2^2)\cos(2\varphi_1 + 2\varphi_2)\right]\Bigg]
\end{aligned} \tag{6.47}$$

Flächenänderung

Eine sehr schöne Folgerung ergibt sich für die Flächenänderung der Ellipse, da sich alle von φ_i und x abhängigen Terme wegheben. Die Fläche ist gegeben als $F = 2\pi rs$, wie zuvor in Abschnitt 6.2 gezeigt wurde. Dann ergibt sich für deren Ableitung

$$\begin{aligned}\frac{\mathrm{d}F}{\mathrm{d}t} &= 2\pi\frac{\mathrm{d}r_1 s_1}{\mathrm{d}t} = 2\pi\frac{\mathrm{d}r_1}{\mathrm{d}t}s_1 + r_1\frac{\mathrm{d}s_1}{\mathrm{d}t} = 2\pi\left(\frac{s_1}{2r_1}\frac{\mathrm{d}r_1^2}{\mathrm{d}t} + \frac{s_1}{2r_1}\frac{\mathrm{d}r_1^2}{\mathrm{d}t}\right) \\ &= 2\pi\frac{s_1}{r_1}\nu + \frac{r_1}{s_1}\nu = 2\pi\frac{r_1^2 + s_1^2}{r_1 s_1}\nu\end{aligned} \tag{6.48}$$

Was hier auffällt ist zum einen, dass die Wirbel ihre Flächenänderung nicht gegenseitig beeinflussen. Zum anderen ändern die Wirbel ihre Fläche nicht, wenn die Viskosität $\nu = 0$ ist. Letzteres wäre im Falle eines einzigen Wirbels auch zu erwarten gewesen, da die Viskosität in der Wirbeltransportgleichung dafür verantwortlich ist, dass die Wirbelstrukturen zerfließen. Hier trifft dies sogar in Gegenwart eines zweiten Wirbels zu.

Runder Wirbel

Im Falle eines runden Wirbels ist $r_1 = s_1$. In Kapitel 6.2 wurde bereits für diesen Fall die Analogie zu einem Lamb-Oseen-Wirbel gezeigt, falls $r^2 = 2\nu t$ gilt. Dies kann nun anhand der Evolutionsgleichungen für die Ausdehnungen des elliptischen Wirbels verifiziert werden.

Für $r_1 = s_1$ folgt für die Zeitableitung von r^2 bzw. s^2:

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(r^2\right) = \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(s^2\right) = \nu \tag{6.49}$$

Daraus folgt direkt das geforderte Ergebnis:

$$r^2 = s^2 = 2\nu t \tag{6.50}$$

Für die Fläche eines Lamb-Oseen-Wirbels, bei dem er auf das e^{-1}-fache seines Maximums abgefallen ist, gilt:

$$F_{\mathrm{LO}} = \pi R^2 = \pi 4\nu t \tag{6.51}$$

wie man aus der Definition 6.13 schnell ablesen kann.

Setzt man nun in Gleichung (6.48) die Bedingung $r = s$ ein und integriert den nun konstanten Term über die Zeit, so erhält man

$$F_{\mathrm{rund}} = 4\pi\nu t \tag{6.52}$$

Die Bewegungsgleichungen sind also in diesem Punkt immer noch konsistent mit dem Lamb-Oseen-Wirbel.

6.6.3. Stabile Winkel

Von besonderer Bedeutung ist die Stabilität des Winkels φ_1, wenn man das Verschmelzen von Wirbel beschreiben möchte. Hierfür müssen nach den Ergebnissen von Kapitel 6.6.1 die Wirbel eine Ausrichtung von ungefähr 45°zur Verbindungsachse besitzen.

Notwendig für Stabilität ist, dass die Drehgeschwindigkeit $\dot{\varphi}_1$ gleich der Rotationsgeschwindigkeit ω_x der beiden Wirbel umeinander ist, wie sie in Gleichung (6.42b) berechnet wurde:

$$\begin{aligned}\dot{\varphi}_1 &= -\frac{\Gamma_1}{2\pi\,(r^2+s^2)} - \Gamma_2 \frac{r_1^2+s_1^2}{r_1^2-s_1^2}\left[\frac{1}{4\pi^2x^2}\cos(2\varphi_1)\right.\\ &\quad\left. - \frac{6}{4\pi^2x^4}\left[(r_1^2-s_1^2)\cos(4\varphi_1) + (r_2^2-s_2^2)\cos(2\varphi_1+2\varphi_2)\right]\right]\\ &= (\Gamma_1+\Gamma_2)\left[\frac{1}{4\pi^2x^2} - \frac{r_2^2-s_2^2}{4\pi^2x^4}\cos(2\varphi_2) - \frac{r_1^2-s_1^2}{4\pi^2x^4}\cos(2\varphi_1)\right]\\ &= \omega_x\end{aligned} \tag{6.53}$$

Löst man diese Formel nach $\dot{\varphi}_1 - \omega_x$ auf und kürzt die längeren Terme ab, so erhält man die Form

$$\begin{aligned}\dot{\varphi}_1 - \omega_x &= a\cos(2\varphi_1) + b\cos(2\varphi_1 - 2\varphi_2) + c\cos(4\varphi_1) + d\\ &= -\frac{\mathrm{d}}{\mathrm{d}\varphi_1}V(\varphi_1,\varphi_2) \stackrel{!}{=} 0\end{aligned} \tag{6.54}$$

mit entsprechenden Konstanten a bis d.

Dabei nimmt man an, dass die Änderung der Ausdehnung des Wirbels klein ist gegenüber der Rotationsgeschwindigkeit. Zur korrekten Berechnung des Potentials müsste zusätzlich noch die Bewegungsgleichung für den zweiten Wirbel betrachtet werden. An dieser Stelle soll aber diese Situation nicht allgemein behandelt werden, sondern erst einmal nur zwei symmetrische Spezialfälle untersucht werden, bei denen diese Gleichung und das Potential etwas einfacher aussehen.

6.6.4. Spezialfall: Gleiche Ausrichtung

Zunächst soll nun der Fall betrachtet werden, dass die beiden Wirbel die gleiche Wirbelstärke haben und als Anfangsbedingung auch die gleiche Wirbelform. Im zeitlichen Verlauf ohne äußere Störung behält das System nun seine Punktsymmetrie bei, wie dies insbesondere in [MZM88] zu sehen ist. Man erwartet in diesem Fall die Verschmelzung der beiden Wirbel. Die Ellipsenparameter beider Wirbel werden hier identisch gewählt: $\Gamma_1 = \Gamma_2 = \Gamma$, $\boldsymbol{r}_1 = \boldsymbol{r}_2 = \boldsymbol{r}$ und $\boldsymbol{s}_1 = \boldsymbol{s}_2 = \boldsymbol{s}$.

Bewegungsgleichungen

Die Bewegungsgleichungen lauten mit diesen Annahmen:

$$\dot{x} = \frac{\Gamma}{\pi^2 x^3}\left(r^2 - s^2\right)\sin(2\varphi) \tag{6.55a}$$

$$\omega_x = \frac{2\Gamma}{4\pi^2 x^2} - \frac{4\Gamma}{4\pi^2 x^4}\left(r^2 - s^2\right)\cos(2\varphi) \tag{6.55b}$$

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(r^2\right) = \Gamma r^2(r^2 - s^2)\left[-\frac{1}{4\pi^2 x^2}\sin(2\varphi) + \frac{12}{4\pi^2 x^4}\sin(4\varphi)\right] + \nu \tag{6.55c}$$

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(s^2\right) = \Gamma s^2(r^2 - s^2)\left[\frac{1}{4\pi^2 x^2}\sin(2\varphi) - \frac{12}{4\pi^2 x^4}\sin(4\varphi)\right] + \nu \tag{6.55d}$$

$$\dot{\varphi} = -\frac{\Gamma}{2\pi\left(r^2 + s^2\right)} - \Gamma\frac{1}{4\pi^2 x^2}\frac{r^2 + s^2}{r^2 - s^2}\cos(2\varphi) + 12\Gamma\frac{r^2 + s^2}{4\pi^2 x^4}\cos(4\varphi) \tag{6.55e}$$

Brennpunkt

Eine Größe die man hieraus ableiten kann ist der Abstand zum Brennpunkt

$$f = \sqrt{2r^2 - 2s^2} \tag{6.56}$$

Diese Definition folgt entsprechend der Definition der Halbachsen der elliptischen Wirbel (Kapitel 6.2). Dieser Abstand ist ein Maß für die Verformung der Wirbel.

Die Zeitableitung des quadratischen Abstandes zum Brennpunkt ist dann

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{f^2}{4}\right) &= \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(r^2 - s^2\right) \\ &= -\Gamma\left(r^2 + s^2\right)\left(r^2 - s^2\right)\left[\frac{1}{4\pi^2 x^2}\sin(2\varphi) - \frac{12}{4\pi^2 x^4}\sin(4\varphi)\right]\end{aligned} \tag{6.57}$$

Die Verformung ist also nicht mehr von der Viskosität abhängig, sondern nur noch von der aktuellen Form und den Ausrichtungen der beiden Wirbel. Ob die Verformung stabil ist, kann man nur im Zusammenhang mit den anderen Gleichungen klären.

Stabiler Winkel

Von großer Bedeutung sind die stabilen Stellungen des Winkels φ. Ebenso wie im letzten Kapitel lässt sich die Gleichung aufstellen, bei der die Wirbel sich stabilisieren müssten. Der Mischterm fällt hierbei heraus.

$$\begin{aligned}\dot{\varphi} - \omega_x &= a\cos(2\varphi) + b\cos(4\varphi) + c \\ &= -\frac{\mathrm{d}}{\mathrm{d}\varphi}V(\varphi) \stackrel{!}{=} 0\end{aligned} \tag{6.58}$$

Dabei ist

$$a = -\Gamma\frac{r^2+s^2}{r^2-s^2}\frac{1}{4\pi^2x^2} + 4\Gamma\frac{r^2-s^2}{4\pi^2x^4} \tag{6.59a}$$

$$b = 12\Gamma\frac{r^2+s^2}{4\pi^2x^4} \tag{6.59b}$$

$$c = -\frac{\Gamma}{2\pi\left(r^2+s^2\right)} - 2\Gamma\frac{1}{4\pi^2x^2} \tag{6.59c}$$

Das Potential hierfür ist

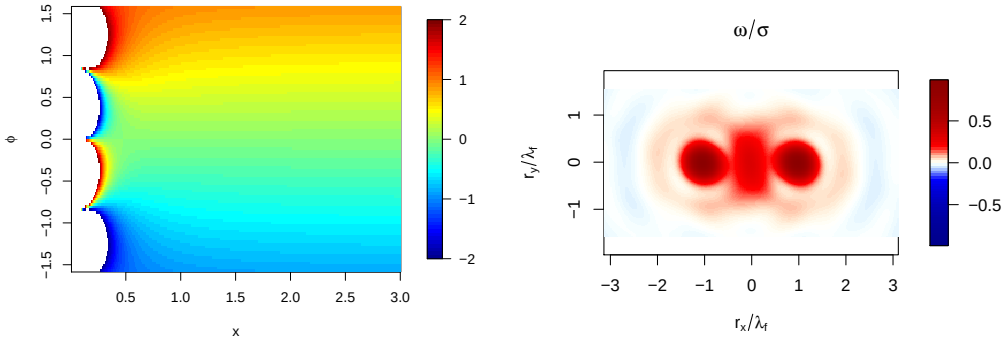
$$V(\varphi) = -\frac{a}{2}\sin(2\varphi) - \frac{b}{4}\sin(4\varphi) - c\varphi \tag{6.60}$$

Nehmen wir nun einen Fall, den wir mit den bedingten Mittelwerten in Kapitel 5.3.2, Abbildung 5.3a beobachten konnten. Hier ist $r \approx 0,6$ und $s \approx 0,8r$. Damit ergibt sich ein Potential, wie es in Abbildung 6.1 dargestellt ist.

Im Vergleichsbild ist der Abstand allerdings mit $x \approx 2$ wesentlich größer, als im Potential überhaupt Minima zu finden sind. In der Potentiallandschaft finden sich die Minima für die Wirbelorientierungen eher bei $x \approx 0,6$. Dieser Abstand wird vom Stärkeverhältnis des Selbstwechselwirkungsterms zum Wechselwirkungsterm der beiden Wirbel bestimmt. Durch eine Korrektur der gegenseitigen Wechselwirkung um ca. Faktor 20 würde diese auch im erwarteten Abstand von $x \approx 2$ stark genug sein, um die Wirbel zu stabilisieren. Das zusätzliche Minimum bei positiven Winkeln verschwindet vermutlich, wenn man in das Potential noch zusätzlich die Variablen r und s einfügt, so dass die Form des Wirbels an dieser Stelle instabil ist und dieser so gestreckt wird, dass man implizit das erwartete Minimum erreicht.

6.6.5. Spezialfall: Entgegengesetzte Ausrichtung

Nun sollen zwei Wirbel mit entgegengesetzer Wirbelstärke betrachtet werden. Die Ellipsenparameter werden dafür folgendermaßen gewählt: $\Gamma_1 = -\Gamma_2 = \Gamma$, $r_1 = r_2 = r$, $s_1 = s_2 = s$ und $\varphi_1 = -\varphi_2 = \varphi$. Man könnte in diesem Fall erwarten, dass sich die Wirbel schräg zueinander stellen, wie dies zuvor bei den bedingten Mittelwerten in Kapitel 5.3.2 beobachtet wurde.



(a) **(b)**

Abbildung 6.1.: Links: Potential für die Ausrichtung zweier elliptischer Wirbel gleicher Wirbelstärke mit $r \approx 0,6$ und $s \approx 0,8r$. Rechts: Ausgewählter numerisch bestimmter bedingter Mittelwert zum Vergleich (siehe Abbildung 5.3a, Kapitel 5.3.2)

Bewegungsgleichungen

Die Bewegungsgleichungen lauten mit diesen Annahmen:

$$\dot{x} = \frac{\Gamma}{\pi^2 x^3}\left(r^2 - s^2\right)\sin(2\varphi) \tag{6.61a}$$

$$\omega_x = 0 \tag{6.61b}$$

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(r^2\right) = -\Gamma r^2(r^2 - s^2)\left[-\frac{1}{4\pi^2 x^2}\sin(2\varphi) + \frac{6}{4\pi^2 x^4}\sin(4\varphi)\right] + \nu \tag{6.61c}$$

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(s^2\right) = -\Gamma s^2(r^2 - s^2)\left[\frac{1}{4\pi^2 x^2}\sin(2\varphi) - \frac{6}{4\pi^2 x^4}\sin(4\varphi)\right] + \nu \tag{6.61d}$$

$$\dot{\varphi} = -\frac{\Gamma}{2\pi\left(r^2 + s^2\right)} + \Gamma\frac{1}{4\pi^2 x^2}\frac{r^2 + s^2}{r^2 - s^2}\cos(2\varphi) - 6\Gamma\frac{r^2 + s^2}{4\pi^2 x^4}\cos(4\varphi) \tag{6.61e}$$

Stabiler Winkel

Nun soll, wie im Spezialfall gleicher Ausrichtung, ebenfalls die Stabilität unter der Annahme untersucht werden, dass sich Ausdehnung und Abstand der Wirbel nur langsam ändern. Da nun $\omega_x = 0$ ist, vereinfachen sich die Gleichungen ein wenig.

$$\begin{aligned}\dot{\varphi} &= a\cos(2\varphi) + b\cos(4\varphi) + c \\ &= -\frac{\mathrm{d}}{\mathrm{d}\varphi}V(\varphi) \stackrel{!}{=} 0\end{aligned} \tag{6.62}$$

Dabei ist

$$a = \Gamma\frac{r^2+s^2}{r^2-s^2}\frac{1}{4\pi^2x^2} \tag{6.63a}$$

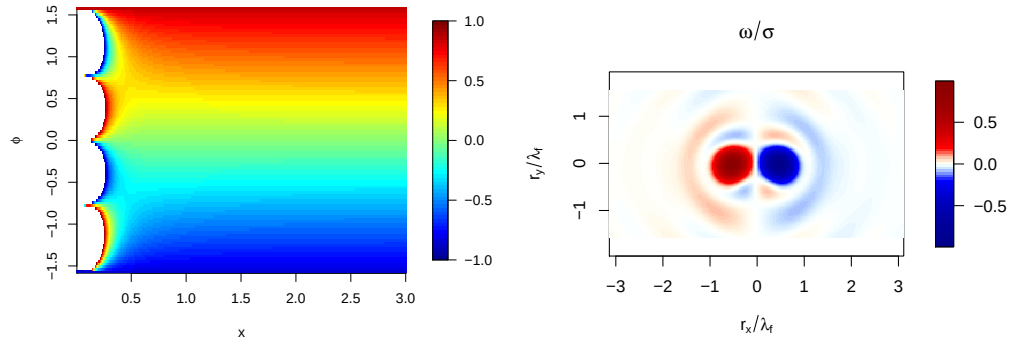
$$b = -6\Gamma\frac{r^2+s^2}{4\pi^2x^4} \tag{6.63b}$$

$$c = -\frac{\Gamma}{2\pi\left(r^2+s^2\right)} \tag{6.63c}$$

Das Potential hierfür ist ebenfalls

$$V(\varphi) = -\frac{a}{2}\sin(2\varphi) - \frac{b}{4}\sin(4\varphi) - c\varphi \tag{6.64}$$

Betrachtet man nun den Fall, den man bei den bedingten Mittelwerten für entgegengesetzte Wirbelstärke in Kapitel 5.3.2, Abbildung 5.4c beobachten konnte. Hier ist $r \approx 0,6$ und $s \approx 0,8r$. Damit ergibt sich ein Potential, wie es in Abbildung 6.1 dargestellt ist.



(a) **(b)**

Abbildung 6.2.: Links: Potential für die Ausrichtung zweier elliptischer Wirbel mit entgegengesetzter Wirbelstärke mit $r \approx 0,6$ und $s \approx 0,8r$. Rechts: Ausgewählter numerisch bestimmter bedingter Mittelwert zum Vergleich (siehe Abbildung 5.4c, Kapitel 5.3.2)

Durch die entgegengesetzten Vorzeichen vertauschen sich nun auch die Vorzeichen der stabilen Winkel im Vergleich zum vorherigen Spezialfall mit Wirbeln gleicher Orientierung. Hierbei ist wie in Kapitel 6.6.4 der Abstand, den die beiden Wirbel zueinander einhalten müssen, damit deren Ausrichtungen stabil zueinander sind, kleiner als erwartet. Eine Korrektur des Wechselwirkungsterms um ca. Faktor 20 würde auch hier das Problem beheben.

6.6.6. Modifikation des Selbstwechselwirkungsterms

Da der Selbstwechselwirkungsterm eine zu starke Winkelgeschwindigkeit verursacht, so dass sich die Wirbel nicht bei der erwarteten Distanz stabilisieren, soll nun noch eine andere Möglichkeit diskutiert werden, die Rotationsgeschwindigkeit des Wirbels zu bestimmen. Man könnte alternativ zum bisherigen Modell den Wirbel als elliptischen Kirchhoff-Wirbel betrachten [Kir83]. Dieser besitzt eine Rotationsgeschwindigkeit von

$$\begin{aligned} \dot{\varphi}_{\text{Kirchhoff}} &= -V_k \frac{r}{s} \frac{1}{(1+\frac{r}{s})^2} \\ &= -V_k \frac{rs}{(r+s)^2} \end{aligned} \tag{6.65}$$

Nun stellt sich die Frage, wie groß die entsprechende Vortizität V_k ist, denn beim Kirchhoff-Modell besitzt der Wirbel überall die gleiche Wirbelstärke. Man könnte hierfür die Wirbelstärke betrachten, bei dem die Amplitude des elliptischen Wirbels auf das $\frac{1}{e}$-fache abgefallen ist. Dann wäre

$$V_k = \frac{\Gamma}{2\pi\sqrt{r^2 s^2}} \mathrm{e}^{-1} \tag{6.66}$$

und somit

$$\dot{\varphi}_{\text{Kirchhoff}} = -\frac{\Gamma}{2\pi e(r+s)^2} \tag{6.67}$$

Setzt man $r \approx 0,6$ und $s \approx 0,8r$ in diese alternative Definition ein, so ist die Rotationsgeschwindigkeit um etwa Faktor 5 langsamer, als mit der bisherigen Definition. Dieses Ergebnis sieht besser aus, reicht aber leider auch noch nicht ganz aus, um bei den gewünschten Distanzen eine stabile Winkelposition zu erhalten.

7. Fazit

Zum Schluss der Arbeit sollen die wichtigsten Ergebnisse der einzelnen Kapitel noch einmal kurz zusammengefasst und abschließend diskutiert werden.

7.1. Simulation

Für diese Arbeit wurde eine parallelisierte Simulation der zweidimensionalen Wirbeltransportgleichung sowohl für den PALMA CPU-Cluster als auch zum Einsatz auf Multi-GPU System bzw. GPU-Clustern neu implementiert. Für beide Implementationen wurde die Performance der Fouriertransformation, der verschiedenen Integrationsschemata und der gesamten Simulation detailliert untersucht.

Von allen getesteten Integrationsverfahren waren sowohl auf dem CPU-Cluster als auch auf dem Multi-GPU-System die Verfahren höchster Ordnung am schnellsten. Dies war sowohl bei eingebetten als auch bei nicht-eingebetteten Verfahren der Fall. Da nur bei den eingebetteten Verfahren der Integrationsfehler automatisch begrenzt werden konnte, wurden nur diese für die Simulation in Betracht gezogen. Für die durchgeführten Simulationen wurde das das Dormand-Prince-Verfahren verwendet, da es von den implementierten Verfahren das schnellste war. Bisherige Turbulenzsimulationen verwenden meist Runge-Kutta-Verfahren 3. Ordnung. Nach den neuen Erkenntnissen sollten allerdings Verfahren höherer Ordnung wegen der deutlich besseren Performance in Erwägung gezogen werden.

Die Parallelisierung der Fouriertransformation auf dem CPU-Cluster wurde von der FFTW-Bibliothek durchgeführt. Diese Implementation skalierte bis zu einem Maximum, das von der Hardware bzw. Konfiguration des Clusters vorgegeben war, nach einem Potenzgesetz.

Für das Multi-GPU-System gab es bisher keine frei verfügbare Bibliothek für verteilte zweidimensionale Fouriertransformationen und somit musste ein eigener Algorithmus entwickelt und implementiert werden, der seinerseits auf die CUDA FFT-Bibliothek zurückgreift. Der Algorithmus zeigte bei bis zu 4 GPUs durchgängig ein negatives Skalierungsverhalten. Dieses entstand durch die hohen Bandbreitenanforderungen im Vergleich zur benötigten Berechnungszeit der FFT. Die verteilten FFT-Implementationen in [CCM10] [NSM12] erzielten deutlich bessere Resultate auf Grund algorithmischer

Vorteile die nur in drei Dimensionen nutzbar sind. Diese verringern den Datenübertragungsaufwand einer Transformation wesentlich.

Für die Simulationsgeschwindigkeit auf dem CPU-Cluster wurde in Abhängigkeit von der Zahl der Prozessorkerne P und der Auflösung N ein Potenzgesetz festgestellt, welches folgende Form annimmt:

$$I(N,P) = C\frac{P^a}{N^2\log(N)}, \quad \text{mit } a = 0,76 \pm 0,047 \tag{7.1}$$

$I(N,P)$ ist dabei die Simulierte Zeit pro Sekunde und die Konstante C ist in Abhängigkeit von der Rechengenauigkeit $C_{\texttt{float}} = (8,60 \pm 0,831) \times 10^6$ bzw. $C_{\texttt{double}} = (6,41 \pm 0,936) \times 10^6$. P^a ist der Speedup gegenüber der Geschwindigkeit eines CPU-Kerns.

Dabei skalierte die Performance der Simulation ebenso wie die der Fouriertransformation bis zu einem Maximum, das durch die Hardware bzw. Konfiguration des Clusters vorgegeben war. Durch Analyse dieser Grenze ließ sich eine optimale Anzahl von Prozessen P_{opt} für die Simulation bestimmen, sowie der daraus folgende maximal möglicher Speedup S_{max} und die maximale Simulationsgeschwindigkeit I_{max}.

$$P_{\text{opt}} = \left(\frac{GN^2\log(N)}{C}\right)^{\frac{1}{a-b}} \qquad S_{\text{max}} = \left(\frac{GN^2\log(N)}{C}\right)^{\frac{a}{a-b}} \tag{7.2}$$

$$I_{\text{max}}(N) = \left(G^a\left(\frac{N^2\log(N)}{C}\right)^b\right)^{\frac{1}{a-b}} \tag{7.3}$$

mit $G = 4003,05 \pm 570,475$ und $b = -1,6 \pm 0,032$. Letztere fällt bei steigender Auflösung exponentiell ab.

Die Abhängigkeit der Simulationsgeschwindigkeit von der Auflösung entsprach $\mathcal{O}(N^2\log(N))$ wie dies von Seiten der FFT her erwartet wurde.

Auf dem Multi-GPU-System erreichte man auf einer GeForce 480 GTX GPU bei 2048×2048 und `float`-Präzision die gut 51-fache Geschwindigkeit gegenüber einem einzelnen CPU-Kern. Damit war man etwa 2,8 mal so schnell wie der PALMA CPU-Cluster bei Maximalleistung. Dieses Ergebnis ist wesentlich besser, als das von [LY12] mit einem ähnlichen Algorithmus erreichte.

Wechselte man von einer GPU auf zwei GPUs, so wurde wegen der vergleichsweise geringen Datenrate des PCIe-Busses die Simulation wesentlich langsamer. Unter Hinzunahme weiterer GPUs sah man ein positives Skalierungsverhalten der Performance

mit einem Exponenten von $a \approx 0,25$. Man erreichte also trotz des negativen Skalierungsverhaltens der FFT eine Beschleunigung. Die Geschwindigkeiten lagen dann nur gut 15% unterhalb der Maximal-Geschwindigkeit des PALMA CPU-Clusters.

Passen die Daten auf eine GPU, so spricht diese Analyse eindeutig für die Verwendung von Grafikkarten zur Simulation von zweidimensionaler Turbulenz. Bei größeren Datenmengen, die nicht mehr auf eine GPU passen, ist der CPU-Cluster von der Simulationsgeschwindigkeit nur geringfügig schneller. Berücksichtigt man zusätzlich noch die Anschaffungs- und Betriebskosten der Hardware, so stellen Rechner mit mehreren GPUs eine günstige Alternative zu großen CPU-Clustern bei adäquater Leistung dar. Moderne Tesla-Hardware, die speziell für den wissenschaftlichen Einsatz konzipiert wurde, besitzt gegenüber gewöhnlichen Grafikkarten einen wesentlich größeren Speicher und bessere Leistung in `double`-Präzision. Mit solchen GPUs dürfte man noch deutlich bessere Ergebnisse erzielen als mit den getesteten GeForce 480 GTX.

Diese durch die Parallelisierung gewonnene Beschleunigung gegenüber der bisherigen Simulation brachte den Vorteil, die Parameter der Simulation wesentlich einfacher und schneller testen und größere Parameterstudien in akzeptabler Zeit durchführen zu können. So werden in Zukunft auch weitere Studien der direkten Kaskade möglich sein, die wesentlich höhere Anforderungen an die Rechenleistung stellen, als bisherige Simulationen der inversen Kaskade.

7.2. Simulationsdaten

Bei der Analyse der Simulationsdaten wurde zunächst festgestellt, dass die Simulation der inversen Kaskade sich relativ robust gegen die Änderung der Simulationsauflösung verhielt, obwohl die räumliche Auflösung der ausgewerteten Daten sehr gering war. Der Exponent der inversen Kaskade betrug statt den erwarteten $-\frac{5}{3}$ eher -2, was aber auch aus anderen Arbeiten [Sco07] bereits bekannt ist und keinerlei Probleme bei der weiteren Auswertung darstellte. Das Energiespektrum besaß bei beiden implementierten Kräften einen langen Inertialbereich und wurde durch die Hyperviskosität nahe der Kraft abgeschnitten.

Die Wahrscheinlichkeitsdichte der Wirbelstärke entsprach eher einer stabilen Verteilung als einer Gaußverteilung, wenn man von den abknickenden Rändern absah, die vermutlich durch die Viskosität verursacht wurden. Diese Tatsache diente später zum Anlass, die bisherige gaußsche Näherung durch alternative Näherungsverfahren zu ersetzen.

Weiterhin wurden mehrere Interpolationsverfahren als alternative Möglichkeiten getestet, um die Auflösung zu verbessern. Von diesen Verfahren wurden einige mit CUDA beschleunigt und brachten dabei erhebliche Geschwindigkeitsvorteile. Die größte

Beschleunigung zeigte die bikubische Interpolation, die auch in [MSW06] zur Interpolation von Teilchentrajektorien eingesetzt wurde. Diese ließ sich unter Verwendung des in [PF05] beschriebenen Algorithmus auf GPUs sehr einfach unter Einbeziehung der Grafikhardware optimieren und erreichte dabei eine Beschleunigung von über 1000 gegenüber der CPU-Implementation auf einem CPU-Kern. Auch die Variante ohne Interpolation war 65 mal schneller, als auf einem CPU-Kern und ermöglichte somit die nötige systematische Auswertung von bedingten Mittelwerten in kürzester Zeit. Im Direktvergleich aller Algorithmen lieferte die Fourierinterpolation das beste Ergebnis.

7.3. Entwicklungsgleichungen, Wahrscheinlichkeitsverteilungen und bedingte Mittelwerte

Zur Analyse der auf zwei Punkte bedingten Mittelwerte wurden zunächst die Ergebnisse von [Fri+10] reproduziert. Die schlechte Approximation der gaußschen Näherung war Anlass, hier genauere Untersuchungen durchzuführen und ein neues Modell für die 1- bzw. 2-Punkt Wahrscheinlichkeitsdichte zu erstellen. Dabei wurde systematisch untersucht, wie sich unterschiedliche Kraftparameter auf die 1-Punkt-Verteilung auswirken. Diese bekommen bei größeren Kraftstärken immer weitere Flügel und ähneln dabei stabilen Verteilungen.

Zunächst wurde ein allgemeiner Polynomansatz verfolgt, um den Exponenten der charakteristischen Funktion der Wahrscheinlichkeitsverteilungen zu approximieren. Unter der Einschränkung, dass die 1-Punkt Verteilung noch gaußförmig sein sollte, tauchten hier Hermite-Polynome auf, mit deren Hilfe man die Wahrscheinlichkeitsverteilung aus den Momenten und damit auch die bedingten Mittelwerte nähern konnte. Dieses Verfahren schien nicht oder zumindest nur sehr langsam zu konvergieren und reproduzierte die Ergebnisse nur näherungsweise bei kleineren Kraftamplituden.

Die Verwendung von stabilen Verteilungen zeigte bei der Näherung der bedingten Mittelwerte gegenüber der Gaußverteilung keine wesentlichen Vorteile. Dieses Problem löste die Faltung verschiedener stabiler Verteilungen. Innerhalb dieses Modells gelang es, die numerischen Ergebnisse des auf einen Punkt bedingten Mittelwertes gut zu approximieren. Eine der beiden Verteilungen schien gaußförmig zu sein, denn nur dann oszillierte deren Parameterfunktion und die der anderen Verteilung nicht. Man könnte die gaußsche Verteilung mit der äußeren Kraft, welche die Turbulenz antreibt, in Verbindung bringen, denn diese erzeugt ohne Wirbeltransport ebenfalls eine gaußsche Verteilung des Wirbelstärkefeldes. Da eine Faltung von Verteilungen einer Summation von Zufallsvariablen entspricht, deutet alles darauf hin, dass man die Kraft hier als additive Zufallsgröße betrachten kann.

Eine offene Frage ist, wie man die zwei Parameterfunktionen der stabilen Verteilungen bestimmt, die bei den bedingten Mittelwerten nur als Produkt zu berechnen sind.

Eine Möglichkeit bietet eventuell der zweite bedingte Moment. Um diesen mit stabilen Verteilungen zu nähern, müssen zunächst die Ränder der Verteilung beschrieben werden, denn diese sorgen innerhalb der Näherung dafür, dass dieser Moment existiert. Diese Beschreibung kann zum Beispiel durch Multiplikation mit einer weiteren Gaußverteilung erfolgen, welche vermutlich über die Viskositäten zu parametrisieren wäre.

Um den auf zwei Punkte bedingten Mittelwert für nah benachbarte Punkte zu nähern, wurde noch eine kurze Rechnung für elliptisch konturierte Verteilungen durchgeführt. An diese Rechnung schließt sich die Frage an, wie gut diese Näherung für nah benachbarte Punkte gilt und wie man dieses Modell erweitern müsste, um für weit entfernte Punkte statistische Unabhängigkeit zu erhalten und den numerisch beobachteten Symmetriebruch zu erfassen.

7.4. Elliptisch-gaußförmige Wirbel

Um die elliptische Verformung und den Symmetriebruch der auf zwei Punkte bedingten mittleren Wirbelstärkefelder genauer zu verstehen, wurden ähnlich zum Ansatz in [FF11] über die Wirbeltransportgleichung im Fourierraum Bewegungsgleichungen für elliptisch-gaußförmige Wirbel hergeleitet. Hier gelang es, ein modifiziertes Punktwirbelmodell zu erstellen, bei dem die Wirbel sich abhängig von ihrer Ausdehnung und Orientierung zueinander anziehen oder abstoßen. Dies ist ein Effekt, der im Punktwirbelmodell nicht beschrieben werden kann. Die Wirbel in diesem neuen Modell verhalten sich im Grenzfall runder Wirbel wie Lamb-Oseen-Wirbel und im Grenzfall verschwindender Ausdehnung wie Punktwirbel. Die Orientierung der elliptischen Wirbel wurde auf Stabilität geprüft und es gelang auch, stabile Winkel zu finden, die mit Simulationen und Experimenten konsistent sind. Ebenfalls ergab sich eine Übereinstimmung mit der Verkippung der bedingten Mittelwerte. Der für die Stabilität notwendige Abstand der beiden Wirbel war etwas gering und lag bei den gewählten Parametern nur etwa bei der Ausdehnung eines Wirbels. Durch eine etwas andere Modellierung des Selbstwechselwirkungsterms könnte man eventuell diesen Abstand erhöhen und somit die Verschmelzung von Wirbeln komplett beschreiben.

A. Simulationsparameter

In Tabelle A.1 befinden sich die Simulationsparameter der einzelnen Simulationen. Die simulierte Feldgröße $L_x \times L_y$ ist für alle Simulationen gleich $2\pi \times 2\pi$.

Name	N_x	N_y	f_A	k_f	Kraft	k_A	γ	μ	k_μ	m	ν	k_ν	n
`4096inverse`	4096	4096	*	100	*	2	0	450	1	2	450	170.667	8
`4096both`	4096	4096	$4\pi^2 \cdot 10^0$	200	det	2	0	450	1	2	450	$1365,333$	8
`resfield`	*	*	1	100	stoch	2	0	450	1	2	450	170.667	8
`resforce`	*	*	*	100	*	2	0	450	1	2	450	170.667	8
`distcond`	4096	4096	$4\pi^2 \cdot 10^-2$	100	stoch	2	0	450	1	2	450	170.667	8
`integration`	512	512	$4\pi^2 \cdot 10^1$	100	det	2	0	450	1	2	450	170.667	8

Tabelle A.1.: Simulationsparameter der Simulationen. Felder mit * haben verschiedene Werte angenommen und werden in der jeweiligen Auswertung ergänzt.

B. Daten der Fits

In den nachstehenden Tabellen befinden sich die Werte der Ausgleichsrechnungen inklusive Fehlerangaben, die zur Bestimmung des Skalierungsverhaltens der FFT und der Simulation durchgeführt wurden.

Auflösung	Exponent
512	0,78 ± 0,029
1024	0,6 ± 0,028
2048	0,5 ± 0,028
4096	0,58 ± 0,023
8192	0,73 ± 0,06

Tabelle B.1.: Fit-Werte für den Exponenten im Performance-Fit der CPU-FFT (Präzision: `float`, Bibliothek: OpenMPI)

Auflösung	Bibliothek	Präzision	Exponent
512	OpenMPI	`float`	0,92 ± 0,012
1024	OpenMPI	`float`	0,82 ± 0,012
2048	OpenMPI	`float`	0,75 ± 0,023
4096	OpenMPI	`float`	0,73 ± 0,015
512	OpenMPI	`double`	0,81 ± 0,012
1024	OpenMPI	`double`	0,74 ± 0,017
2048	OpenMPI	`double`	0,64 ± 0,023
4096	OpenMPI	`double`	0,7 ± 0,015
512	MVAPICH2	`float`	0,85 ± 0,023
1024	MVAPICH2	`float`	0,83 ± 0,022
2048	MVAPICH2	`float`	0,83 ± 0,051
512	MVAPICH2	`double`	0,82 ± 0,019
1024	MVAPICH2	`double`	0,85 ± 0,029
2048	MVAPICH2	`double`	0,96 ± 0,061

Tabelle B.2.: Fit-Werte für den Exponenten im Performance-Fit der CPU-Simulation

Literaturverzeichnis

[Arg+10] J. Argyris u. a. *Die Erforschung des Chaos*. 2. Auflage. Vieweg Braunschweig, 2010.

[Arm67] B.H. Armstrong. "Spectrum line profiles: The Voigt function". In: *Journal of Quantitative Spectroscopy and Radiative Transfer* 7.1 (1967), S. 61–88. ISSN: 0022-4073. DOI: 10.1016/0022-4073(67)90057-X. URL: http://www.sciencedirect.com/science/article/pii/002240736790057X.

[Bat00] G.K. Batchelor. *An introduction to fluid dynamics*. Cambridge university press, 2000.

[BCD02] G. Boffetta, M. Cencini und J. Davoudi. "Closure of two-dimensional turbulence: The role of pressure gradients". In: *Phys. Rev. E* 66 (1 Juli 2002), S. 017301. DOI: 10.1103/PhysRevE.66.017301. URL: http://link.aps.org/doi/10.1103/PhysRevE.66.017301.

[BE12] Guido Boffetta und Robert E. Ecke. "Two-Dimensional Turbulence". In: *Annual Review of Fluid Mechanics* 44.1 (2012), S. 427–451. DOI: 10.1146/annurev-fluid-120710-101240. eprint: http://www.annualreviews.org/doi/pdf/10.1146/annurev-fluid-120710-101240. URL: http://www.annualreviews.org/doi/abs/10.1146/annurev-fluid-120710-101240.

[BM10] G. Boffetta und S. Musacchio. "Evidence for the double cascade scenario in two-dimensional turbulence". In: *Phys. Rev. E* 82 (1 Juli 2010), S. 016307. DOI: 10.1103/PhysRevE.82.016307. URL: http://link.aps.org/doi/10.1103/PhysRevE.82.016307.

[Boo] Boost. *Boost C++ Libraries*. URL: http://www.boost.org.

[Bot12] M.M. Botezatu. "A study on compiler flags and performance events". In: (2012).

[BSW07] Håvard Berland, Bård Skaflestad und Will M. Wright. "EXPINT—A MATLAB package for exponential integrators". In: *ACM Trans. Math. Softw.* 33.1 (März 2007). ISSN: 0098-3500. DOI: 10.1145/1206040.1206044. URL: http://doi.acm.org/10.1145/1206040.1206044.

[BW03] J.C. Butcher und J. Wiley. *Numerical methods for ordinary differential equations*. Bd. 2. Wiley Online Library, 2003.

[CCM10] Y. Chen, X. Cui und H. Mei. "Large-scale FFT on GPU clusters". In: *Proceedings of the 24th ACM International Conference on Supercomputing*. ACM. 2010, S. 315–324.

[CFL28] R. Courant, K. Friedrichs und H. Lewy. "Über die partiellen Differenzengleichungen der mathematischen Physik". German. In: *Mathematische Annalen* 100 (1 1928), S. 32–74. ISSN: 0025-5831. DOI: 10.1007/BF01448839. URL: http://dx.doi.org/10.1007/BF01448839.

[Cha08] John M. Chambers. *Software for Data Analysis: Programming with R*. New York: Springer, 2008. ISBN: 978-0-387-75935-7. URL: http://stat.stanford.edu/~jmc4/Rbook/.

[CJW06] J.A. Carlson, A. Jaffe und A. Wiles. *The millennium prize problems*. Amer Mathematical Society, 2006.

[CK90] J. R. Cash und Alan H. Karp. "A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides". In: *ACM Trans. Math. Softw.* 16.3 (Sep. 1990), S. 201–222. ISSN: 0098-3500. DOI: 10.1145/79505.79507. URL: http://doi.acm.org/10.1145/79505.79507.

[CM02] S.M. Cox und P.C. Matthews. “Exponential Time Differencing for Stiff Systems”. In: *Journal of Computational Physics* 176.2 (2002), S. 430–455. ISSN: 0021-9991. DOI: 10.1006/jcph.2002.6995. URL: http://www.sciencedirect.com/science/article/pii/S0021999102969950.

[CT65] J.W. Cooley und J.W. Tukey. “An algorithm for the machine calculation of complex Fourier series”. In: *Mathematics of computation* (1965), S. 297–301.

[Dan+10] A. Danalis u. a. “The scalable heterogeneous computing (SHOC) benchmark suite”. In: *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*. ACM. 2010, S. 63–74.

[DP80] J.R. Dormand und P.J. Prince. “A family of embedded Runge-Kutta formulae”. In: *Journal of Computational and Applied Mathematics* 6.1 (1980), S. 19–26. ISSN: 0377-0427. DOI: 10.1016/0771-050X(80)90013-3. URL: http://www.sciencedirect.com/science/article/pii/0771050X80900133.

[Eig] Eigen. *Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms*. URL: http://eigen.tuxfamily.org.

[Fal94] Gregory Falkovich. “Bottleneck phenomenon in developed turbulence”. In: *Physics of Fluids* 6.4 (1994), S. 1411–1414. DOI: 10.1063/1.868255. URL: http://link.aip.org/link/?PHF/6/1411/1.

[FCY99] M. Folk, A. Cheng und K. Yates. “HDF5: A File Format and I/O Library for High Performance Computing Applications”. In: *Proc. Supercomputing 1999*. Portland, OR, 1999. URL: http://www.hdfgroup.org/HDF5/.

[Feh70] E. Fehlberg. “Klassische Runge-Kutta-Formeln vierter und niedrigerer Ordnung mit Schrittweiten-Kontrolle und ihre Anwendung auf Wärmeleitungsprobleme”. In: *Computing* 6 (1 1970). 10.1007/BF02241732, S. 61–71. ISSN: 0010-485X. URL: http://dx.doi.org/10.1007/BF02241732.

[FF11] J. Friedrich und R. Friedrich. “Vortex-Model for the Inverse Cascade of 2D-Turbulence”. In: *arXiv preprint arXiv:1111.5808* (2011).

[FFM] FFMpeg. *FFMpeg*. URL: http://ffmpeg.org.

[FJ05] M. Frigo und S. G. Johnson. “The Design and Implementation of FFTW3”. In: *Proceedings of the IEEE* 93.2 (Feb. 2005), S. 216–231. ISSN: 0018-9219. DOI: 10.1109/JPROC.2004.840301. URL: http://www.fftw.org/fftw-paper-ieee.pdf.

[Fri+08] Uriel Frisch u. a. “Hyperviscosity, Galerkin Truncation, and Bottlenecks in Turbulence”. In: *Phys. Rev. Lett.* 101 (14 Sep. 2008), S. 144501. DOI: 10.1103/PhysRevLett.101.144501. URL: http://link.aps.org/doi/10.1103/PhysRevLett.101.144501.

[Fri+10] R. Friedrich u. a. “Two-Point Vorticity Statistics in the Inverse Turbulent Cascade”. In: *ArXiv e-prints* (Dez. 2010). arXiv: 1012.3356 [physics.flu-dyn].

[Gab+04] Edgar Gabriel u. a. “Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation”. In: *Proceedings, 11th European PVM/MPI Users’ Group Meeting*. Budapest, Hungary, Sep. 2004, S. 97–104.

[Gea71] C. William Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1971. ISBN: 0136266061.

[GLS99] W. Gropp, E. Lusk und A. Skjellum. *Using MPI: portable parallel programming with the message passing interface*. Bd. 1. MIT press, 1999.

[HH79] Edwin Hewitt und Robert Hewitt. “The Gibbs-Wilbraham phenomenon: An episode in fourier analysis”. In: *Archive for History of Exact Sciences* 21 (2 1979). 10.1007/BF00330404, S. 129–160. ISSN: 0003-9519. URL: http://dx.doi.org/10.1007/BF00330404.

[HNW93] E. Hairer, S.P. Nørsett und G. Wanner. *Solving ordinary differential equations: Nonstiff problems*. Bd. 1. Springer Verlag, 1993.

[HO10] Marlis Hochbruck und Alexander Ostermann. "Exponential integrators". In: *Acta Numerica* 19 (2010), S. 209–286. DOI: 10.1017/S0962492910000048. URL: http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=7701740&fulltextType=RA&fileId=S0962492910000048.

[Hua+06] W. Huang u. a. "Design of high performance MVAPICH2: MPI2 over InfiniBand". In: *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on*. Bd. 1. IEEE. 2006, S. 43–48.

[Ji+12] F. Ji u. a. "Efficient intranode communication in GPU-accelerated systems". In: *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*. IEEE. 2012, S. 1838–1847.

[KC12] M. Magcalas K. Spyksma und N. Campbell. "Quantifying effects of hyperviscosity on isotropic turbulence". In: *Phys. Fluids* submitted (2012). eprint: http://cs.redeemer.ca/kspyksma/papers/Spyksma_hyperviscosity2012.pdf. URL: http://cs.redeemer.ca/kspyksma/index.html.

[KG02] Hamid Kellay und Walter I Goldburg. "Two-dimensional turbulence: a review of some recent experiments". In: *Reports on Progress in Physics* 65.5 (2002), S. 845. URL: http://stacks.iop.org/0034-4885/65/i=5/a=204.

[KH38] Theodore de Kármán und Leslie Howarth. "On the Statistical Theory of Isotropic Turbulence". English. In: *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* 164.917 (1938), ISSN: 00804630. URL: http://www.jstor.org/stable/97087.

[Kir07] David Kirk. "NVIDIA cuda software and gpu parallel computing architecture". In: *Proceedings of the 6th international symposium on Memory management*. ISMM '07. Montreal, Quebec, Canada: ACM, 2007, S. 103–104. ISBN: 978-1-59593-893-0. DOI: 10.1145/1296907.1296909. URL: http://kr.nvidia.com/content/cudazone/download/showcase/kr/Tutorial-DKIRK.pdf.

[Kir83] G. Kirchhoff. *Vorlesungen über mathematische Physik*. BG Teubner, 1883.

[Kol41a] A. N. Kolmogorov. "Dissipation of energy in locally isotropic turbulence [In Russian]". In: *Dokl. Akad. Nauk SSSR* 32 (1941), S. 19–21.

[Kol41b] A. N. Kolmogorov. "On degeneration of isotropic turbulence in an incompressible viscous liquid". In: *Dokl. Akad. Nauk SSSR*. Bd. 31. 1941, S. 538–540.

[Kol41c] A. N. Kolmogorov. "The Local Structure of Turbulence in Incompressible Viscous Fluid for Very Large Reynolds' Numbers". In: *Akademiia Nauk SSSR Doklady* 30 (1941), S. 301–305.

[Kor+12] P. Korošec u. a. "Multi-core implementation of the differential ant-stigmergy algorithm for numerical optimization". In: *The Journal of Supercomputing* (2012), S. 1–16.

[Kra67] Robert H. Kraichnan. "Inertial Ranges in Two-Dimensional Turbulence". In: *Physics of Fluids* 10.7 (1967), S. 1417–1423. DOI: 10.1063/1.1762301. URL: http://link.aip.org/link/?PFL/10/1417/1.

[Kra71] Robert H. Kraichnan. "Inertial-range transfer in two- and three-dimensional turbulence". In: *Journal of Fluid Mechanics* 47.03 (1971), S. 525–535. DOI: 10.1017/S0022112071001216. eprint: http://journals.cambridge.org/article_S0022112071001216. URL: http://dx.doi.org/10.1017/S0022112071001216.

[Kro05] S. Krogstad. "Generalized integrating factor methods for stiff PDEs". In: *Journal of Computational Physics* 203.1 (2005), S. 72–88. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2004.08.006. URL: http://www.sciencedirect.com/science/article/pii/S0021999104003183.

[KT05] A. Kassam und L. Trefethen. “Fourth-Order Time-Stepping for Stiff PDEs”. In: *SIAM Journal on Scientific Computing* 26.4 (2005), S. 1214–1233. DOI: 10.1137/S1064827502410633. eprint: http://epubs.siam.org/doi/pdf/10.1137/S1064827502410633. URL: http://epubs.siam.org/doi/abs/10.1137/S1064827502410633.

[Kut01] W. Kutta. *Beitrag zur näherungsweisen Integration totaler Differentialgleichungen*. Bd. 46. 1901, S. 435–453.

[Lam91] J.D. Lambert. *Numerical methods for ordinary differential systems: the initial value problem*. John Wiley & Sons, Inc., 1991.

[Law67] J. Douglas Lawson. “Generalized Runge-Kutta Processes for Stable Systems with Large Lipschitz Constants”. English. In: *SIAM Journal on Numerical Analysis* 4.3 (1967), ISSN: 00361429. URL: http://www.jstor.org/stable/2949405.

[LCP05] A. G. Lamorgese, D. A. Caughey und S. B. Pope. “Direct numerical simulation of homogeneous turbulence with hyperviscosity”. In: *Physics of Fluids* 17.1, 015106 (2005), S. 015106. DOI: 10.1063/1.1833415. URL: http://link.aip.org/link/?PHF/17/015106/1.

[Li+11] Yan Li u. a. “Automatic FFT Performance Tuning on OpenCL GPUs”. In: *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*. Dez. 2011, S. 228–235. DOI: 10.1109/ICPADS.2011.32.

[Lil69] Douglas K. Lilly. “Numerical Simulation of Two-Dimensional Turbulence”. In: *Physics of Fluids* 12.12 (1969), DOI: 10.1063/1.1692444. URL: http://link.aip.org/link/?PFL/12/II-240/1.

[Luk60] E. Lukacs. *Characteristic functions*. Bd. 4. Griffin London, 1960.

[Lun67] T. S. Lundgren. “Distribution Functions in the Statistical Theory of Turbulence”. In: *Physics of Fluids* 10.5 (1967), S. 969–975. DOI: 10.1063/1.1762249. URL: http://link.aip.org/link/?PFL/10/969/1.

[LY12] Kaiyuan Lou und Zhaohua Yin. “A CUDA Pseudo-spectral Solver for Two-Dimensional Navier-Stokes Equation”. In: *Distributed Computing and Applications to Business, Engineering Science (DCABES), 2012 11th International Symposium on*. Okt. 2012, S. 62–66. DOI: 10.1109/DCABES.2012.93.

[MGM11] A. Munshi, B. Gaster und T.G. Mattson. *OpenCL programming guide*. Addison-Wesley Professional, 2011.

[Mic97] H. Michels. “DISLIN Manual”. In: *Max-Planck-Institut fur Aeronomie, Katlenburg-Lindau* (1997). URL: http://www.mps.mpg.de/dislin/.

[MK97] P. Moin und J. Kim. “Tackling turbulence with supercomputers”. In: *Scientific American* 276.1 (1997), S. 46–52.

[MN88] D.P. Mitchell und A.N. Netravali. “Reconstruction filters in computer graphics”. In: *Computer Graphics* 22.4 (1988), S. 221–228.

[Mol09] Ilya Molchanov. “Convex and star-shaped sets associated with multivariate stable distributions, I: Moments and densities”. In: *Journal of Multivariate Analysis* 100.10 (2009), S. 2195–2213. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2009.04.003. URL: http://www.sciencedirect.com/science/article/pii/S0047259X09000864.

[Mon67] A. S. Monin. “Equations of turbulent motion”. In: *Journal of Applied Mathematics and Mechanics* 31.6 (1967), S. 1057–1068. ISSN: 0021-8928. DOI: 10.1016/0021-8928(67)90210-9. URL: http://www.sciencedirect.com/science/article/pii/0021892867902109.

[MSW06] Ana M. Mancho, Des Small und Stephen Wiggins. “A comparison of methods for interpolating chaotic flows from discrete velocity data”. In: *Computers & Fluids* 35.4 (2006), S. 416–428. ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2005.02.003. URL: http://www.sciencedirect.com/science/article/pii/S004579300500040X.

[MW05] Borislav V. Minchev und Will M. Wright. “A review of exponential integrators for first order semi-linear problems”. In: *Preprint Numerics* 2 (2005). URL: http://www.ii.uib.no/%7Eborko/pub/N2-2005.pdf.

[MZM88] M. V. Melander, N. J. Zabusky und J. C. Mcwilliams. “Symmetric vortex merger in two dimensions: causes and conditions”. In: *Journal of Fluid Mechanics* 195 (1988), S. 303–340. DOI: 10.1017/S0022112088002435. eprint: http://journals.cambridge.org/article_S0022112088002435. URL: http://dx.doi.org/10.1017/S0022112088002435.

[Nav23] C. Navier. “Mémoire sur les lois du mouvement des fluides”. In: *Mémoires de l’Académie Royale des Sciences de l’Institut de France* 6 (1823), S. 389–440.

[NMH06] K. Nitadori, J. Makino und P. Hut. “Performance tuning of< i> N</i>-body codes on modern microprocessors: I. Direct integration with a hermite scheme on x86_64 architecture”. In: *New Astronomy* 12.3 (2006), S. 169–181.

[NMM12] Akira Nukada, Yutaka Maruyama und Satoshi Matsuoka. “High performance 3-D FFT using multiple CUDA GPUs”. In: *Proceedings of the 5th Annual Workshop on General Purpose Processing with Graphics Processing Units*. GPGPU-5. London, England: ACM, 2012, S. 57–63. ISBN: 978-1-4503-1233-2. DOI: 10.1145/2159430.2159437. URL: http://doi.acm.org/10.1145/2159430.2159437.

[Nol12] J. P. Nolan. *Stable Distributions - Models for Heavy Tailed Data*. In progress, Chapter 1 online at academic2.american.edu/∼jpnolan. Boston: Birkhauser, 2012.

[Nov68] E. A. Novikov. “Kinetic equations for a vortex field”. In: *Soviet Physics Doklady*. Bd. 12. 11. 1968, S. 1006–1008.

[NSM12] A. Nukada, K. Sato und S. Matsuoka. “Scalable Multi-GPU 3-D FFT for TSUBAME 2.0 Supercomputer”. In: *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press. 2012, S. 44.

[NVIa] NVIDIA. *CUDA C Programming Guide*. URL: http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html.

[NVIb] NVIDIA. *cuFFT User Guide*. URL: http://docs.nvidia.com/cuda/cufft/index.html.

[NVIc] NVIDIA. *NVIDIA GPUDirect*. URL: https://developer.nvidia.com/gpudirect.

[NVId] NVIDIA. *Why choose Tesla?* URL: http://www.nvidia.com/object/why-choose-tesla.html.

[OP72] S.A. Orszag und GS Patterson Jr. “Numerical simulation of three-dimensional homogeneous isotropic turbulence”. In: *Physical Review Letters* 28.2 (1972), S. 76–79.

[PD81] P.J. Prince und J.R. Dormand. “High order embedded Runge-Kutta formulae”. In: *Journal of Computational and Applied Mathematics* 7.1 (1981), S. 67–75. ISSN: 0377-0427. DOI: 10.1016/0771-050X(81)90010-3. URL: http://www.sciencedirect.com/science/article/pii/0771050X81900103.

[PF05] M. Pharr und R. Fernando. *GPU Gems 2: Programming Techniques For High-Performance Graphics And General-Purpose Computation*. Addison-Wesley Professional, 2005.

[Pop00] S.B. Pope. *Turbulent flows*. Cambridge university press, 2000.

[Pre+86] W.H. Press u. a. *Numerical recipes*. Bd. 547. Cambridge Univ Press, 1986.

[Run95] C. Runge. “Ueber die numerische Auflösung von Differentialgleichungen”. In: *Mathematische Annalen* 46 (2 1895). 10.1007/BF01446807, S. 167–178. ISSN: 0025-5831. URL: http://dx.doi.org/10.1007/BF01446807.

[Saf93] P.G. Saffman. *Vortex dynamics*. Cambridge University Press, 1993.

[Sco07] R. K. Scott. “Nonrobustness of the two-dimensional turbulent inverse cascade”. In: *Phys. Rev. E* 75 (4 Apr. 2007), S. 046301. DOI: 10.1103/PhysRevE.75.046301. URL: http://link.aps.org/doi/10.1103/PhysRevE.75.046301.

[Sin+11] A.K. Singh u. a. “MPI alltoall personalized exchange on GPGPU clusters: Design alternatives and benefit”. In: *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*. IEEE. 2011, S. 420–427.

[Sto51] G.G. Stokes. “On the effect of the internal friction of fluids on the motion of pendulums”. In: (1851).

[Tay35] G. I. Taylor. “Statistical Theory of Turbulence”. English. In: *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* 151.873 (1935), ISSN: 00804630. URL: http://www.jstor.org/stable/96557.

[Wan+11] H. Wang u. a. “MVAPICH2-GPU: optimized GPU to GPU communication for InfiniBand clusters”. In: *Computer Science-Research and Development* 26.3 (2011), S. 257–266.

Erklärung zur Diplomarbeit

Hiermit versichere ich, diese Arbeit selbständig angefertigt und, außer den angegebenen, keine weiteren Hilfsmittel verwendet zu haben.

Münster, Januar 2013