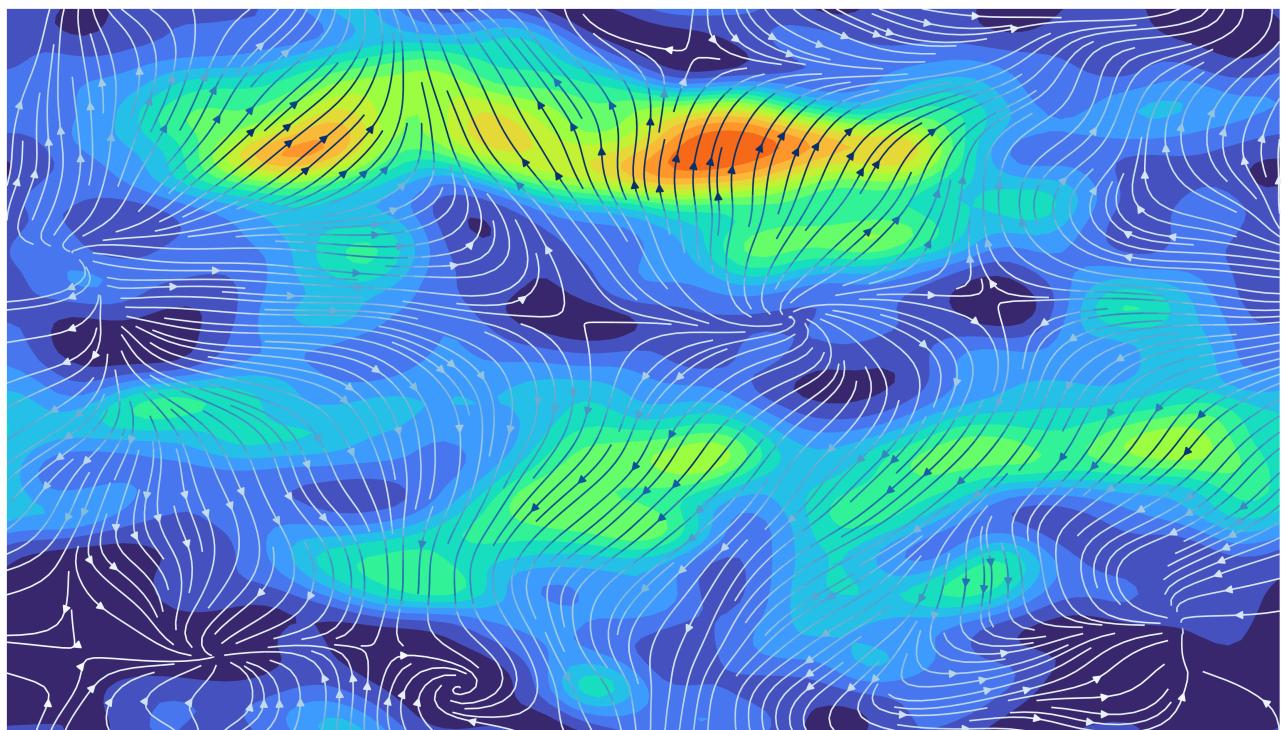


Universität Freiburg  
Projekt zum Bestehen des Kurses:  
„Spektrale Simulationsmethoden mit Python“ SoSe 2024  
Lars Pastewka, Andreas Greiner, Martin Ladecky

# Lösen der Navier-Stokes-Gleichung unter Verwendung Pseudo-Spektraler-Methoden

Johann Gockel

13. September 2024



# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>3</b>
<b>2 Pseudo-Spektrale-Methoden</b>	<b>3</b>
<b>3 Die Navier-Stokes-Gleichung</b>	<b>4</b>
3.1 Beschreibung der Navier-Stokes-Gleichung . . . . .	4
3.2 Fourier-Darstellung der Navier-Stokes-Gleichung . . . . .	5
<b>4 Numerische Berechnung der Navier-Stokes-Gleichung</b>	<b>5</b>
4.1 Implementierung des Lösungsverfahrens in Python . . . . .	5
4.2 Aliasing und De-Aliasing . . . . .	6
4.3 Testen des Algorithmus am Taylor-Green-Vortex . . . . .	7
4.3.1 Der Taylor-Green-Vortex und dessen analytische Zeitentwicklung . . . . .	7
4.3.2 Das Vergleichsverfahren . . . . .	8
4.3.3 Vergleich der numerischen und analytischen Zeitentwicklung . . . . .	9
4.3.4 Stabilität des Algorithmus . . . . .	10
4.4 Anwendungsbeispiel: Turbulenzen . . . . .	11
4.4.1 Das getriebene Geschwindigkeitsfeld . . . . .	11
4.4.2 Das Energie- und Dissipationsspektrum . . . . .	14
<b>5 Anhang</b>	<b>16</b>
5.1 Abbildungen . . . . .	16

# 1 Einleitung

Die Navier-Stokes-Gleichung ist eine grundlegende Gleichung der Fluidodynamik, die das Verhalten von Fluiden wie Luft und Wasser beschreibt. Sie ist daher von zentraler Bedeutung für viele Bereiche der Physik, einschließlich der Aerodynamik, der Astrodynamik, der Magnetodynamik und der Meteorologie. Aufgrund ihrer Komplexität sind analytische Lösungen für die Navier-Stokes-Gleichung nur in wenigen speziellen Fällen möglich (siehe z.B. „Taylor-Green-Vortex“). Tatsächlich zählt der Beweis der Existenz und Eindeutigkeit von analytischen Lösungen zu den ungelösten Millennium-Problemen der Mathematik [7]. Daher werden numerische Methoden verwendet, um spezifische Probleme zu lösen. In diesem Projekt soll ein Algorithmus in Python entwickelt werden, der eine numerische Lösung der Navier-Stokes-Gleichung, mithilfe Pseudo-Spektraler-Methoden, berechnet. Dieser soll dann an Beispielen getestet werden. Das Ziel ist, ein robustes Python-Programm zu entwickeln, das in der Lage ist, die dreidimensionale, inkompressible Navier-Stokes-Gleichung für eine Vielzahl an Situationen zu lösen. Der vorliegende Bericht soll, zusammen mit dem beigelegten Code, durch die Schritte der Entwicklung führen. Dazu werde ich die verwendeten Methoden beschreiben und anschließend auf die Navier-Stokes-Gleichung anwenden. Dann beschreibe ich die Implementierung des Lösungscodes und teste ihn anhand von zwei verschiedenen Beispielen. Diese sind der Taylor-Green-Vortex, der verwendet wird, um die Qualität und Stabilität des Codes zu testen und ein zufällig generiertes Geschwindigkeitsfeld mit einem vorgegebenen Energiespektrum. Zum Schluss soll eine Methode getestet werden, um das zufällig generierte Geschwindigkeitsfeld zur Erzeugung von Turbulenzen anzutreiben („Forcing“).

## 2 Pseudo-Spektrale-Methoden

Pseudo-Spektrale-Methoden werden dazu verwendet, um partielle Differentialgleichungen numerisch zu lösen. Die Wirksamkeit der Methoden beruht auf der Eigenschaft der Fouriertransformation, Ableitungen so zu vereinfachen, dass sie numerisch leichter zu berechnen sind.

$$\mathcal{F} \left[ \frac{\partial f(x, t)}{\partial x} \right] (q, t) = iq\mathcal{F}(f(x, t)) \quad (1)$$

Dabei ist  $i$  die imaginäre Einheit und  $q$  die normierte Wellenzahl. Es bietet sich somit an, die Berechnung von Differentialgleichungen im Fourier- bzw. Frequenzraum durchzuführen, da sich die Differentiation zu einer Multiplikation vereinfacht. Die Transformation in den Frequenzraum entwickelt das ursprüngliche Feld  $\vec{f}$  in einer Basis, die auf Wellenfunktionen, d.h. sin und cos, beruht. Daher stammt die Bezeichnung der Spektralen Methode. Allerdings nur Pseudo-Spektral, da das Ziel dennoch ist, Ergebnisse im Konfigurationsraum zu erhalten, weshalb die Berechnungen auf einem dreidimensionalen, kartesischen Gitter durchgeführt werden. Die Operation in Gleichung (1) kann auf dreidimensionale Probleme erweitert werden. Es lassen sich somit die wichtigsten Differentialoperatoren, d.h. die Divergenz, die Rotation, der Gradient und der Laplace-Operator leicht im Fourierraum darstellen.

$$\mathcal{F} \left[ \nabla \cdot \vec{f}(\vec{x}, t) \right] (\vec{q}, t) = i\vec{q} \cdot \mathcal{F}(\vec{f}(\vec{x}, t)) \quad (2)$$

$$\mathcal{F} \left[ \nabla \times \vec{f}(\vec{x}, t) \right] (\vec{q}, t) = i\vec{q} \times \mathcal{F}(\vec{f}(\vec{x}, t)) \quad (3)$$

$$\mathcal{F} [\nabla g(\vec{x}, t)] (\vec{q}, t) = i\vec{q} \mathcal{F}(g(\vec{x}, t)) \quad (4)$$

$$\mathcal{F} [\Delta \vec{f}(\vec{x}, t)] (\vec{q}, t) = -|\vec{q}|^2 \mathcal{F}(\vec{f}(\vec{x}, t)) \quad (5)$$

Damit erhält man die Werkzeuge, um die Navier-Stokes-Gleichung in den Fourierraum zu transformieren. Allerdings müssen später bei den numerischen Berechnungen gewisse Regeln, die durch die Periodizität der Basisfunktionen des Fourierraums entstehen, eingehalten werden. Zum Beispiel ist darauf zu achten, dass die physikalische Größe des Gebiets, in dem die Berechnungen durchgeführt werden,  $\pi$ -periodisch ist. Ist dies nicht der Fall, entstehen große numerische Fehler, die durch die obigen Differentialoperationen verstärkt werden. Dies gilt besonders für iterative Verfahren wie die, die hier bei der Zeitentwicklung des Geschwindigkeitsfelds verwendet wurden. Auch das Aliasing ist eine methodenspezifische Unsicherheit, auf die zu achten ist. Darauf gehe ich in Abschnitt 4.2 näher ein.

### 3 Die Navier-Stokes-Gleichung

#### 3.1 Beschreibung der Navier-Stokes-Gleichung

Die Navier-Stokes-Gleichung (NSG) ist eine partielle Differentialgleichung, die die räumliche und zeitliche Entwicklung von viskosen, newton'schen Fluiden makroskopisch beschreibt. Die Bezeichnung als newton'sch stammt daher, dass die Gleichung über das Aufstellen einer Kräftebilanz hergeleitet wird. Man betrachtet die Entwicklung des Geschwindigkeitsfelds  $\vec{u}$ . Beachtet man die Massen- und Impulserhaltung, erhält man

$$\frac{\partial \vec{u}}{\partial t} = \vec{u} \times (\nabla \times \vec{u}) + \nu \Delta \vec{u} - \nabla P, \quad (6)$$

wobei  $\nu$  die kinematische Viskosität,  $-\nabla P$  das Druckkraftfeld und  $\Delta$  der Laplace-Operator ist [1]. Der zweite Term auf der rechten Seite beschreibt die inneren Kräfte, die durch Wärmeübertragung, dissipative Erscheinungen, innere Reibung oder der Viskosität auftreten. Weitere äußere Kräfte können passend hinzugefügt werden, allerdings werden diese im Folgenden nicht explizit betrachtet. Um die Gleichung zu vereinfachen, kann angenommen werden, dass es sich bei dem betrachteten Medium um eine inkompressible Flüssigkeit handelt. Mathematisch kann das durch die Divergenzfreiheit von  $\vec{u}$  gefasst werden. Mit  $\nabla \cdot \vec{u} = 0$  erhält die NSG die folgende Form:

$$\frac{\partial \vec{u}}{\partial t} = 2\vec{u} \times \vec{\omega} + \nu \Delta \vec{u} - \nabla P \quad (7)$$

Hierbei ist  $\vec{\omega} = \frac{1}{2} \nabla \times \vec{u}$  und beschreibt die Wirblichkeit des Geschwindigkeitsfeldes. Durch

Anwendung der Divergenz auf Gleichung (7), verschwinden die linearen Terme aufgrund der Inkompressibilität und des Satzes von Schwarz. Man erhält die Druck-Poisson Gleichung:

$$\Delta P = 2\nabla \cdot (\widehat{\vec{u} \times \vec{\omega}}) \quad (8)$$

### 3.2 Fourier-Darstellung der Navier-Stokes-Gleichung

Um die NSG numerisch lösen zu können bietet es sich an, Gleichung (7) in den Fourierraum zu transformieren. Dazu wird zuerst die Druck-Poisson Gleichung transformiert. Man erhält

$$\hat{P} = -\frac{2i}{|\vec{q}|^2} \vec{q} \cdot (\widehat{\vec{u} \times \vec{\omega}}), \quad (9)$$

dabei ist  $\vec{q}$  der normierte Wellenvektor und  $i$  die imaginäre Einheit. Es ist anzumerken, dass  $\vec{u} \times \vec{\omega}$  als Ganzes transformiert wird, da dies numerisch günstiger zu berechnen ist. Als Nächstes kann Gleichung (7) transformiert werden. Durch anschließendes Einsetzen von Gleichung (9), erhält man die Fourier-Darstellung der NSG, die aufgrund der Inkompressibilität allein von  $\hat{\vec{u}}$  abhängig ist.

$$\frac{\partial \hat{\vec{u}}}{\partial t} = 2(\widehat{\vec{u} \times \vec{\omega}}) - \nu |\vec{q}|^2 \hat{\vec{u}} - \frac{2}{|\vec{q}|^2} \vec{q} (\vec{q} \cdot (\widehat{\vec{u} \times \vec{\omega}})) \quad (10)$$

## 4 Numerische Berechnung der Navier-Stokes-Gleichung

Jede Form von Code, auf den im Folgenden verwiesen wird, kann in dem verlinkten GitHub repository [6] eingesehen werden. Sämtliche Berechnungen werden mit den Bibliotheken NumPy, muFFT und muGrid durchgeführt. Um die Ergebnisse graphisch darstellen zu können, werden Matplotlib und SciencePlots [5] verwendet.

### 4.1 Implementierung des Lösungsverfahrens in Python

Da Gleichung (6) von der Form eines nicht-linearen Anfangswertproblems

$$\frac{\partial \vec{u}}{\partial t} = \vec{f}(t, \vec{u}) \quad (11)$$

ist, bei dem das Feld zum Zeitpunkt  $t = 0$  gegeben ist, bietet es sich an, die Zeitentwicklung mit einem interativen Verfahren, z.B. mit der Euler- oder Runge-Kutta-Methode, zu bestimmen. Dazu wird eine Python-Funktion „rhs“ definiert, die für jeden Zeitschritt die rechte Seite von Gleichung (10) bzw. von Gleichung (11) im Konfigurationsraum oder wahlweise im Fourierraum berechnet. Als Argumente nimmt die Funktion die momentane Zeit  $t$  und das

momentane Feld. Je nach Problem kann es sinnvoll sein, die Zeitentwicklung im Fourierraum durchzuführen (Forcing). Die Funktion erkennt durch den Datentyp des Arrays, ob es sich um ein Feld im Fourier- oder Konfigurationsraum handelt. Sie nimmt an, dass der Array eines Felds im Fourierraum komplexwertig und der des Felds im Konfigurationsraum reellwertig ist. Die beiden Ergebnisse der Berechnungen der rechten Seite unterscheiden sich geringfügig. Es wurden Unterschiede in der Ordnung  $10^{-14}$  festgestellt (siehe dazu die Python-Datei „test\_fourier\_propagation\_turbulence.py“). Bei der Verwendung der Funktion im Fourierraum ist es wichtig, auf die richtige Normalisierung zu achten, da der Algorithmus sonst sehr schnell divergieren kann. Ein sinnvolles Vorgehen ist es, die einzelnen Terme von Gleichung (10) mit dem angegebenen Feld im Fourierraum zu berechnen und anschließend zu addieren. Für das Ergebnis im Konfigurationsraum muss der gesamte Term noch rücktransformiert werden. Die Funktion gibt einen `nd.array` aus, der  $\vec{f}(t, \vec{u})$  in numerischer Form entspricht. Der Code und die genauen Details zur Berechnung sind in „SSP.py“ in der Klasse „SpecOps“ aufgeführt. Wichtig anzumerken ist, dass die Berechnung des ersten Terms die Rotation des Vektorfelds  $\vec{u}$  beinhaltet. Diese wird durch eine separate Funktion „curl“ bestimmt, welche ebenfalls als Methode in der Klasse gespeichert ist.

Die Funktion „rhs“ kann in dieser Form verwendet werden, um die Zeitentwicklung  $\vec{u}_{t+dt}$  von  $\vec{u}_t$  zu berechnen, wobei  $dt$  die Zeitschrittgröße ist. Um das Feld durch die Zeit zu propagieren, wird die Runge-Kutta-Methode vierter Ordnung verwendet.

$$\vec{u}_{t+dt} = \vec{u}_t + \frac{dt}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad (12)$$

Dazu müssen die Runge-Kutta-Koeffizienten  $k_1, k_2, k_3$  und  $k_4$  ebenfalls für jeden Zeitschritt berechnet werden. Diese sind abhängig von  $\vec{f}(t, \vec{u})$  und  $dt$ . Es bietet sich deshalb wieder an, eine Funktion „Runge\_Kutta\_4“ zu definieren, die als Argumente die Funktion „rhs“, die momentane Zeit  $t$ , das momentane Feld und die Zeitschrittgröße  $dt$  nimmt. „Runge\_Kutta\_4“ berechnet dann den zweiten Term der rechten Seite aus Gleichung (12). Auch diese Funktion ist in der Klasse „SpecOps“ gespeichert. Durch das Einführen der Klasse reduzieren sich die übrigen Simulationsritte auf die Initialisierung und das Entwickeln des Anfangszustandes  $\vec{u}_0$  mittels Gleichung (12). Letzteres kann anschließend mit einer for-Schleife bewerkstelligt werden, die durch die Zeitschritte iteriert.

## 4.2 Aliasing und De-Aliasing

Aliasing ist ein Problem, welches spezifisch bei den hier verwendeten Methoden auftritt. Wie in [2] aufgeführt, entsteht es durch die Fouriertransformation von nicht-linearen Termen in Differentialgleichungen. Besteht ein Term zum Beispiel aus einer Funktion und ihrer Ableitung, also

$$u(x, t) = \sum_{k=-N}^N c_k e^{ikx} \quad \text{und} \quad \frac{\partial}{\partial x} u(x, t) = \sum_{l=-N}^N c_l i l e^{ilx}, \quad (13)$$

folgt daraus die Entwicklung des ganzen Terms

$$u(x, t) \frac{\partial}{\partial x} u(x, t) = \sum_{k,l=-N}^N c_k c_l i l e^{i(k+l)x} \quad (14)$$

Der Bereich der möglichen Wellenzahlen wird so von  $[-N, N]$  auf  $[-2N, 2N]$  erweitert. Da das verwendete Gitter nur ersteres auflösen kann, werden die großen Wellenzahlen fehlinterpretiert. Eine einfache Lösung des Problems bietet das De-Aliasing. Dabei werden die Fourierkoeffizienten der Entwicklung des nicht-linearen Terms, die zu den zu großen Wellenzahlen gehören, gleich null gesetzt. Eine Orientierung, welche der Fourierkoeffizienten gleich null gesetzt werden, ist die  $2N/3$  Regel. Hier werden alle Koeffizienten herausgefiltert, dessen Betrag größer ist als  $2/3$  der maximalen Wellenzahl  $N$ .

Dieses Schema ist in der Funktion „rhs“ der SpecOps Klasse, nach der Berechnung des nicht-linearen Terms realisiert (Zeile 190 u. 191). Hier werden zuerst die Positionen der hohen Wellenzahlen im Wellenvektor-Array bestimmt. Sie werden dann genutzt, um dieselben Stellen des Arrays des nicht-linearen Terms null zu setzen. Allerdings trägt das nur zu einer leichten Verbesserung der Stabilität des Algorithmus bei. Am ehesten fällt die Verbesserung bei der Betrachtung der Turbulenzen in späteren Abschnitten auf.

## 4.3 Testen des Algorithmus am Taylor-Green-Vortex

Der Code für dieses Kapitel ist in „taylor\_green\_vortex.py“ zu finden. In „stabilität\_001.py“ und „stabilität\_0005.py“ befindet sich der Code zur Erstellung der Stabilitätsdiagramme.

### 4.3.1 Der Taylor-Green-Vortex und dessen analytische Zeitentwicklung

Der Taylor-Green-Vortex (TGV) ist ein dreidimensionales, periodisches Geschwindigkeitsfeld. Es beschreibt einen oder mehrere Wirbel, die sich periodisch im Raum wiederholen, weshalb er sich optimal als Anfangsbedingung eignet. Zu dieser Anfangsbedingung existiert eine analytische Lösung der NSG. Dadurch kann der TGV als analytisches Ideal bei Tests des Lösungsalgorithmus verwendet werden. Die Berechnung der analytischen Lösung werde ich im Folgenden skizzieren. Man betrachtet hier aus Gründen der Übersichtlichkeit den TGV in der Ebene. So erhält man die allgemeine Form

$$\vec{u}(\vec{x}, t) = \begin{pmatrix} A(t) \cos(ax) \sin(by) \\ B(t) \sin(ax) \cos(by) \\ 0 \end{pmatrix} \quad (15)$$

Hierbei stellen  $A$  und  $B$  die im Allgemeinen zeitabhängigen Amplituden dar. Die Variablen  $a$  und  $b$  sind Größen, die die Geometrie des TGV beschreiben. Die Aufgabe wird sein diese Größen zu bestimmen bzw. zu wählen. Dazu wird im ersten Schritt die Inkompressibilität (d.h. Divergenzfreiheit) des Vektorfeldes verlangt. Man erhält die Bedingung

$$\nabla \cdot \vec{u} = A(t)a + B(t)b = 0 \quad (16)$$

Dieser Ausdruck vereinfacht sich, wenn man eine quadratische Geometrie d.h.  $a = b$  wählt. Man erhält die Bedingung  $A(t) = -B(t)$ , wodurch das Problem auf zwei Unbekannte reduziert wird. Um diese zu bestimmen, setzt man  $\hat{\vec{u}}$  in die Fouriertransformierte NSG ein. Tatsächlich heben sich der Druckgradient und der nicht-lineare Term für dieses Vektorfeld auf, sodass nur noch der dissipative Term übrig bleibt.

$$\frac{\partial \hat{\vec{u}}}{\partial t} = -\nu |\vec{q}|^2 \hat{\vec{u}} \quad (17)$$

Diese Gleichung ist leicht durch Separation der Variablen lösbar. Man erhält die zeitliche Entwicklung für  $A(t)$ .

$$A(t) = A_0 e^{-\nu |\vec{q}|^2 t} \quad (18)$$

Bei weiterer Analyse der Fouriertransformierten des Vektorfeldes, erhält man die Bedingung, dass  $a = q_x = q_y = q$ . Im Allgemeinen ist  $q = 2\pi/L$ . Da die numerischen Berechnungen mittels spektraler Methoden auf  $\pi$ -periodischen Gebieten geschehen müssen, setzen wird  $L = 2\pi$ . Der finale analytische Ausdruck für  $\vec{u}$  ist dann

$$\vec{u}(\vec{x}, t) = A_0 \begin{pmatrix} \cos(x) \sin(y) \\ -\sin(x) \cos(y) \\ 0 \end{pmatrix} e^{-2\nu t} \quad (19)$$

Er beschreibt einen exponentiellen Abfall der Amplitude des Vektorfeldes, dessen Steigung maßgeblich durch die Viskosität und die Größe des Gebiets gegeben ist. Eine Veranschaulichung der Zeitentwicklung ist im Anhang in Abb. 6 zu sehen. Bei fortschreitender Zeit erkennt man deutlich einen Abfall des Absolutbetrages des Vektorfeldes.

#### 4.3.2 Das Vergleichsverfahren

Um die analytische mit der numerischen Lösung anschaulich vergleichen zu können, benötigt man ein Verfahren, welches die beiden Lösungen zu jedem Zeitpunkt miteinander vergleicht. So kann man sehen, wie sich die Abweichungen, die durch die numerische Berechnung entstehen, mit der Zeit entwickeln. Es bietet sich an, die Amplituden der Lösungen zu vergleichen, allerdings müssen dazu die beiden Tensoren, die die Lösung darstellen, so angepasst werden, dass man einfache Aussagen über die Entwicklung treffen kann. Das Verfahren dazu kann wie folgt beschrieben werden: Sei  $U_{t,d,i,j,k}$  ( $i = 0 \dots m, j = 0 \dots n, k = 0 \dots o$ ) ein Tensor, der die Lösung an jedem Punkt im  $d = 3$  dimensionalen Raum  $(i, j, k)$ , zu jedem Zeitpunkt  $t$  beschreibt. Dann wird im ersten Schritt der Amplitudentensor  $A_{t,i,j,k}$  berechnet.

$$A_{t,i,j,k} = \sqrt{U_{t,1,i,j,k}^2 + U_{t,2,i,j,k}^2 + U_{t,3,i,j,k}^2} \quad (20)$$

Dieser wird dann auf einen Tensor niedrigerer Stufe  $B_{t,l}$  verringert, wobei  $l = 0 \dots mno$ . Durch Summieren über  $l$ , erhält man einen  $t$ -dimensionalen Vektor, der als Maß für die „totale Amplitude“ des Vektorfeldes zum Zeitpunkt  $t$  verwendet werden kann. Dies kann man nun für beide

Lösungen durchführen und dadurch den zeitlichen Verlauf der totalen Amplituden graphisch darstellen. Um nun den Vergleich der numerischen und der analytischen Lösung quantifizieren zu können, kann man den zeitlichen Verlauf der totalen Amplitude an einen exponentiellen Abfall der Form

$$g(t; A, \nu') = Ae^{-\nu' t} \quad (21)$$

anpassen. Dazu wird die Methode `curve_fit` der Bibliothek `scipy.optimize` verwendet. Hierbei entspricht  $A$  der totalen Amplitude zum Zeitpunkt  $t = 0$  und  $\nu'$  dem „Zerfallsparameter“, wobei dieser die Viskosität  $\nu$  und  $|\vec{q}|^2$  enthält (siehe Gleichung (18)). Die durch Anpassung bestimmten Parameter können direkt verglichen werden und auf Kompatibilität überprüft werden. Dazu soll die relative Abweichung der Anpassungsparameter an den numerischen und den analytischen Verlauf verwendet werden. So können die Qualität und die Stabilität des Lösungsalgorithmus für verschiedene Werte für die Viskosität, die anfängliche Amplitude  $A_0$  und die Größe der Zeitschritte überprüft werden.

#### 4.3.3 Vergleich der numerischen und analytischen Zeitentwicklung

In diesem Abschnitt soll das oben beschriebene Verfahren angewendet werden. Da die Entwicklung stark von der Viskosität, der anfänglichen Amplitude, der Zeitschrittgröße und der Anzahl an Punkten im Gebiet abhängt, werde ich hier das Ergebnis des Verfahrens an einem Beispiel zeigen. Alle Tests werden auf einem Gebiet der Größe  $(2\pi, 2\pi, 2\pi)$  mit  $(50, 50, 2)$  Punkten durchgeführt. Die Felder werden für Zeiten von  $t = 0$  bis  $t = 1$  berechnet. Tests ergaben, dass größere Intervalle nur geringfügige Auswirkungen auf die Genauigkeit der Anpassungsparameter haben. Für den ersten Vergleich werden  $\nu = 1$ ,  $A_0 = 1$  und  $dt = 0.001$  gewählt.

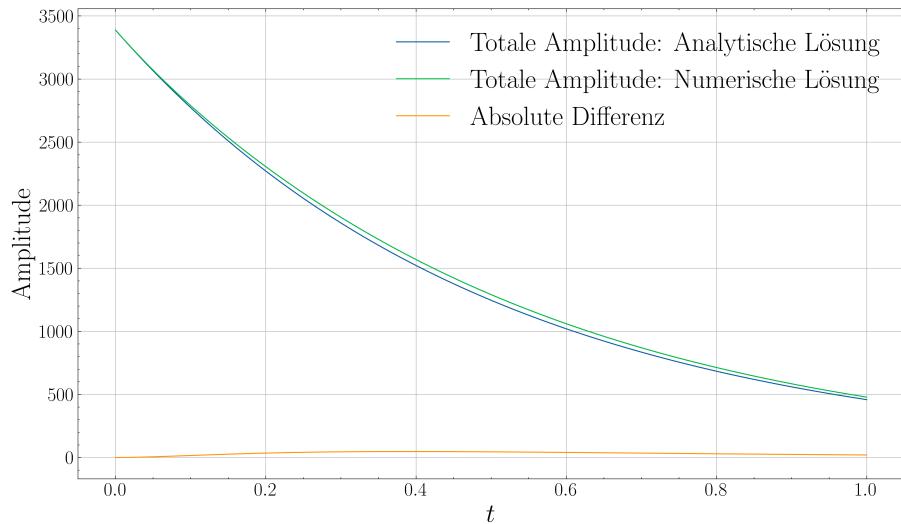


Abbildung 1: Zeitliche Entwicklung der totalen Amplituden der analytischen und der numerischen Lösung für ein Gebiet der Größe  $(2\pi, 2\pi, 2\pi)$  mit  $(50, 50, 2)$  Punkten, für  $\nu = 1$ ,  $dt = 0.001$  und  $A_0 = 1$ .

Die zeitlichen Verläufe der totalen Amplituden sind in Abb. 1 dargestellt. Hier erkennt man eindeutig den exponentiell abfallenden Verlauf sowohl der analytischen, als auch der numerischen Lösung. Man erkennt weiterhin, dass sich die Lösungen mit fortschreitender Zeit für

die gegebenen Rahmenbedingungen bis zu einem gewissen Punkt voneinander entfernen. Diese Abweichungen sind auch in der Darstellung des Geschwindigkeitsfeldes in Abb. 7, in Form von leichten Verformungen der Vortices bei fortschreitender Zeit, zu erkennen. In der Darstellung des analytischen Feldes (Abb. 6) bleiben diese Verformungen aus. Interessant bei diesen Parametern ist allerdings, dass die absolute Differenz der beiden Verläufe ein erkennbares Maximum bei ca.  $t = 0.39$  besitzt. Es zeigt sich, dass die numerische Entwicklung für größere Zeiten gegen die analytische Entwicklung konvergiert, d.h. dass die Verformungen für große Zeiten wieder abnehmen. Dieses Verhalten bedeutet, dass die Simulation für diese Parameter stabil ist. Das soll später genauer untersucht werden. Es ergaben sich die folgenden Anpassungsparameter und relative Abweichungen:

	$A$	$\nu'$
Analytisch	3387.202	2.000
Numerisch	3391.887	1.939
Relative Abweichung	0.138%	3.067%

Tabelle 1: Parameter der Anpassung an die zeitlichen Verläufe der analytischen und numerischen Lösung im Vergleich. Berechnet für  $\nu = 1$ ,  $A_0 = 1$  und  $dt = 0.001$ .

Die Unsicherheiten der bestimmten Parameter wird hier nicht aufgeführt, da diese aufgrund der verwendeten Anpassungsmethode zu gering sind, um sinnvolle Schlüsse aus ihnen ziehen zu können.

Es zeigt sich, dass  $\nu'$  für die Anpassung an den analytischen Zeitverlauf genau dem erwarteten exponenten  $2\nu$  entspricht. Somit kann die Anpassung zuverlässig als Ideal für den Vergleich genommen werden. Da die relativen Abweichungen der Anpassungsparameter eher gering ist, kann in diesem Beispiel von einer stabilen und qualitativ hochwertigen Simulation ausgegangen werden.

#### 4.3.4 Stabilität des Algorithmus

Tests ergaben, dass die Stabilität des Algorithmus von mehreren Parametern abhängt. Diese sind die anfängliche Amplitude  $A_0$ , die Viskosität  $\nu$ , die Größe der Zeitschritte  $dt$  und die Anzahl an Punkten im Gebiet  $(n_x, n_y, n_z)$ , an denen das Feld bestimmt wird. Die ersten drei genannten Parameter müssen aufeinander abgestimmt sein. Abgestimmt heißt, dass eine Kombination der Parameter nicht zu großen Änderungen zwischen aufeinander folgenden Zeitentwicklungsstufen führen darf, da sonst die Berechnungen divergieren oder unphysikalische Ergebnisse entstehen. Instabilitäten aufgrund der Wahl der  $n_i$ , können durch Anpassen von  $dt$  behoben werden. Um den Bereich der Stabilität zu untersuchen, wurde eine Testreihe durchgeführt, bei der für ein festes  $dt$ ,  $A_0$  vorgegeben wurde und anschließend  $\nu$  variiert wurde, bis Instabilität eintrat. Es wurde die obere Grenze  $\nu_{\max}$  und untere Grenze  $\nu_{\min}$  notiert. Gekennzeichnet sind diese entweder durch Divergenz des Algorithmus oder durch einen negativen Wert für  $\nu'$  in der Anpassung an die numerische Zeitentwicklung. Dies entspricht einem exponentiellen Anstieg und ist weder physikalisch (es existieren keine negativen Viskositäten), noch mit der analytischen Lösung vereinbar. Für einen Zeitschritt von  $dt = 0.001$  und  $t \in [0, 1]$  ergab sich das folgende Stabilitätsdiagramm:

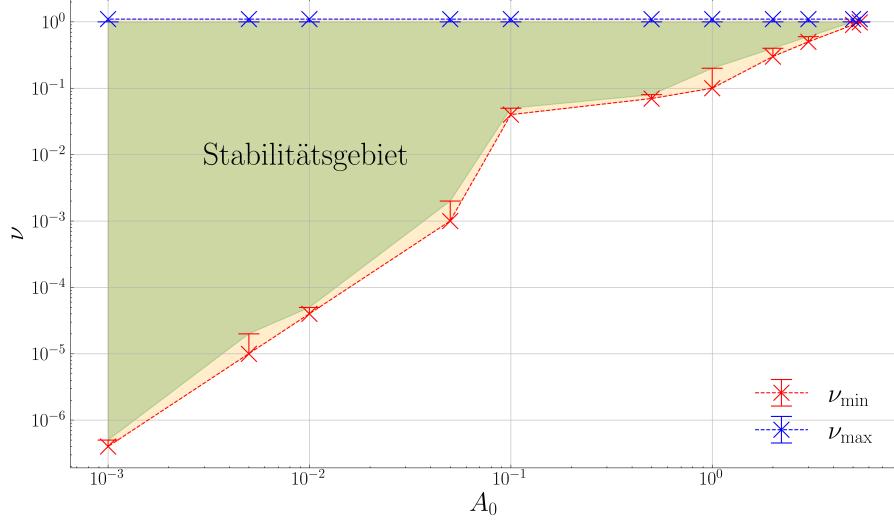


Abbildung 2: Stabilitätsdiagramm: Erlaubter Bereich für  $\nu$  für verschiedene  $A_0$ .

Die Fehlerbalken entstehen durch die endliche Größe der Schritte, die beim Variieren von  $\nu$  gemacht worden sind. Der grüne Bereich ist deshalb der Bereich, in dem Stabilität garantiert werden kann. Für die Werte in diesem Bereich tritt die in Abschnitt 4.3.1 erwähnte Konvergenz auf. Allerdings weichen die Werte der Anpassungsparameter  $A$  und  $\nu'$  bei Simulationen mit Werten für  $A_0$  und  $\nu$  am Rand des Gebiets immer stärker voneinander ab. Das bedeutet, dass auch die Zeitverläufe, trotz Konvergenz für geringe bis mittlere Zeiten, voneinander abweichen (vgl. Abb. 1). Da der stabile Bereich für kleine Amplituden breiter wird, sollte man also zu einer gegebenen Viskosität eine möglichst geringe, dennoch sinnvolle Amplitude wählen. Nach oben ist das Gebiet begrenzt, da hier  $\nu_{\min}$  und  $\nu_{\max}$  zusammenlaufen. Da die kinematische Viskosität von Flüssigkeiten in einem Bereich von ca.  $0.115 \text{ mm}^2/\text{s}$  (Quecksilber) bis  $1170 \text{ mm}^2/\text{s}$  (Glycerin) [8] liegt, ist der obere Bereich aber ohnehin wenig von Interesse. Weitere Testläufe zeigen, dass sich der Stabilitätsbereich für kleinere  $dt$  nach oben erweitert, allerdings nach unten gleich bleibt (siehe Anhang Abb. 8). Eine geringe Viskosität und Amplitude erlaubt die Wahl eines größeren  $dt$ . Das kann nützlich sein, wenn große Zeitspannen von Interesse sind und Rechenzeit gespart werden soll.

## 4.4 Anwendungsbeispiel: Turbulenzen

### 4.4.1 Das getriebene Geschwindigkeitsfeld

Um den Algorithmus an einem weiteren Problem zu testen, werden im Folgenden Turbulenzen, die durch konstante Energiezufuhr entstehen, simuliert. Dazu wird der Anfangszustand des Vektorfelds zufällig generiert und anschließend das Energiespektrum  $E(\vec{q}) = \frac{1}{2}|\hat{u}(\vec{q})|^2$  des Feldes an das von Kolmogorov hergeleitete Gesetz [4]

$$E(\vec{q}) \propto |\vec{q}|^{-5/3} \quad (22)$$

angepasst. Des Weiteren muss sichergestellt werden, dass das zufällig generierte Feld inkompressibel d.h. divergenzfrei ist. Es lässt sich zeigen, dass die Divergenzfreiheit im Fourierraum

$\vec{q} \cdot \hat{\vec{f}} = 0$  für jedes zufällig generierte Feld  $\hat{\vec{f}}$  gegeben ist, wenn es durch die Matrix

$$M_{\text{ink}} = \left( \bar{1} - \frac{\vec{q} \otimes \vec{q}}{|\vec{q}|^2} \right) \quad (23)$$

transformiert wird. Dabei stellt  $\bar{1}$  die  $3 \times 3$  Einheitsmatrix dar. Als Anfangszustand des Geschwindigkeitsfeldes wird das mit  $M_{\text{ink}}$  transformierte, zufällig generierte Feld verwendet. Diese Schritte sind in der Methode „random\_field\_generator“ der SpecOps Klasse realisiert und geschieht in „random\_field\_generator.py“. Als Input benötigt die Methode die Eigenschaften der Klasse für das vorliegende Problem und die Amplitude des Feldes zum Zeitpunkt  $t = 0$ . Propagiert man nun das Feld mit dem oben beschriebenen Algorithmus durch die Zeit, kann man das Betragsquadrat des Feldes zu verschiedenen Zeitpunkten betrachten. Die Zeitentwicklung des Feldes geschieht in „turbulence\_forcing.py“.

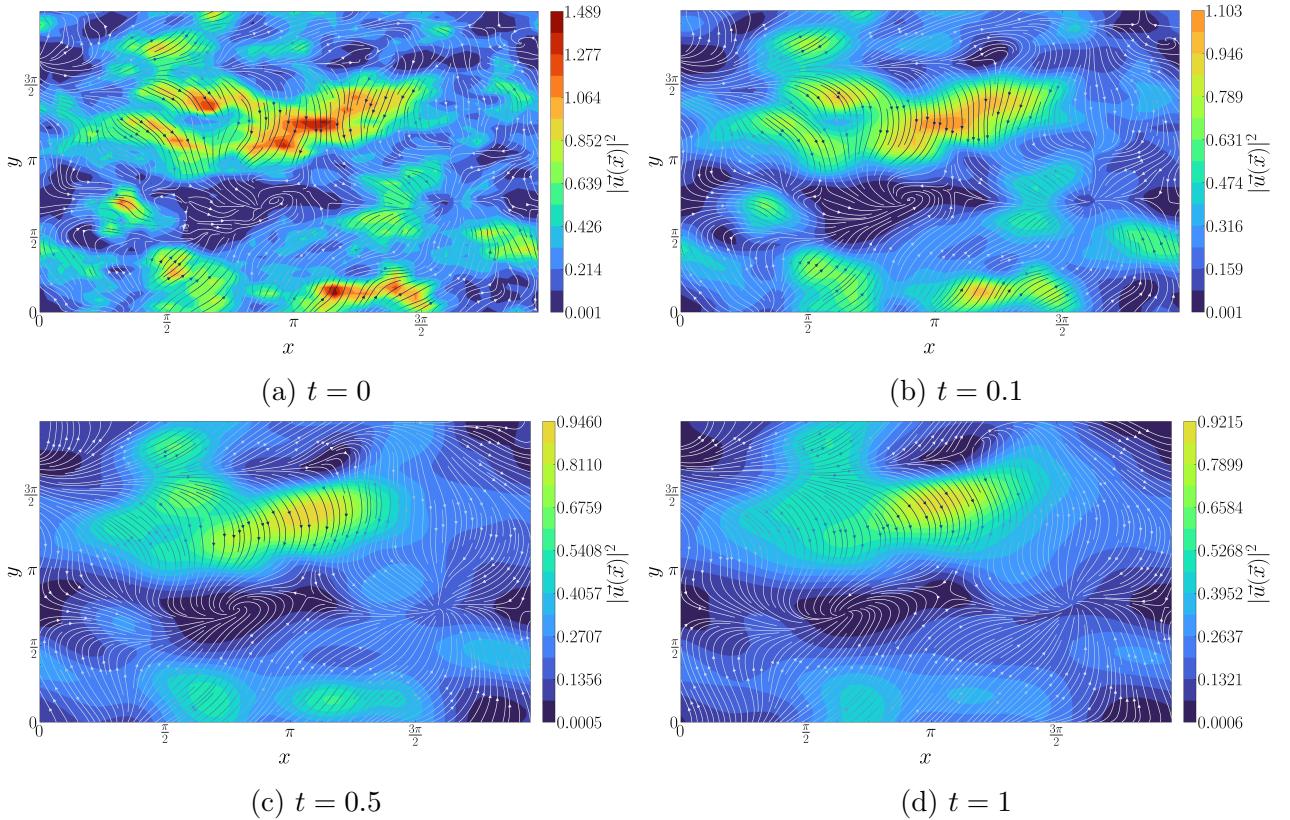


Abbildung 3: Querschnitt ( $z = 0$ ) des Betragsquadrates eines zufällig generierten, inkompressiblen Geschwindigkeitsfeldes mit Kolmogorov-Energiepektrum in einem Gebiet der Größe  $(2\pi, 2\pi, 2\pi)$  mit  $(50, 50, 2)$  Punkten für  $\nu = 0.1$ ,  $A_0 = 0.1$  und  $dt = 0.001$  zu vier verschiedenen Zeitpunkten.

Die Erstellung der Grafiken geschah in „plot\_driven\_fields.py“. Soll das nicht getriebene Feld dargestellt werden, muss darauf geachtet werden, dass die für das Forcing verantwortliche Zeile (Zeile 86) in „turbulence\_forcing.py“ auskommentiert ist. Man sieht, dass das Feld anfänglich sehr rau ist. Der Unterschied in der Amplitude zwischen zwei benachbarten Gitterpunkten kann also sehr groß sein. Die eingezeichneten Vektoren zeigen die Richtung, d.h. die Stromlinien des Feldes an den Punkten. Die Farbe der Vektoren entspricht dem Hintergrund und kann

ignoriert werden. Mit fortschreitender Zeit werden die starken Unterschiede geglättet und die Amplitude (siehe Entwicklung der Werte auf dem Farbbalken) verringert sich. Das Vektorfeld „zerfließt“ also mit der Zeit. Dieses Ergebnis ist so weit im Einklang mit den Beobachtungen der bisherigen Sektionen. Um nun Turbulenzen zu erzeugen, wird eine Methode benötigt, mit der das Geschwindigkeitsfeld an gewissen Stellen getrieben werden kann. Gleichzeitig wird dadurch teilweise das Zerfließen unterdrückt. Erreichen kann man das zum Beispiel mit einem zusätzlichen Term in Gleichung (10), der eine antreibende Kraft im Fourierraum darstellt. Weitere Methoden, die spezifisch für Pseudo-Skalare Simulationen ausgelegt sind, sind in [3] aufgeführt. Die einfachste, dort genannte Methode ist das sogenannte Einfrieren kleiner Moden. Dazu wird ein Intervall an Wellenzahlen festgelegt, das „Forcingband“. Es entählt geringe Wellenzahlen. Bei der Zeitentwicklung werden die zu diesen Wellenzahlen gehörigen Fourierkoeffizienten nicht mitentwickelt. Dadurch entsteht ein konstanter Beitrag im Energiespektrum. Es bietet sich für dieses Problem an, die Zeitentwicklung im Fourierraum durchzuführen. So müssen in jedem Zeitentwicklungschnitt die passenden Fourierkoeffizienten zurückgesetzt werden. Dies kann von Anfang an oder ab einer gewissen Zeit gemacht werden. Auch hier ist auf die richtige Normalisierung zu achten. Der Code für die Simulation des getriebenen Feldes ist in „turbulence\_forcing.py“ zu finden. Es wird zuerst ein zufälliges Feld generiert. Dann werden die Indices der fünf niedrigsten Moden bestimmt. Diese werden in der for-Schleife der Zeitentwicklung genutzt, um die zugehörigen Fourierkoeffizienten bei jeder Iteration zu ihren Anfangswerten zurückzusetzen.

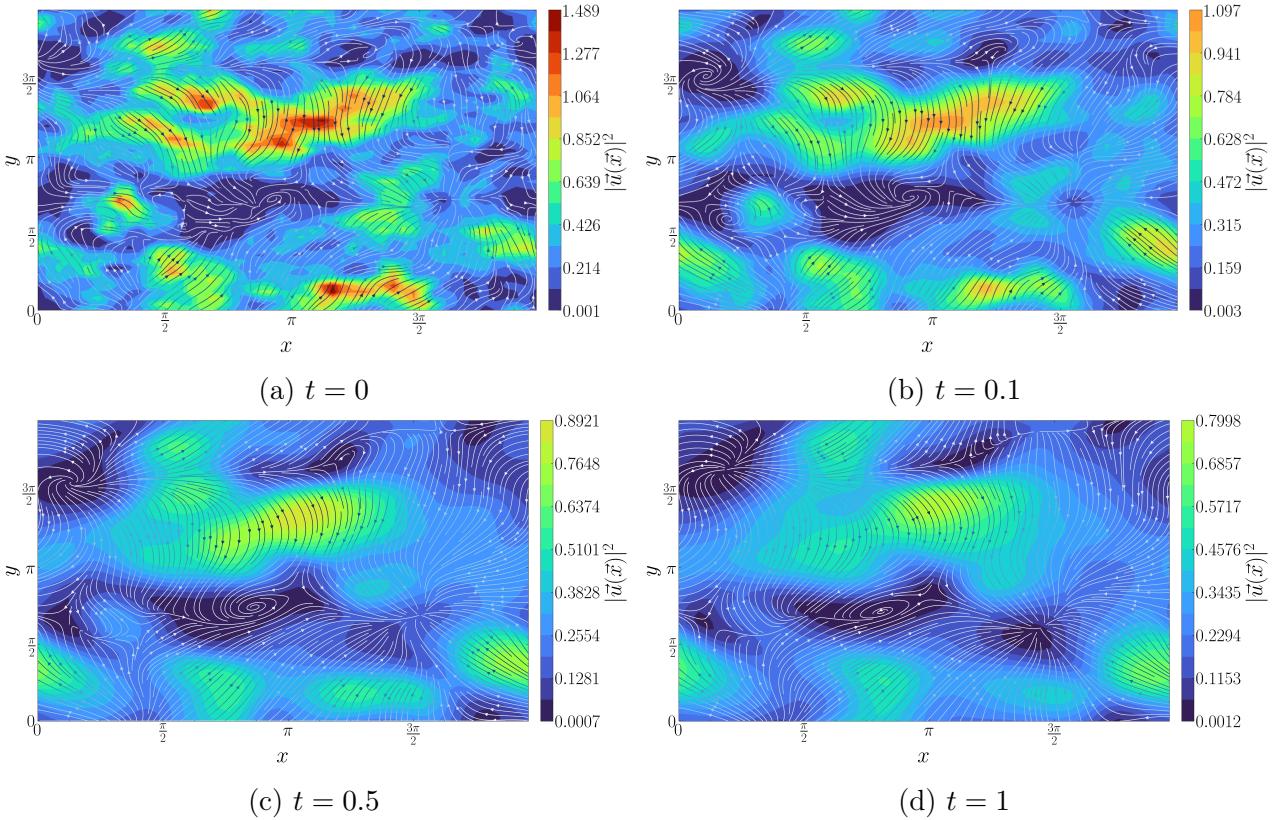


Abbildung 4: Querschnitt ( $z = 0$ ) des Betragsquadrates eines zufällig generierten, inkompressiblen getriebenen Geschwindigkeitsfeldes mit Kolmogorov-Energiespektrum in einem Gebiet der Größe  $(2\pi, 2\pi, 2\pi)$  mit  $(50, 50, 2)$  Punkten für  $\nu = 0.1$ ,  $A_0 = 0.1$  und  $dt = 0.001$  zu vier verschiedenen Zeitpunkten.

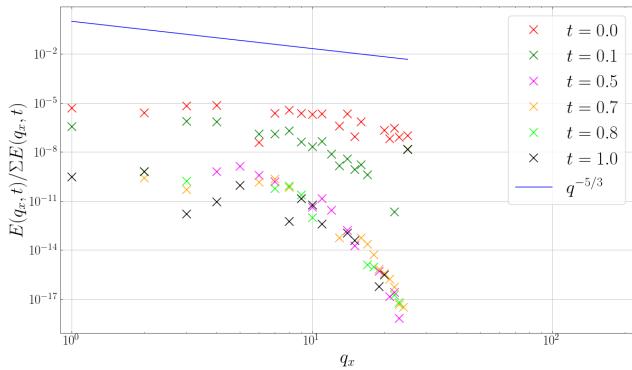
Da die zufällig generierten Felder in Abb. 3 und in Abb. 4 gleich sind und sich nur durch den Antrieb unterscheiden, kann man sie nun miteinander vergleichen. Wie zu erwarten, sind die Felder zum Zeitpunkt  $t = 0$  identisch, da hier die Zeitentwicklung, in der das Forcing implementiert ist, noch nicht greift. Allerdings kann man beim nächsten abgebildeten Zeitschritt deutliche Unterschiede erkennen. Ein Unterschied ist der Amplitudenpeak im rechten unteren Bildrand. Im nicht getriebenen Feld ist dieser ca. 0.3 Einheiten kleiner als im Getriebenen. Das spiegelt die angesprochene Energiezufuhr wider. Man erkennt weiterhin, dass die Stromlinien, die von dem Peak ausgehen, in das Gebiet zeigen und so als einfließende Flüssigkeitsmenge interpretiert werden können. Eine ähnliche, jedoch schwächere Quelle findet sich im unteren linken Bildrand. Ein weiterer Unterschied sind die Wirbel, die bei dem getriebenen Feld im linken Bildrand auftreten. Im nicht getriebenen Bild liegen die Stromlinien an diesen Stellen beinahe parallel. Das bedeutet, dass das Einfrieren der Moden hier eine Art Störung erzeugt, die zu den Wirbeln führt. Des Weiteren scheint der Wirbel oben links im Bild eine schwache Quelle des Feldes zu sein, da alle abgehenden Stromlinien von dem Zentrum des Wirbels aus weg zeigen. Deutlicher zu erkennen ist das in den nächsten Zeitschritten. Auch das kann als Zufluss einer Flüssigkeit interpretiert werden, allerdings in einem deutlich geringeren Maßstab, als bei dem Peak im unteren rechten Bildrand. Die Störung, die den Wirbel erzeugt, ist also anschaulich als „Einspritzen“ der Flüssigkeit an der Stelle des Wirbels zu verstehen. Der Wirbel unten links scheint allerdings eher durch ein Wechselwirken der Quelle unten links und der durch den oberen Wirbel entstandenen Störung zu entstehen. Das Auftreten dieser Phänomene zeigt, dass das Festhalten der kleinsten Wellenzahlen reicht, um das Geschwindigkeitsfeld signifikant zu ändern und Merkmale von Turbulenzen zu erzeugen.

#### 4.4.2 Das Energie- und Dissipationsspektrum

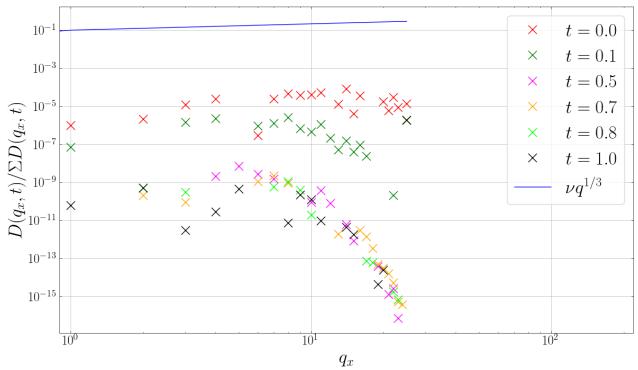
Um die oben berechneten Felder weiter zu untersuchen, kann man das Energiespektrum  $E(\vec{q})$  und das Dissipationsspektrum

$$D(\vec{q}) = \nu |\vec{q}|^2 |\hat{\vec{u}}|^2 \quad (24)$$

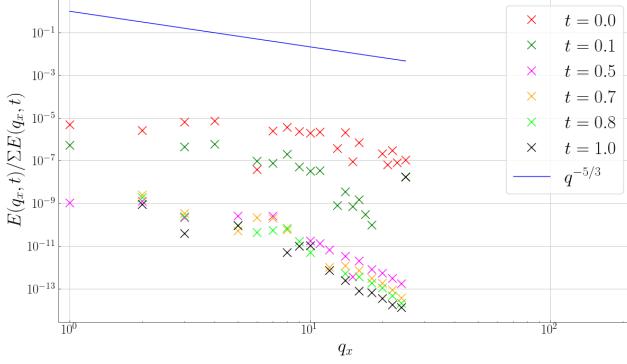
über die Zeitentwicklung hinweg, betrachten. Sie berechnen sich aus dem vorliegenden Feld im Fourierraum. In den folgenden Plots sind beide Spektren, jeweils mit und ohne De-Aliasing für verschiedene Zeiten dargestellt. Zusätzlich ist die erwartete Proportionalität abgebildet. Die Proportionalität des Dissipationsspektrums wurde aus dem Zusammenhang zwischen den beiden Spektren abgeleitet.



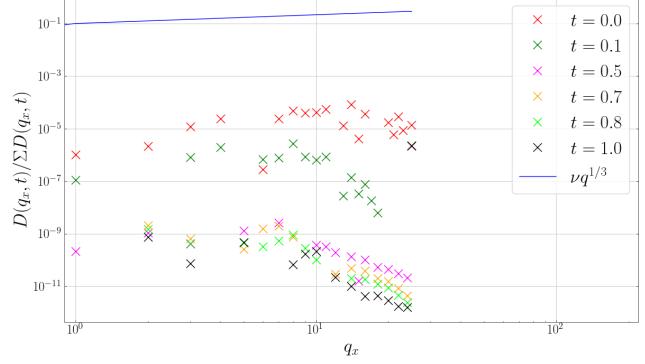
(a) Energiespektrum ohne De-Aliasing



(b) Dissipationsspektrum ohne De-Aliasing



(c) Energiespektrum mit De-Aliasing



(d) Dissipationsspektrum mit De-Aliasing

Abbildung 5: Querschnitt des Energie- und Dissipationsspektrum bei  $q_y = 40$  und  $q_z = 0$  des Zufalls-Kolmogorov generierten Feldes zu verschiedenen Zeiten bei einer Gebietsgröße von  $(2\pi, 2\pi, 2\pi)$  mit  $(50, 50, 2)$  Punkten,  $\nu = 0.1$ ,  $A_0 = 0.1$  und  $dt = 0.001$ . Ohne und mit De-Aliasing.

Der Code zur Berechnung der normierten Spektren und zur Erstellung der Grafiken ist in „spectra.py“ zu finden. Man erkennt sofort, dass die Werte des Energiespektrums sowohl bei höheren Wellenzahlen, als auch über die Zeit abfallen. Der Abfall für höhere Wellenzahlen steht im Einklang mit Abb. 9. Man kann davon ausgehen, dass man auch für die ersten Zeiten einen Abfall für hohe Wellenzahlen beobachtet hätte, wenn mehr Gitterpunkte verwendet worden wären. Darauf wurde hier allerdings aufgrund der verfügbaren Rechenkapazitäten verzichtet.

Der Abfall der Energie über die Zeit ist allerdings unerwartet. Da durch das Forcing Energie zugeführt wird, wäre zu erwarten gewesen, dass die Energie zunimmt. Möglich wäre, dass die zugeführte Energie, im Vergleich zum exponentiellen Abfall aus den bisherigen Abschnitten, vernachlässigbar klein ist. Eine Möglichkeit das zu testen wäre eine Art Over-Forcing, wobei nicht nur 5, sondern deutlich mehr Werte festgehalten werden. Dies soll hier aber nicht weiter diskutiert werden.

Weiterhin machen sich jetzt die Folgen des De-Aliasing erkennbar. Die Spektren, bei deren Berechnung De-Aliasing verwendet wurde, zeigen einen deutlich schwächeren Abfall sowohl bei fortschreitender Zeit als auch bei hohen Wellenzahlen.

Zusätzlich kann man erkennen, dass die Werte des Energiespektrums für den ersten Zeitschritt, bis auf den nötigen Proportionalitätsfaktor, grob mit dem Gesetz von Kolmogorov übereinstimmt. Es scheint, als befände sich das rechte Ende der Darstellung ca. in der Mitte des Inertialbereichs, d.h. der Bereich, in dem das  $-5/3$ -Gesetz gilt (siehe z.B. Abb. 9 als Vergleich). Durch den zeitlichen Abfall verschiebt sich allerdings der Inertialbereich und entfernt sich von der dargestellten Proportionalität. Außerdem scheint es, als hätten die Spektren zu späteren Zeiten einen steilern Verlauf. Ähnliches gilt für das Dissipationsspektrum. Man erkennt für kleine Wellenzahlen einen leichten Anstieg, der ca. der erwarteten Proportionalität entspricht.

## 5 Anhang

### 5.1 Abbildungen

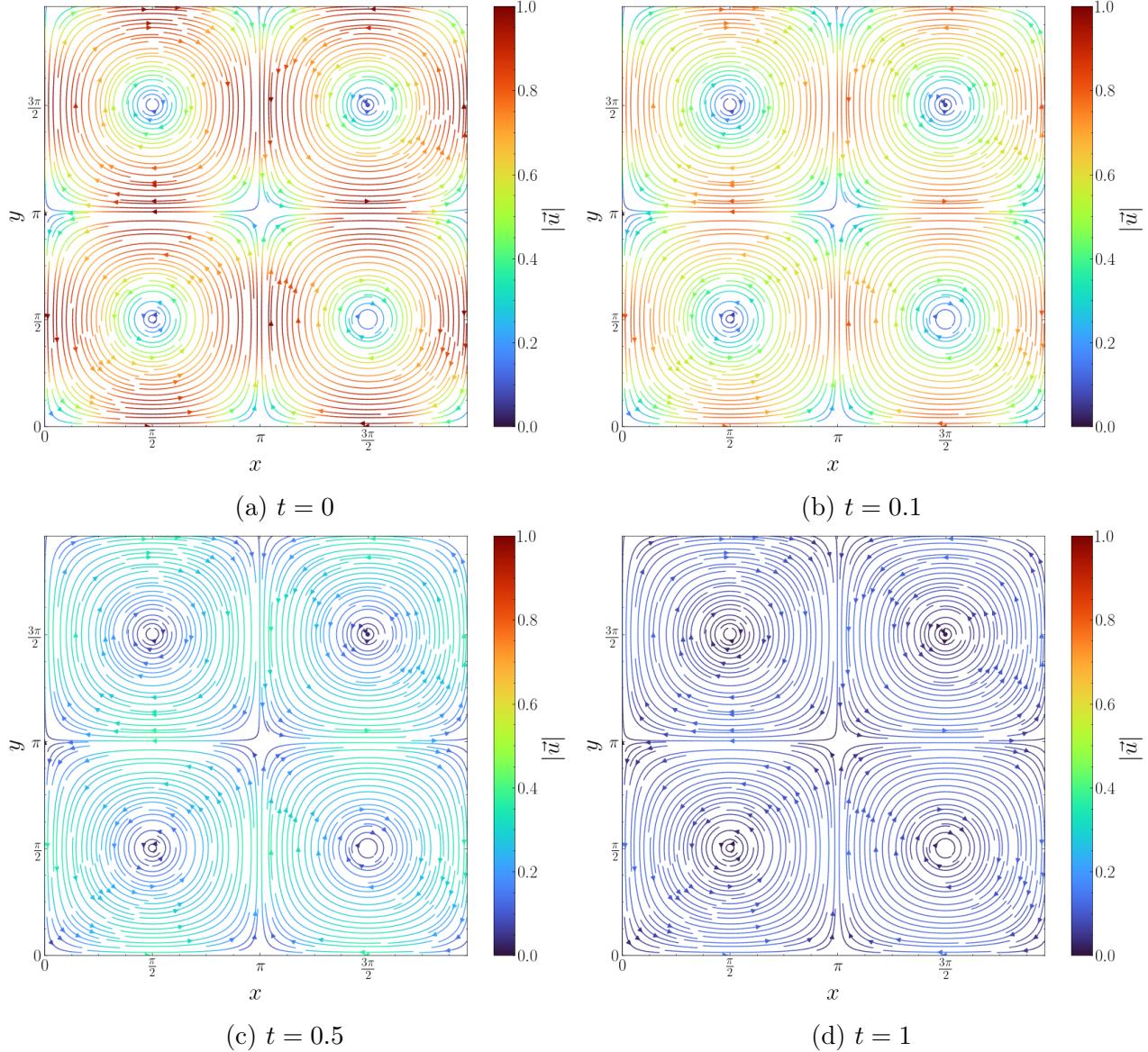


Abbildung 6: Querschnitt der analytischen Zeitentwicklung des Taylor-Green-Vortex für ein Gebiet der Größe  $(2\pi, 2\pi, 2\pi)$  mit  $(50, 50, 2)$  Punkten für  $\nu = 1$ ,  $A_0 = 1$  und  $dt = 0.001$  zu vier verschiedenen Zeiten.

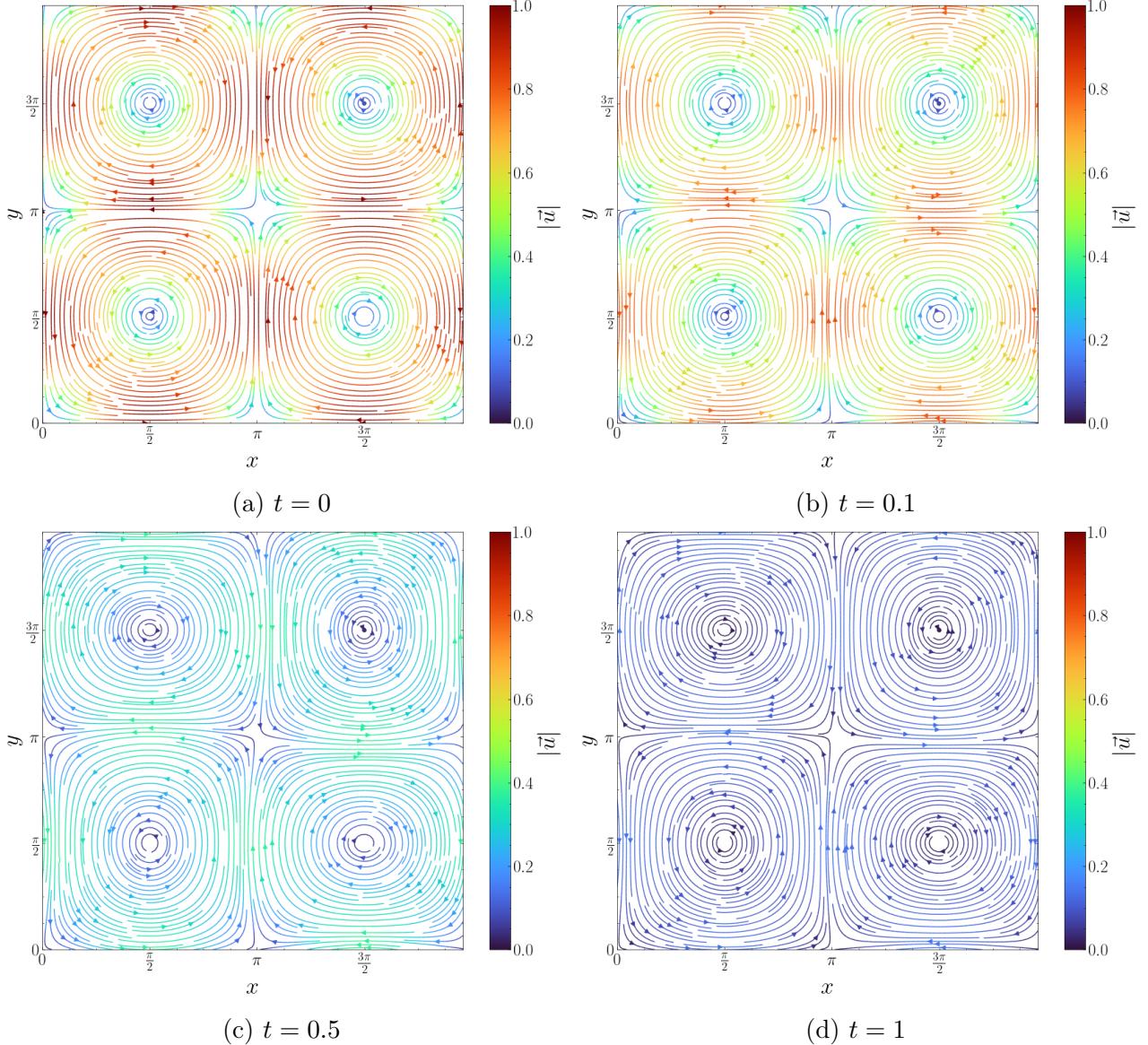


Abbildung 7: Querschnitt der numerischen Zeitentwicklung des Taylor-Green-Vortex für ein Gebiet der Größe  $(2\pi, 2\pi, 2\pi)$  mit  $(50, 50, 2)$  Punkten für  $\nu = 1$ ,  $A_0 = 1$  und  $dt = 0.001$  zu vier verschiedenen Zeiten.

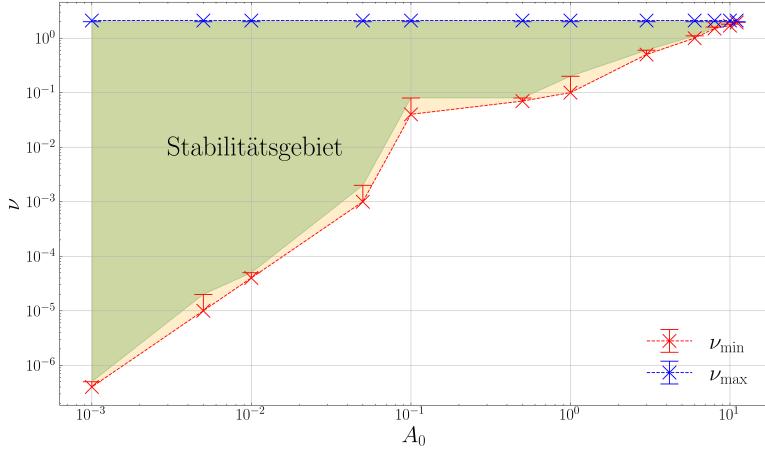


Abbildung 8: Stabilitätsdiagramm: Erlaubter Bereich für  $\nu$  für verschiedene  $A_0$  bei  $dt = 0.0005$  für  $t \in [0, 1]$ .

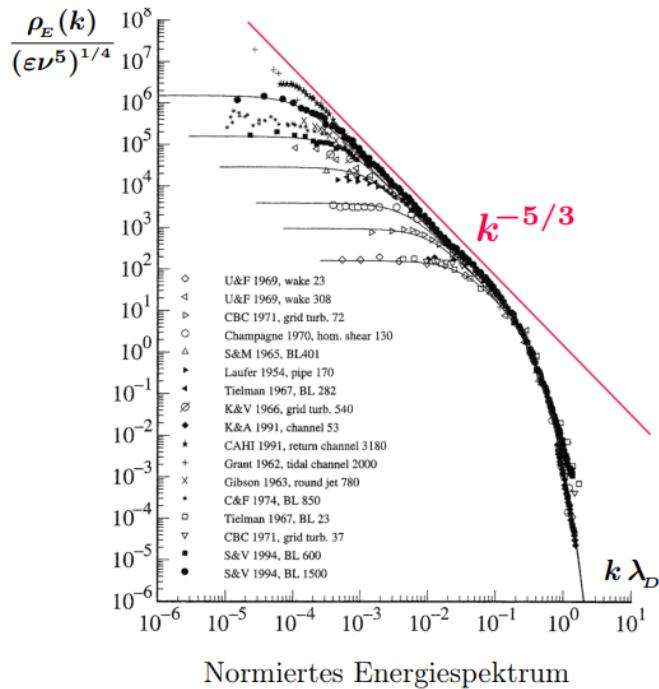


Abbildung 9: Beispiel eines Energiespektrums mit markiertem Inertialbereich, in dem das  $-5/3$ -Gesetz gilt. Entnommen aus [4]

## Literatur

- [1] Dario Götz. *Einführung in das Navier-Stokes-Problem*. 2006.
- [2] Michael Wilczek. „Kohärente Strukturen in turbulenten Strömungen“. Magisterarb. Institut für Theoretische Physik, Westfälische Wilhelms-Universität Münster, 2007. Kap. A.3 Stabilität des Verfahrens.
- [3] Anton Daitche. „Statistische und geometrische Eigenschaften turbulenter Strömungen“. Diss. Institut für Theoretische Physik, Westfälische Wilhelms-Universität Münster, 2009.

- [4] Heinz Horner. *Turbulenz*. Institut für Theoretische Physik der Ruprecht-Karls-Universität Heidelberg. 2011.
- [5] John D. Garrett. „garrettj403/SciencePlots“. Version 1.0.9. In: *Zenodo* (Sep. 2021). DOI: [10.5281/zenodo.4106649](https://doi.org/10.5281/zenodo.4106649). URL: <http://doi.org/10.5281/zenodo.4106649>.
- [6] Johann Gockel. „JohannGockel/SSP“. In: *GitHub* (Aug. 2024). URL: <https://github.com/JohannGockel/SSP>.
- [7] Navier-Stokes-Equation. *Navier-Stokes-Equation — Millennium Problems — Clay Mathematics Institute*. Online; zugegriffen am 7.08.2024. URL: <https://www.claymath.org/millennium/navier-stokes-equation/>.
- [8] Viskosität. *Viskosität — Physikalische Größen — Wikipedia, The Free Encyclopedia*. Online; zugegriffen am 5.08.2024. URL: <https://de.wikipedia.org/wiki/Viskosit%C3%A4t>.