Institut für Theoretische Physik
der Universität zu Köln

Prof. Dr. Simon Trebst
Michael Becker, Peter Bröcker

Computerphysik

Übungsblatt 10

SS 2014

Website: http://www.thp.uni-koeln.de/trebst/Lectures/2014-CompPhys.shtml
Abgabedatum: Montag, 23. Juni 2014 vor Beginn der Vorlesung

40. 

Programmiertechniken

Bislang haben wir alle Algorithmen auf Probleme angewandt und am Ende ein numerisches Ergebnis erhalten. Dies ist aber nicht zwangsläufig der Fall und in vielen Fällen möchte man einen Algorithmus so gestalten, dass er mittels einer wohldefinierten Anzahl von Parametern auf eine ganze Klasse von Problemen angewandt werden kann. Sie haben eine Verallgemeinerung dieser Art vielleicht selber schon angewandt, in dem Sie Ihre Lösung in einer Funktion implementiert haben, die zum Beispiel eine zu integrierende Funktion sowie den Integrationsbereich als Argumente akzeptiert und den Wert des entsprechenden Integrals zurückgibt. Im nächsten Schritt wäre es nun praktisch, wenn man die Lösung des Problems für *alle* möglichen Parameter gleichzeitig kennt. Im obigen Beispiel könnten wir zum Beispiel die untere Integralgrenze fixieren, aber die Lösung als Funktion der oberen Integralgrenze auftragen wollen.

Um dies zu erreichen, benutzen wir **symbolische Mathematik**, die in Python mit dem Paket **SymPy** implementiert ist. In dieser Aufgabe möchten wir Ihnen zunächst eine Reihe von Anwendungsmöglichkeiten vorstellen, bevor wir uns dann in einer der Übungsaufgaben anschauen wollen, wie man sogar quantenmechanische Rechnungen einfach durchführen kann.

Vorweg sei gesagt, dass es für diese Aufgabe sehr praktisch sein wird, das IPython Notebook zu verwenden. Nutzer von Mathematica, Maxima und ähnlichen Programmen werden sich dort schnell heimisch fühlen, da es dem gleichen Prinzip folgt. Zum Starten im CIP-Pool öffnen Sie eine Konsole, tippen

```
ipython notebook
```

ein und bestätigen mit `Return`. Es wird sich nun der Browser öffnen und die Startseite des iPython Notebooks öffnet sich. Oben rechts befindet sich der Button um ein neues Notebook zu öffnen. Das Prinzip ist schnell erklärt: In die grau hinterlegten Zellen können Sie den normalen Pythoncode tippen, den Sie dann mit `Shift+Return` ausführen können. Alle Ausgaben werden direkt in dem weißen Bereich darunter angezeigt.

Wir beginnen diese Aufgabe damit, das Paket `SymPy` zu laden und die Latex-Ausgabe zu aktivieren. So bekommen Sie die Ergebnisse nicht in der schnell unleserlich werdenden Form `x**2` präsentiert, sondern in gut lesbaren Latexvariante x^2:

```
import sympy as sp
sp.init_printing()
```

Im nächsten Schritt definieren wir Variablen, x und a, die von nun an symbolisch behandelt werden sollen:

```
x, a = sp.symbols('x a')
```

Diese können wir nun verwenden um Ausdrücke zu erzeugen. Zum Beispiel können wir eine Funktion f definieren durch

```
f = x**a + sp.sin(x)
f
```

Sie können die gewohnten Ausdrücke für mathematische Funktionen wie `sin(x)` verwenden, nur müssen Sie die aus dem SymPy Modul nutzen. Das alleinstehende `f` in einer Zeile sorgt dafür, dass der Ausdruck in Ihrem Notebook in formatierter Form ausgegeben wird. Diesen Ausdruck können wir nun weiter bearbeiten. Zum Beispiel können wir ihn nach x ableiten:

```
f_prime = sp.diff(f, x)
f_prime
```

Sie werden feststellen, dass der so erhaltene Ausdruck vereinfacht werden kann. Auch diese Operation beherrscht `SymPy`:

```
sp.simplify(f_prime)
```

SymPy kennt außerdem eine Reihe von Funktionen aus der linearen Algebra. Zum Beispiel können wir die uns bekannten Formeln für das Invertieren einer 2×2 Matrix und der Berechnung der Determinante nachvollziehen. Zunächst definieren wir vier neue Symbole `a, b, c, d`, welche die Einträge unserer Matrix darstellen:

```
a, b, c, d = sp.symbols('a b c d')
M = sp.Matrix(( [a, b], [c, d] ))
M.det()
M.inv()
```

Beachten Sie, dass Sie den invertierten Ausdruck unter Umständen noch vereinfachen sollten, um das gewohnte Ergebniss zu erhalten.

Besonders nützlich ist die Möglichkeit, einzelne Gleichungen oder auch ganze Systeme von Gleichungen nach einer oder mehreren Variablen aufzulösen. Als Beispiel möchten wir das Gleichungssystem

$$\begin{aligned} a+b &= 1 \\ a-b &= 0 \end{aligned}$$

lösen. Dazu bringen wir sie zunächst in eine Form, in der auf der rechten Seite nur der Nullvektor steht

$$\begin{aligned} a+b-1 &= 0 \\ a-b &= 0 \end{aligned}$$

und erstellen dann eine Liste der Gleichungen. Diese übergeben wir dann an den Solver von `SymPy`, der außerdem noch wissen muss, nach welchen Variablen aufgelöst werden soll:

```
eqs = [a + b - 1, a - b]
sp.solve(eqs, a, b)
```

Anhand dieser Beispiele konnten Sie schon einmal einen kleinen Einblick in die vielfältigen Einsatzmöglichkeiten SymPys gewinnen. Es existiert eine große Menge weiterer Untermodule,

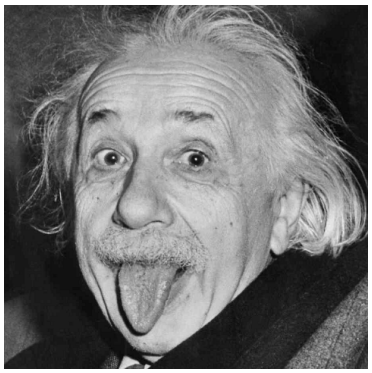
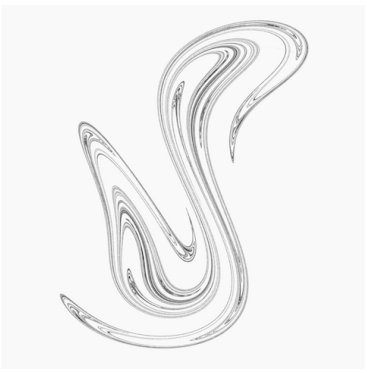
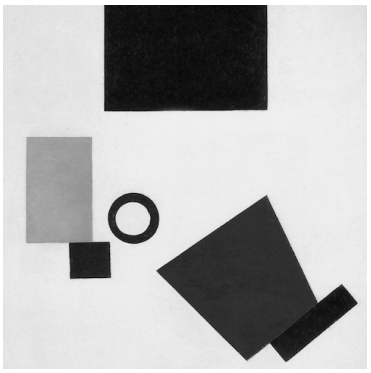

Abbildung 1: Die drei Bilder, die mit der Singulärwertzerlegung untersucht werden sollen.

die Gebiete wie Zahlentheorie oder Differentialgeometrie umfassen. Eines dieser Untermodule, nämlich das Modul um quantenmechanische Rechnungen durchzuführen, wollen wir in der nächsten Aufgabe vorstellen. Für alle weiteren Funktionen möchten wir auf die umfassende Dokumentation verweisen.

41. Einstein singulär zerlegt

5 Punkte

In der Vorlesung haben sie die **Singulärwertzerlegung** von Matrizen kennengelernt. Dabei wird eine beliebige Matrix M als Produkt dreier spezieller Matrizen dargestellt:

$$M = U\Sigma V^*, \tag{1}$$

wobei Σ eine Diagonalmatrix ist, deren Einträge gerade die Singulärwerte von M sind. Je größer der einzelne Singulärwert ist, desto wichtiger ist sein Beitrag zur Berechnung von M. Daraus folgt, dass man die Matrix M durch eine Matrix $\tilde{M}^{(n)}$ *approximieren* kann, indem man

$$\tilde{M}^{(n)} = U\tilde{\Sigma}^{(n)}V^* \tag{2}$$

berechnet, wobei in der Matrix $\tilde{\Sigma}^{(n)}$ nur die n größten Singulärwerte enthalten sind und alle restlichen auf Null gesetzt wurden.

Diese Tatsache wollen wir nun zur **"Bildkompression"** ausnutzen. Der Einfachheit halber wollen wir dazu ein graustufiges Bild betrachten, so dass wir die Graustufen des Bildes als eine Matrix mit Elementen zwischen 0 und 1 interpretieren können. Wir bilden nun zunächst die Singulärwertzerlegung, behalten die n größten Singulärwerte und verwerfen die restlichen Werte wie oben beschrieben. Nun wollen wir untersuchen, wieviel Bildinformation in diesen ersten n Singulärwerten verschlüsselt ist und rekonstruieren das Bild, indem wir $\tilde{M}^{(n)}$ berechnen. Wieviele Singulärwerte sind nötig, um Albert Einstein wiederzuerkennen?

Für die Singulärwertzerlegung können Sie dabei auf die Funktion `svd(m)` aus dem Lineare-Algebra-Modul von NumPy zurückgreifen, welche als einzigen Parameter die Matrix M übernimmt, und wie folgt aufgerufen wird:

```
u, s, v = numpy.linalg.svd(m)
```

Die Rückgabewerte sind die Matrizen U und V^* und eine Liste `s`, die die Singulärwerte in absteigender Reihenfolge enthält.

Auf der Kurswebsite finden Sie drei Bilder (**Einstein**, **Duffing**, **Malewitsch**), die Sie mit dem Befehl `imread(path, flatten=True)` aus dem `scipy.misc` Modul einlesen können (siehe Aufgabe 26).

Aufgaben

- Implementieren Sie die Berechnnung der approximierten Version eines Bildes wie oben angegeben für eine fixe Zahl n von berücksichtigten Singulärwerten.
- Untersuchen Sie für die drei verlinkten Bilder die Resultate, die Sie für $n = 2$, $n = 5$ und $n = 20$ Singulärwerte erhalten. Welche Eigenschaften muss ein Bild haben, um besonders gut durch wenige Singulärwerte approximiert werden zu können?

42. Verschränkte Spins

5 Punke

Einer der fundamentalsten Unterschiede zwischen der klassischen und der Quantenmechanik ist die Möglichkeit der **Verschränkung** quantenmechanischer Objekte. Einstein, Podolsky und Rosen (EPR) schlugen 1935 ein Gedankenexperiment vor, um zu demonstrieren, dass diese Verschränkung eine Unvollständigkeit der Quantenmechanik impliziert[1]. Mittlerweile ist aber klar, dass die Quantenmechanik nicht unvollständig ist, sondern dass ihr Gedankenexperiment die nichtlokale Natur beispielhaft illustriert.

Im EPR Experiment werden zur gleichen Zeit zwei Teilchen erzeugt, die dann beliebig weit voneinander entfernt werden. Unter bestimmten Umständen wissen wir bei der Messung des Zustands eines der Teilchen, in welchem Zustand sich das andere Teilchen befindet. In diesem Fall sind beide Teilchen miteinander verschränkt.

In dieser Aufgabe möchten wir genau diese Situation untersuchen. Konkret betrachten wir einen quantenmechanischen Zustand zweier Spins der gegeben ist als

$$|\psi\rangle = \cos(\alpha)\,|{+}{+}\rangle + \sin(\alpha)\,|{-}{-}\rangle\,. \tag{3}$$

Dieser Zustand hängt ab von einem Winkel α, der den Mischgrad der Superposition bezeichnet. Wir wollen herausfinden, unter welchen Bedingungen dieser als verschränkt bezeichnet werden kann und wie wir diese Verschränkung quantifizieren können. Dies werden wir mithilfe der sogenannten **Verschränkungsentropie** tun. Bis wir diese aber berechnen können, müssen wir ein theoretisches Werkzeug – die reduzierte Dichtematrix – einführen und lernen, wie wir mit `SymPy` Quantenmechanik betreiben können.

Der nun folgende Text ist sicherlich länger als Sie es von anderen Aufgaben gewohnt sind. Dies ist aber dem Umstand geschuldet, dass wir versucht haben, die Aufgabe so vollständig wie möglich zu formulieren und möglichst wenig Vorkenntnisse vorauszusetzen. Am Ende finden Sie noch einmal eine ganz konkrete Anleitung was Sie tun müssen, um die Verschränkungsentropie auszurechnen.

[1] Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?

Verschränkung

Die Frage, ob zwei (oder mehr) Teilchen miteinander verschränkt sind, ist äquivalent zu der Frage, ob sich der Zweiteilchenzustand als einfaches Produkt von Einteilchenzuständen schreiben lässt. Wir möchten dies an einem Beispiel näher erläutern: Betrachten wir zwei Teilchen, deren Zustand jeweils durch eine Quantenzahl beschrieben werden, deren Werte jeweils 0 oder 1 sind, also zum Beispiel zwei Qubits. Zwei der möglichen Zweiteilchenzustände sind

$$|\psi\rangle = |00\rangle = |0\rangle \otimes |0\rangle = |0\rangle\,|0\rangle\,,$$
$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right).$$

Der erste Zustand ist unverschränkt, da er als Produktzustand geschrieben werden kann. (Wir haben dabei verschiedene Notationen angegeben, die Sie in der Literatur antreffen können.) Für den zweiten Zustand ist eine solche Umformung nicht möglich, er ist also verschränkt.

Dichtematrizen

Dichtematrizen stellen einen alternativen Zugang zu Zuständen der Quantenmechanik dar und sind gerade für Systeme mehrerer Teilchen von großem Vorteil. Die grundlegende Idee lässt sich an folgendem Beispiel gut illustrieren: Ein quantenmechanischer Zustand kann äquivalent durch einen Zustandsvektor $|\psi\rangle$ aus dem Hilbertraum oder durch einen Projektor $\mathcal{P}_\psi = |\psi\rangle\langle\psi|$ auf diesen Zustand beschrieben werden. Diesen Projektor werden wir später als Dichtematrix ρ bezeichnen. Die Angabe des Projektors hat sogar den Vorteil, dass die frei wählbare Phase von $|\psi\rangle$ nicht auftaucht. Erwartungswerte von beliebigen Observablen O lassen sich ebenfalls leicht in diesem Formalismus berechnen:

$$\begin{aligned}\langle O\rangle &= \langle\psi|\,O\,|\psi\rangle = \sum_{m,n}\langle\psi|\,n\rangle\,\langle n|\,O\,|m\rangle\,\langle m|\,\psi\rangle \\ &= \sum_{m,n}\langle m|\,\psi\rangle\,\langle\psi|\,n\rangle\,\langle n|\,O\,|m\rangle \\ &= \sum_{m}\langle m|\,\mathcal{P}_\psi O\,|m\rangle \\ &= \mathrm{Tr}\left(\mathcal{P}_\psi O\right)\end{aligned}$$

Der Ausdruck $\mathrm{Tr}(\cdot)$ steht für *trace*, also die Summe über die Eigenzustände der gewählten Basis. Ist der Projektor als Matrix dargestellt, so ist es einfach die Spur der Matrix. Die Erwartungswerte von Observablen werden also berechnet, indem man die Erwartungswerte in allen Basiszuständen des Operators $\mathcal{P}_\psi O$ berechnet und aufaddiert. In Kurzform schreiben wir $\rho = |\psi\rangle\langle\psi| = \mathcal{P}_\psi$

Diese Prozedur lässt sich nun auch auf Zustände mehrerer Teilchen verallgemeinern. Wir betrachten konkret einen quantenmechanischen Zustand zweier Teilchen. Der Hilbertraum eines einzelnen Spins ist $\mathcal{H} = \{|+\rangle, |-\rangle\}$. Ein Zustand des Zweispinsystems lässt sich als Tensorprodukt zweier Einteilchenzustände schreiben: $|\pm,\pm\rangle = |\pm\rangle\,|\pm\rangle = |\pm\rangle \otimes |\pm\rangle$ Die Dichtematrix eines solchen Zustand ist wie gewohnt definiert durch $\rho = |\pm\pm\rangle\langle\pm,\pm|$ und um Observablen auszuwerten muss die Spur nun über alle Quantenzahlen, d.h. über alle möglichen Spinkonfigurationen $|++\rangle, |+-\rangle, |-+\rangle, |--\rangle$, genommen werden.

Reduzierte Dichtematrizen

Die zu Beginn gestellte Frage, ob ein Zustand $|\psi\rangle$ sich als Produktzustand $|\chi\rangle \otimes |\phi\rangle$ schreiben lässt, verallgemeinert sich im Dichtematrixformalismus auf die Frage, ob sich eine eine Dichtematrix ρ als ein Produkt $\rho_1 \otimes \rho_2$ schreiben lässt. Man kann diese Frage beantworten, indem man die sogenannte **reduzierte Dichtematrix** ρ_r bildet, für die man die Spur über alle Zustände aus dem Einteilchenhilbertraum $\mathcal{H}_2$ von Teilchen 2 nimmt und so die Anzahl der Freiheitsgrade von ρ auf die von ρ_1 reduziert:

$$\rho_r = \sum_{|\phi\rangle \in \mathcal{H}_2} \langle\phi|\,\rho\,|\phi\rangle\,. \tag{4}$$

Dieses Konzept mag auf den ersten Blick ein wenig Abstrakt wirken, weshalb wir einige Formeln konkret in Matrixschreibweise darstellen wollen. Die ausgewertete Dichtematrix eines Einteilchenzustands mit den Zuständen $\{|0\rangle, |1\rangle\}$ ist gegeben durch:

$$\begin{pmatrix} \langle 0|\,\rho\,|0\rangle & \langle 0|\,\rho\,|1\rangle \\ \langle 1|\,\rho\,|0\rangle & \langle 1|\,\rho\,|1\rangle \end{pmatrix}$$

Ein unverschränkter Produktzustand ergäbe sich aus dem Kronecker-Produkt zweier dieser Matrizen

$$\begin{pmatrix} \langle 0|\,\rho_1\,|0\rangle & \langle 0|\,\rho_1\,|1\rangle \\ \langle 1|\,\rho_1\,|0\rangle & \langle 1|\,\rho_1\,|1\rangle \end{pmatrix} \otimes \begin{pmatrix} \langle 0|\,\rho_2\,|0\rangle & \langle 0|\,\rho_2\,|1\rangle \\ \langle 1|\,\rho_2\,|0\rangle & \langle 1|\,\rho_2\,|1\rangle \end{pmatrix}$$

Die Matrixform der reduzierten Dichtematrix ergibt sich dann mit obiger Formel (5) zu:

$$\begin{pmatrix} \langle 00|\,\rho\,|00\rangle + \langle 01|\,\rho\,|01\rangle & \langle 00|\,\rho\,|10\rangle + \langle 01|\,\rho\,|11\rangle \\ \langle 10|\,\rho\,|00\rangle + \langle 11|\,\rho\,|01\rangle & \langle 10|\,\rho\,|10\rangle + \langle 11|\,\rho\,|11\rangle \end{pmatrix} \tag{5}$$

Das Maß, um die Stärke der Verschränkung zu quantifizieren ist die **von-Neumann Entropie**, die definiert ist durch

$$S = -\mathrm{Tr}\left[\rho_r \log\left(\rho_r\right)\right].$$

Wir geben die genaue Form dieses Maß zunächst unmotiviert an und werden es dann in dieser Aufgabe charakterisieren. Das generelle Ziel ist, die reduzierte Dichtematrix auf eine Zahl zu komprimieren. Die Bedeutung des Logarithmus einer Matrix sollten wir allerdings näher erläutern. Er ist definiert als die inverse Operation des Matrixexponentials und wird durch das Modul `scipy.linalg` mit der Funktion `logm` bereitgestellt. Die Spur ist wie zuvor die gewohnte Spur über den Hilbertraum des verbleibenden Teilchens.

Spins in SymPy

Ultimativ möchten wir die Verschränkungseigenschaften zweier Spins auf dem Computer studieren. In `SymPy` steht uns das Untermodul `sympy.physics.quantum` zur Verfügung, mit dem wir unter anderem Spins erzeugen und manipulieren können. Wie Sie mittlerweile in der Quantenmechanikvorlesung gehört haben, benötigen wir zur Festlegung des Spinzustands zwei Quantenzahlen j und m, die in unserem Fall die Werte $j = \frac{1}{2}$ und damit $m = \pm\frac{1}{2}$ annehmen. Mit genau dieser Information können wir nun einen entsprechenden Zustand, also einen Ket, erzeugen:

```
from sympy.physics.quantum.spin import JzKet, JzBra
j, m = sp.symbols('j m')
j = '1/2'
```

```
m = '-1/2'
a_ket = JzKet(j, m)
a_bra = JzBra(j, m)
```

Wir benötigen nun noch weitere Begriffe und Funktionalitäten um unsere verschränkten Spins zu untersuchen. Beginnen wir mit dem Produkt zweier Zustände. Man multipliziert zunächst nur zwei Ausdrücke aneinander, ohne sie auszuwerten. Für die Auswertung müssen wir `sympy.physics.quantum.qapply` verwenden. Zuletzt müssen wir um die Dichtematrix zu berechnen Ausdrücke wie $|\psi\rangle\langle\psi|$ aufstellen und berechnen. Wenn wir $|\psi\rangle$ kennen, so können wir durch *daggern* den entsprechenden Bra erzeugen. Diese Funktion ist durch das Untermodul `sympy.physics.quantum.Dagger` implementiert.

All diese Werkzeuge finden im untenstehenden Skript Verwendung. Wir definieren jeweils einen Ket für Up- und Downspin und bilden dann eine Superposition aus beiden. Um die Normierungskonstante zu finden, bilden wir zunächst das innere Produkt aus Bra und Ket und normieren dann den Zustand um die Dichtematrix aufzustellen:

```
import sympy as sp
from sympy.physics.quantum.spin import JzKet, JzBra
from sympy.physics.quantum import *
j, m = sp.symbols('j m')
j = '1/2'
m = '-1/2'
down = JzKet(j, m)
up = JzKet(j, j)
state = up + down; state
norm = qapply(Dagger(state)*state)
rho = 1/sp.sqrt(norm)*(state*Dagger(state))
```

Ein praktischer Tipp: Denken Sie daran, dass Sie Zwischenergebnisse ausgeben können, in dem Sie die gewünschte Variable alleine in die letzte Zeile einer IPythonzelle schreiben und diese dann mit `Shift+Return` ausführen, d.h. in etwa so:

```
rho = 1/sp.sqrt(norm)*(state*Dagger(state))
rho
```

Um mehrere Spins zu untersuchen, müssen wir das Tensorprodukt zweier Spinzustände bilden, welches wir mit dem Befehl `TensorProduct` erledigen. Mit obigen Variablen erzeugen wir nun einen Zustand, der nicht die Superposition von Up und Down Zustand eines Teilchens ist, sondern ein Teilchen im Zustand Up und ein Teilchen im Zustand Down beschreibt:

```
two_particle_state = TensorProduct(up, down); two_particle_state
```

In der derzeitigen Implementierung von SymPy müssen wir auf ein Produkt zweier Tensorzustände noch den Befehl `represent` anwenden um ein numerisches Ergebnis zu erhalten:

```
represent(qapply(two_particle_state * Dagger(two_particle_state)))
```

Aufgabe

Wir sind nun mit den nötigen Rüstzeug gewappnet um ein (möglicherweise) verschränktes Zweispinsystem zu untersuchen. Zur Erinnerung: Untersuchen wollen wir den von einem Winkel α abhängigen Zustand

$$\cos(\alpha)\,|{+}{+}\rangle + \sin(\alpha)\,|{-}{-}\rangle \tag{6}$$

für den wir nun die von-Neumann Entropie als Funktion von α berechnen wollen. Plotten Sie diese als Funktion von α und beschreiben Sie, an welchem Punkt die beiden Spins minimal bzw. maximal verschränkt sind.

Anleitung zur Berechnung der Entropie

Dieser Abschnitt soll die wichtigsten Punkte der etwas längeren Einleitung noch einmal zusammenfassen um Ihnen die Bearbeitung zu erleichen:

Erstellen Sie zunächst mit der Klasse `JzKet` zwei Zustände $|+\rangle$ und $|-\rangle$, welche die Einteilchenbasis darstellen. Mit `TensorProduct`, `sympy.sin` und `sympy.cos` können Sie dann den Zustand (6) erzeugen, der aus zwei Teilchen besteht. Unter zuhilfenahme der Funktion `Dagger` erstellen Sie dann den Dichteoperator ρ. Nun müssen Sie noch die reduzierte Dichtematrix berechnen, indem Sie die Spur über die Zustände eines der Teilchen nehmen. Die konkrete Form der reduzierten Dichtematrix ist in Gleichung (5) gegeben. Die Ergebnisse eines jeden Erwartungswerts können Sie in einer `numpy`-Matrix abspeichern, die Sie dann mit den Funktionen `scipy.linalg.logm` und `numpy.trace` weiterverarbeiten können um die Entropie zu berechnen.

43. Tsunami-Wellen

optionale Aufgabe – 6 Punkte
Abgabe am Montag, 30. Juni

Wir wollen in dieser Aufgabe die Entstehung von **Tsunamis** simulieren. Diese Wasserwellen entstehen durch Seebeben und breiten sich sehr schnell aus, bevor sie in flachen Küstengebieten an Höhe gewinnen was zu teils katastrophalen Folgen führen kann.

Ihre theoretisches Beschreibung geht auf eine der bedeutendsten Gleichungen der Hydrodynamik zurück, die **Navier-Stokes-Gleichung**

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t}+\mathbf{v}\cdot\nabla\mathbf{v}\right)=-\nabla p+\mu\nabla^2\mathbf{v}+\left(\zeta+\frac{\mu}{3}\right)\nabla(\nabla\cdot\mathbf{v})+\mathbf{f},$$

wobei die zentrale Variable die Geschwindigkeit eines fluiden Mediums $\mathbf{v}$ ist. Die weiteren auftretenden Konstanten sind die Dichte ρ, der Druck p, die (Gravitations-)Kraft f, sowie die Volumensviskosität ζ und die Scherviskosität μ.

Ausgehend von dieser Gleichung kann man unter der Verwendung einiger Annahmen, beispielsweise der Inkompressibilität von Wasser ($\frac{\partial\rho}{\partial t}=\nabla\rho=0$), für ein zwei-dimensionales Problem zwei sogenannte **Seichtwassergleichungen** für die Geschwindigkeit u und die Höhe h des Wasserstands herleiten

$$\begin{aligned}\frac{\partial h(t,x)}{\partial t}+\frac{\partial}{\partial x}(h(t,x)-b(x)u(t,x))&=0,\\ \frac{\partial u(t,x)}{\partial t}+\frac{1}{2}\frac{\partial}{\partial x}u(t,x)^2+g\frac{\partial h(t,x)}{\partial x}&=0.\end{aligned}$$

Dabei sei $g=0.0981$ die Erdbeschleunigung und $b(x)$ das Bodenprofil des Ozeans. Für unsere Simulationen wollen wir ein derartiges Bondeprofil im Bereich $-x_0$ bis $x_0=100\,(\equiv 10km)$

betrachten, welches wie folgt parametrisiert sei

$$b(x) = 30\left(\frac{x}{x_0}\right)^6 - 71\left(\frac{x}{x_0}\right)^4 + 56\left(\frac{x}{x_0}\right)^2 - 15$$

Lösen Sie nun einen Tsnumi aus, indem sie eine gaußförmige Welle zum Zeitpunkt $t = 0$ betrachten

$$h(t = 0, x) = h_0 e^{(-x^2/w^2)}$$

mit einer anfänglichen Höhe $h_0 = 0.01$ und Breite $w = 10$ der Welle. Die Geschwindigkeit $u(t,x)$ sei zu Beginn überall 0. Legen Sie dabei als Randbedingung

$$h(t, x = -x_0) = h(t, x = x_0) = 0 = u(t, x = -x_0) = u(t, x = x_0)$$

fest.

Sie sollen als Algorithmus die **Leap-Frog Methode** verwenden. Diese verwendet die Ableitungen sowohl im Ort als auch in der Zeit *zentriert*. Für eine partielle Differentialgleichung der Form

$$\frac{\partial}{\partial t} a = -\frac{\partial F(a)}{\partial x}$$

lautet die Iterationsgleichung

$$a_i^{(n+1)} = a_i^{(n-1)} - \frac{\tau}{h_x}\left(F(a_{i+1}^{(n)}) - F(a_{i-1}^{(n)})\right).$$

Finden Sie eine geeignete Funktion $F(h,u)$ für obige Seichtwassergleichungen. Für die Integration wählen Sie $h_x = 0.1$ und $\tau = 0.05$.

Simulieren Sie nun den Tsunami und beobachten Sie, was geschieht, wenn die Welle das flache Wasser und schließlich die Küste erreicht. Beenden Sie die Simulation, sobald die Welle die Küste erreicht hat. Weil wir die Küste selbst zu stark vereinfacht modelliert haben, ist das hier beobachten Verhalten (im Gegensatz zum Seichtwasser) physikalisch nicht mehr korrekt.