Institut für Theoretische Physik
der Universität zu Köln

Prof. Dr. Simon Trebst
Michael Becker, Peter Bröcker

Computerphysik

Übungsblatt 2

SS 2014

Website: http://www.thp.uni-koeln.de/trebst/Lectures/2014-CompPhys.shtml
Abgabedatum: Dienstag, 22. April 2014, bis 12 Uhr

Eine häufig wiederkehrende Aufgabe ist die **Bestimmung von Nullstellen** einer beliebigen Funktion. Um dies zu tun, haben Sie in der Vorlesung einige iterative Verfahren kennengelernt, die wir auf diesem Übungszettel implementieren und vergleichen wollen. Wenn wir uns mit der Nullstellensuche in die komplexe Zahlenebene begeben, kann eine derartige Suche ohne viel Aufwand zu fraktale Strukturen führen – den Newton-Fraktalen.

8. Divide and Conquer *5 Punkte*

Als erste Methode soll eine **Grid Search** implementiert werden. Dazu wird ein Stück des Definitionsbereich um die Nullstelle herum in kleinere Bereiche unterteilt. Als nächstes wird der Bereich identifiziert, in dem die Nullstelle liegt und dieser wieder unterteilt. Wenn die gewünschte Präzision erreicht ist, wird das Verfahren abgebrochen.

Alternativ dazu kann ein Verfahren angewendet werden, das Sie bereits in der Mathematik als Intervallschachtelung kennengelernt haben und auch als **Binary Search** bekannt ist. Nachdem zwei Endpunkte gewählt wurden, wird der Bereich so halbiert, dass die Nullstelle wieder im Bereich liegt. Wie zuvor wird auch dieser Vorgang dann abgebrochen, wenn die gewünschte Genauigkeit erreicht ist.

In dieser und der nächsten Aufgabe soll eine Funktion untersucht werden, die Sie in der Quantenmechanik kennenlernen werden oder bereits kennen gelernt haben. Bei der Lösung des Problems eines Teilchens in einem endlichen Potentialtopf trifft man auf die folgende Gleichung:

$$k\tan(ka) - \kappa = 0$$

Verwenden Sie die oben beschriebenen Verfahren um die Nullstelle der obigen Funktion zu bestimmen. Setzen Sie dazu $a, \kappa = 1$.

9. Newton Verfahren *5 Punkte*

In der Vorlesung haben Sie das **Newton Verfahren** kennengelernt. Dieses soll nun ebenfalls angewendet werden, um die Nullstelle der oben genannten Funktion zu bestimmen. Implementieren Sie den Algorithmus und bestimmen Sie die Nullstelle für Startpunkte im Intervall $[0, 100]$. Tragen Sie dann die Werte der gefundenen Nullstellen gegen die Startpunkte auf.

10. Newton-Fraktale

optionale Aufgabe – 6 Punkte
Abgabe am Montag, 28. April

Das Newtonverfahren lässt sich problemlos auch auf **komplexe Funktionen** erweitern, die eine vielfältige Struktur an Nullstellen aufweisen können. Insbesondere kann es Punkte geben, an denen der Algorithmus nicht konvergiert, ganz gleich wie die Parameter gewählt werden.

Für diese Aufgabe betrachten wir konkret die Funktion $f(z) = \cos(z)\,z^4 - 1$ auf der komplexen Menge $[-2,2] \times [-2\mathrm{i}, 2\mathrm{i}]$. Das Ergebnis des Newtonverfahrens auf dieser Menge lässt sich sehr schön visualisieren. Dazu speichern wir für jeden Startpunkt aus der obigen Menge die Anzahl an Iterationschritten, die wir benötigen um mit vorgegebener Genauigkeit ε an 0 heranszukommen. Das bedeutet, wir diskretisieren die Menge, beginnen einen Newtonlauf an jedem der Punkte z_0 und prüfen, ob nach dem i-ten Schritt $|f(z_i)| < \varepsilon$ erfüllt ist. Falls dem so ist, speichern wir den Wert i und beginnen am nächsten Punkt von vorn. So erhalten wir für jeden Punkt der diskretisierten Menge einen Wert.

Diese Werte müssen aber während des Programmlaufs gespeichert werden und dazu benötigen wir einen geeigneten Datentyp. Für unseren Zweck bestens geeignet ist der Typ *ndarray* aus dem Paket **numpy**, das Sie während dieser Vorlesung insbesondere in der linearen Algebra noch ausführlich kennenlernen werden. Die Bedienung ist sehr einfach und alles was Sie benötigen wird durch den folgenden Codeschnipsel erklärt:

```
import numpy as np

num_points = 30
newton_data = np.zeros(shape=(num_points, num_points))
newton_data[3, 5] = 5
```

Zunächst wird das Paket numpy importiert und unter der Abkürzung `np` zur Benutzung bereit gestellt. Die Anzahl der Diskretisierungsschritte setzen wir sowohl in reeller als auch in imaginärer Richtung auf 30 und erzeugen dann mittels `np.zeros` eine Matrix, deren Abmessungen (`shape`) genau dieser Schrittzahl entspricht und jeden einzelnen Wert schon auf 0 gesetzt hat. Einen anderen Wert können Sie den Einträgen mit Hilfe des Operators `[row, col]` unter Angabe der gewünschten Zeile und Spalte zuweisen.

Zu guter Letzt müssen Sie diese Matrix nur noch visualisieren. Dies können Sie, wie schon auf dem vorherigen Übungsblatt erklärt, mit der Funktion `imshow` erledigen. Als guter Satz von Parametern hat sich eine Matrixgröße von 720×720, eine Maximalanzahl von 200 Iterationen pro Punkt, ein Diskretisierungsschritt der Ableitung von $dh = 1e-6$ sowie einer Präzision $\varepsilon = 1e-10$ herausgestellt. Gut bedeutet in diesem Fall, dass das Bild auf einem durchschnittlichen Computer innerhalb von fünf Minuten fertig berechnet sein sollte. Um ihren Algorithmus zu testen können Sie aber zunächst auch mit kleinerer Auflösung rechnen und dann erst am Ende ein höher aufgelöstes Bild generieren.