Computerphysik WS 2015/2016

Prof. Dr. Roland Netz, FU Berlin

Übungsblatt 1: Einführung und Numerische Genauigkeit

Alexander Schlaich, Johann Hansing
22. Oktober 2015

Allgemeine Hinweise

Abgabetermin für die Lösungen ist

- **Sonntag, 25.10., 24:00 Uhr.**

Die Lösungen sollten in Form eines IPython Notebooks (`*.ipynb`) abgegeben werden. Bitte achten Sie dabei auf eine sinnvolle Benennung aus der Ihr Name hervorgeht. Zur Abgabe schicken Sie die Lösungsdatei im Anhang einer Email an Ihren Tutor.

Aufgabe 1.1: Gleitkommastandard nach IEEE 754 (6 Punkte)

Moderne Computer verarbeiten Gleitkommazahlen intern nach dem Standard IEEE 754[1]. Dabei wird eine Gleitkommazahl x dargestellt als Binärzahl der Form

$$\tilde{x} = s \cdot y \cdot B^E, \tag{1}$$

mit dem Vorzeichen s (-1 oder 1), der Mantisse y, der Basis $B = 2$ und dem Exponent E. Dazu definiert das *Institute of Electrical and Electronics Engineers (IEEE)* genaue Verfahren für mathematische Operationen und Rundungen, insbesondere legt es fest, wieviele Bit zum Speichern der Mantisse und des Exponenten für einen bestimmten Datentyp verwendet werden.

- 1.1.1 (2 Punkte) Schreiben Sie ein Python-Programm, das den folgenden Pseudo-Code implementiert:
  - Initialisiere $\epsilon = 1.0$, $\alpha = 1.0$.
  - Solange ($\alpha \neq \alpha + \epsilon$) halbiere ϵ.
  - Gib den Wert von ϵ aus.

  Führen Sie das Programm aus und dokumentieren Sie die Werte von ϵ für verschiedene Startwerte von $\alpha = [1, 1 \cdot 10^{-5}, 1 \cdot 10^{-30}, 5 \cdot 10^{12}]$ in Ihrer Lösungsdatei.
- 1.1.2 (2 Punkte) Welche Bedeutung hat der Wert von ϵ? Welche Bedeutung hat das Verhältnis ϵ/α?
- 1.1.3 (2 Punkte) Kann man ϵ/α auch aus dem IEEE Standard bestimmen? Welcher Wert ergibt sich für eine 32 bit single precision Gleitkommazahl?

[1] siehe `http://de.wikipedia.org/wiki/IEEE_754`

Aufgabe 1.2: Rundungsfehler und Gleitkommaarithmetik (6 Punkte)

Wandelt man eine n-stellige Dezimalzahl x in eine Gleitkommazahl $\tilde{x} = \pm(0{,}z_1 z_2 \dots z_n) \cdot 2^E$ im Binärsystem um, so pflanzt sich der dabei enstehende Rundungsfehler in allen weiteren Berechnungen fort. Um hierfür einen einheitlichen Standard zu schaffen, sind verbindliche Regeln zur internen Rechengenauigkeit, die so genannte *Gleitkommaarithmetik*, festgelegt. Addiert man mehrere Zahlen der gleichen Größenordnung zu einer deutlich größeren Zahl, kann es durch Rundungsfehler zu sogenannten Auslöschungen kommen.

- 1.2.1 (2 Punkt): Beschreiben Sie mit eigenen Worten die Begriffe Gleitkommaarithmetik und Auslöschung. Warum kann numerisch die Reihenfolge der Summation einen Einfluss auf das Ergebnis haben?
- 1.2.2 (2 Punkt): Schreiben Sie ein kurzes Programm, welches die Reihe

$$s_k = \sum_{n=1}^{k} \frac{1}{n^2} \overset{k\to\infty}{\longrightarrow} \frac{\pi^2}{6} \tag{2}$$

  numerisch für $k = 300$ auswertet. Benutzen Sie dabei die untenstehende Funktion `mround(x, n)` um sowohl die Summanden der Reihe als auch in jedem Schritt das Ergebnis der Summe mit einer Genauigkeit von nur $n = 3$ bzw. $n = 5$ Stellen zu berechnen. Wofür wird `mround(x,n)` benötigt?
  Was ist das Ergebnis bei Verwendung von $n = 3$, $n = 5$ und für single precision? Welches Ergebnis erhält man bei Summation in umgekehrter Reihenfolge?
- 1.2.3 (2 Punkt): Schätzen Sie analytisch ab, ab welchem Wert k das Ergebnis s_k für $n = 3$ bzw. $n = 5$ stagniert. Überprüfen Sie Ihr analytisches Ergebnis durch Erweitern Ihres Programms.

```
def mround(x,N):
    if (x==0):
        return x
    return round(x, int(N - math.ceil(math.log10(abs(x)))))
```

Aufgabe 1.3: Taylorentwicklung (8 Punkte)

Für viele Anwendungen ist es nötig, mittels eines Computers schnell und effizient transzendente Funktionen wie beispielsweise sin, cos, exp oder log auszuwerten. Um die Berechnungen zu ermöglichen, werden entweder sog. Nachschlagetabellen verwendet, in denen bereits vorberechnete Werte gespeichert werden, oder es muss ein Näherungsverfahren angewendet werden. Moderne Software-Bibliotheken greifen dafür auf effiziente Algorithmen wie beispielsweise *CORDIC* zurück, die intern eine Koordinatentransformationen mittels Drehmatrizen nutzen.
Eine weitere Alternative, welche wir in dieser Aufgabe anwenden möchten, ist die Verwendung eines Taylorpolynoms:

$$\operatorname{arctanh}(x) = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1} = x + \frac{x^3}{3} + \frac{x^5}{5} + \cdots, \text{ für } -1 < x < 1. \tag{3}$$

- 1.3.1 (3 Punkte): Implementieren Sie in Python eine Funktion `tarctanh(x, N)`, welche das Taylorpolynom der Arcustangens Hyperbolicus-Funktion an einem Punkt x auswertet und den berechneten Wert zurückgibt:

$$\operatorname{tarctanh}_N(x) = \sum_{n=0}^{N} \frac{x^{2n+1}}{2n+1} \tag{4}$$

- 1.3.2 (3 Punkte): Benutzen Sie die Python-Bibliotheken *matplotlib* und *numpy*, um $\text{tarctanh}_N(x)$ im Intervall [0,1[für verschiedene Werte von N darzustellen. Achten Sie auf eine aussagekräftige Ausgabe (Achsenskalierung, Beschriftung, etc.) und vergleichen Sie außerdem mit der Methode `numpy.arctanh(x)`. Für welche Werte N kann die Taylorentwicklung qualitativ den Verlauf der Arcustangens Hyperbolicus-Funktion wiedergeben? Wie hängt die Güte der Näherung von x ab?

- 1.3.3 (2 Punkte): Erweitern Sie Ihr Programm mittels `time.clock()` aus dem Python-Modul `time`, um die Zeit zu Beginn und am Ende der Ausführung der Funktion `tarctanh` zu bestimmen. Lassen Sie die Laufzeit für die verschiedenen Werte von N ausgeben.
  Hinweis: Um zuverlässige Werte für die Laufzeit zu erhalten sollte die Funktion n mal aufgerufen werden und die dabei ermittelte Zeit durch n geteilt werden. Verwenden Sie $n = 25$.

  Welches Verhalten kann man beobachten? Wie verhält sich `numpy.arctanh()`? Wie kann man die Beobachtungen begründen?