Fachbereich Mathematik & Informatik
Freie Universität Berlin
Prof. Dr. Carsten Gräser, Lasse Hinrichsen

2. Übung zur Vorlesung

Computerorientierte Mathematik I

WS 2019/2020

http://numerik.mi.fu-berlin.de/wiki/WS_2019/CoMaI.php

Abgabe: Fr., 15. November 2019, 12:00 Uhr

1. Aufgabe (4 TP)
Führen Sie die folgenden Rechenaufgaben mit Dualzahlen aus, ohne in das Dezimalsystem umzurechnen:

a) $0{,}1100101_2 \cdot 10101{,}111_2 = ?$

b) $\frac{10_2}{110_2} + \frac{101_2}{10100_2} = ?$

2. Aufgabe (4 TP)
Beweisen oder widerlegen Sie die folgenden Aussagen:

a) Jeder endliche Dualbruch ist auch ein endlicher Dezimalbruch.

b) Jeder endliche Dezimalbruch ist auch ein endlicher Dualbruch.

3. Aufgabe (4 PP)
In dieser Aufgabe wollen wir in Python Rundungsfehler bei Festkommazahlen untersuchen. Gehen Sie dabei wie folgt vor:

a) Schreiben Sie eine Funktion `fixedpoint(X)`, die zu einem Vektor `X` von Eingabedaten einen Vektor `Y` desselben Formats zurückgibt. Hierbei soll `Y[i]` diejenige Zahl sein, die man durch kaufmännisches Runden von `X[i]` auf eine Nachkommastelle erhält.

   Hinweis: Sie dürfen hier die von Python bereitgestellten Funktionen zum Runden verwenden.

b) Schreiben Sie eine Funktion `absoluteError(X, Y)`, die zu Vektoren `X` und `Y` den Vektor `errAbs` der absoluten Fehler von `X[i]` zu `Y[i]` berechnet.

c) Schreiben Sie eine Funktion `relativeError(X, errAbs)`, die zu einem Vektor `X` von Eingabedaten und einem Vektor `errAbs` von Fehlern den Vektor `errRel` der relativen Fehler zurückgibt.

d) Sei $\tau = 0{,}001$ und $X = [\tau, 2\tau, 3\tau, \ldots, 100 - \tau, 100]$. Nutzen Sie die auf der Vorlesungswebsite bereitgestellte Funktion `plotErrors.py` sowie Ihre Funktionen aus den vorangegangenen Unteraufgaben, um eine Skriptdatei `run_2_3.py` zu schreiben, die die absoluten und relativen Fehler von `X` und `fixedPoint(X)` bezüglich verschiedener Skalierungen plottet. Was beobachten Sie und welche Skalierungen sind dafür am sinnvollsten? Schreiben Sie Ihre Antwort in die Datei `beobachtungen.txt`.

Hinweis: Wenn die Datei `plotErrors.py` im selben Verzeichnis wie Ihr Skript liegt, können Sie die Funktion `plotErrors(...)` mittels `from plotErrors import *` einbinden.

4. Aufgabe (4 PP)
Informieren Sie sich zu der Python Bibliothek MATPLOTLIB und schreiben Sie anschließend eine Funktion `plotGraphics()`, die bei Aufruf die folgenden Grafiken erstellt:

a) Einen Linienplot von $\sin(x)$ für $x \in [0, 2\pi]$ mit Titel „Plot der Sinusfunktion“, der als PNG-Datei `plot1.png` abgespeichert wird.

b) Einen Punktplot der Funktion `np.floor(x)` für $x \in [0, 10]$, der als PNG-Datei `plot2.png` abgespeichert wird. (Gemeint ist, dass die Datenpunkte, die für das Plotten verwendet werden, nicht durch eine Linie verbunden werden, sondern als Punkte/Kreuze/Kreise dargestellt werden sollen.)

c) Einen Linienplot von x^2 für $x \in [0, 10]$, wobei sowohl x- als auch y-Achse logarithmisch skaliert sind. Die Grafik wird als PNG-Datei `plot3.png` abgespeichert.

Hinweis: Sehen Sie sich die Funktionen `plt.semilogx`, `plt.semilogy` und `plt.loglog` an.

d) Ein Linienplot, der für $x \in [0, 2\pi]$ die Graphen von $\sin(x)$ in Rot, von $\cos(x)$ in Blau und von $\frac{1}{\pi^2} x \cdot (2\pi - x)$ in Gelb in einer einzigen Abbildung inklusive einer Legende in der oberen rechten Ecke darstellt. Die Grafik soll als PNG-Datei `plot4.png` abgespeichert werden.

Wählen Sie dabei beim Plotten immer ausreichend Auswertungspunkte, damit der Graph der geplotteten Funktion „sinnvoll“ dargestellt wird.

Allgemeine Hinweise

Die Punkte unterteilen sich in Theoriepunkte (TP) und Programmierpunkte (PP). Bitte beachten Sie die auf der Vorlesungshomepage angegebenen Hinweise zur Bearbeitung und Abgabe der Übungszettel, insbesondere der Programmieraufgaben.