Übungsserie 2

Abgabe: gemäss Angaben Dozent

Erzeugen Sie für jede der folgenden Aufgaben wie beschrieben eine Datei *Name_S2_AufgX* (S2 steht für die Serie 2, X ist die Aufgabennummer), fassen Sie diese in eine ZIP-Datei *Name_S2.zip* zusammen und laden Sie dieses File vor der nächsten Übungsstunde nächste Woche auf Moodle hoch. Die Python-Files müssen ausführbar sein und in den Kommentarzeilen (beginnen mit #) soll bei Funktionen ein Beispiel eines funktionierenden Aufrufs angegeben werden. Verspätete Abgaben können nicht mehr berücksichtigt werden.

Aufgabe 1 (ca. 20 Minuten):

Lösen Sie die folgende Aufgabe manuell auf einem Blatt Papier und scannen Sie dieses in die Datei *Name_S2_Aufg1.pdf:*

a) Bestimmen Sie die Anzahl verschiedener Maschinenzahlen auf einem Rechner, der 15-stellige Gleitpunktzahlen mit 5-stelligen Exponenten sowie dazugehörige Vorzeichen im Dualsystem verwendet.

b) Geben Sie die Maschinengenauigkeit einer Rechenmaschine an, die mit 16-stelliger Dezimalarithmetik arbeitet.

c) Gegeben seien zwei verschiedene Rechenmaschinen. Die erste davon arbeite mit einer 52-stelligen Binärarithmetik (entspricht double Precision im IEEE Format) und die zweite mit einer 14-stelligen Hexadezimalarithmetik. Welche Maschine rechnet genauer? (Mit Begründung!)

Aufgabe 2[1] (ca. 40 Minuten):

Erzeugen Sie ein ausführbares Skript *Name_S2_Aufg2.py*, welches die folgenden Aufgaben lösen soll:

a) Zeichnen Sie die beiden Polynome

$$\begin{aligned} f_1(x) &= x^7 - 14x^6 + 84x^5 - 280x^4 + 560x^3 - 672x^2 + 448x - 128 \\ f_2(x) &= (x-2)^7 \end{aligned}$$

auf dem Intervall $x \in [1.99,\, 2.01]$ mit 501 äquidistanten Punkten. Was stellen Sie fest? Analytisch sind die beiden Polynome identisch aber weshalb weichen die beiden von Python gezeichneten Polynome voneinander ab? Schreiben Sie Ihre Antwort als Kommentar in Ihr Skript.

b) Plotten Sie die Funktion

$$g(x) = \frac{x}{\sin(1+x) - \sin(1)}$$

auf dem Intervall $x \in [-10^{-14}, 10^{-14}]$ mit einer Schrittweite von 10^{-17}. Ist die numerische Berechnung des Grenzwertes $\lim_{x\to 0} g(x)$ auf diese Weise stabil?

c) Formen Sie manuell den Nenner von $g(x)$ unter Zuhilfenahme des Additionstheorems $\sin(a) - \sin(b) = 2\cos\left(\frac{a+b}{2}\right)\sin\left(\frac{a-b}{2}\right)$ um und plotten Sie diese 'neue' Funktion nochmals auf dem gleichen Intervall wie b). Was stellen Sie fest und weshalb? Welcher Wert nimmt $\lim_{x\to 0} g(x)$ also an? Schreiben Sie Ihre Antwort als Kommentar in Ihr Skript.

[1] Beispiele übernommen aus 'Numerik-Algorithmen', G. Engeln-Müllges, K. Niederdrenk, R. Wodicka

Aufgabe 3 (ca. 40 Minuten):

Erzeugen Sie ein ausführbares Skript *Name_S2_Aufg3.py*, welches die folgenden Aufgaben lösen soll:

Bereits Archimedes entwickelte einen Algorithmus, um die Kreiszahl π zu bestimmen. Er verwendete dafür die Idee, einem Kreis gleichförmige Vielecke einzuschreiben. Wir betrachten eine leicht abgeänderte Variante dieses Algorithmus.

a) Gehen wir zum Beispiel von einem dem Einheitskreis eingeschriebenen Sechseck aus (mit also $n = 6$ auf dem Kreis liegenden Ecken). Dieses Sechseck setzt sich zusammen aus sechs gleichseitigen Dreiecken mit der Seitenlänge $s_6 = 1$. Summiert man die Länge der Seiten, die den Kreis berühren, erhält man $6 \cdot 1 \approx 2\pi = 6.28...$. Je grösser die Anzahl der Ecken eines solchen 'Vielecks', umso genauer muss sich die Summe der Seitenlängen 2π annähern. Es lässt sich zeigen, dass für die Seitenlänge eines Vielecks mit $2n$ Ecken gilt:

$$s_{2n} = \sqrt{2 - 2\sqrt{1 - \frac{s_n^2}{4}}}$$

Implementieren Sie einen Algorithmus, der Ihnen ausgehend von $n = 6$ die Summe der Seitenlänge $2n \cdot s_{2n}$ berechnet. Plotten Sie $2n \cdot s_{2n}$ als Funktion von $2n$. Was passiert für grosse n und weshalb?

b) Der Ausdruck s_{2n} lässt sich umschreiben unter Verwendung von

$$a^2 - b^2 = (a + b)(a - b) \text{ bzw. } a - b = \frac{a^2 - b^2}{a + b}$$

als

$$s_{2n} = \sqrt{\frac{s_n^2}{2(1 + \sqrt{1 - \frac{s_n^2}{4}})}}$$

Berechnen sie 2π erneut mit dieser Varianten. Was beobachten Sie?

Aufgabe 4 (ca. 20 Minuten):

Erzeugen Sie ein ausführbares Skript *Name_S2_Aufg4.py*, welches die folgenden Aufgaben lösen soll:

- Überlegen Sie sich einen kurzen iterativen Algorithmus, der die Maschinengenauigkeit eps Ihres Rechners berechnet. Schliessen Sie aus dem Ergebnis, mit welcher Stellenzahl er operiert. Nähern Sie dazu die Maschinengenauigkeit als die kleinste positive Maschinenzahl an, für die $1 + eps > 1$ gilt.
- Berechnen Sie analog die grösstmögliche positive Maschinenzahl q_{max} vom Typ 'float', für die noch $1 + q_{max} > q_{max}$ gilt. Lässt sich q_{max} direkt aus eps berechnen?