WS23/24: Numerische Mathematik
Übungszettel 1

Kreuzerlliste. Kreuzerln Sie bis 1 Stunde vor dem Übungstermin auf der Moodle Seite all jene Beispiele an, die Sie gelöst haben. Bei Nichterscheinen zur Übung verfallen Ihre Kreuzerl.

Abgabe Programmierbeispiele. Laden Sie bis 1 Stunde vor dem Übungstermin auf der Moodle Seite alle Programmierbeispiele in **einem** Jupyter Notebook pro Person und Termin hoch.

Gemeinsames Arbeiten. Gemeinsames Arbeiten an den Beispielen ist erlaubt und erwünscht. Voraussetzung zum Kreuzerln ist, dass Sie alle Schritte verstanden haben und erklären können. Für Programmierbeispiele müssen Sie eine eigene Codeversion erstellt haben, ausführen, live modifizieren und erklären können.

Programmieren. Verwenden Sie Jupyter Notebooks (Python). Schreiben Sie Ihre Programme selbst. Erstellen Sie "sauberen" Code, i.e. wählen Sie sinnvolle Variablennamen, verwenden Sie erklärende Kommentare und beschriften Sie ihre Plots (Titel, Achsen, Legenden, etc.).

---

1. Finden Sie heraus, was die Zahlen 32 und 64 in den IEEE 754 Standards *binary32* and *binary64* genau bedeuten und wie sie sich ergeben.

2. (P) Mehr Floating Point Arithmetik.

   (a) Lassen Sie sich von Python das machine epsilon für single und double precision ausgeben.

   (b) Was bekommen Sie in Python für folgende Rechnungen:

   $$a = (10 - 10) - 10^{-16}, \quad b = 10 - (10 + 10^{-16}).$$

   Was ist passiert?

   (c) Finden Sie die (möglicherweise unendlichen) Binärdarstellungen von 0.2 und 0.5 (also Exponent und Stellen) und erklären Sie damit die Resultate in Python für `0.1*n+0.1*n+0.1*n==0.3*n` für $n = 2$ und $n = 5$?)

   (d) Was bekommen Sie in Python für folgende Eingaben: `10**16+1` und `1.0*10**16+1` Was ist passiert?

3. Floating Point Arithmetik.

   (a) Es sei die Basis $\beta = 3$, Präzision $t = 2$ und minimaler und maximaler Exponenten $e_{\text{min}} = -1$ und $e_{\text{max}} = 3$. Finden Sie F, die Menge aller positiven normalisierten F_N und subnormalisierten Zahlen F_S in diesem System. Schreiben Sie alle Element von $F = F_N \cup F_S$ auf. Was ist Ihre größte darstellbare ganze Zahl $N \in F$, für die $N + 1 \notin F$?

(b) Für allgemeines β und t: Angenommen Sie bilden reelle Zahlen $x \in [\mathsf{min}(F_N), \mathsf{max}(F_N)]$ durch runden zum nächstgelegenen Element in Ihr System ab, $z \mapsto \mathsf{fl}(z) \in F$. Leiten Sie eine Abschätzungen für den maximalen relativen Fehler her.

(c) (P) Schreiben Sie Code, der Ihnen "reelle" Zahlen durch Runden von $x \in [\mathsf{min}(F_N), \mathsf{max}(F_N)]$ nach F definiert wie in (a) abbildet. Überprüfen Sie Ihre Abschätzungen aus (b) numerisch, in dem Sie viele (zB 1000) Zahlen runden und den relativen Fehler darstellen.

4. (P) As you know, for differentiable functions f holds

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}.$$

Thus, to numerically approximate a derivative of a function for which the derivative is hard to derive analytically, one can use, for a small h, the approximation

$$f'(x_0) \approx f'_{\mathsf{num}}(x_0) := \frac{f(x_0 + h) - f(x_0)}{h}.$$

Compute approximations of the derivative of the function

$$f(x) = \frac{\exp(x)}{\cos(x)^3 + \sin(x)^3}$$

at $x_0 = \pi/4$. In order to do so, use progressively smaller perturbations $h = 10^{-k}$ for $k = 1, \ldots 16$. Present the errors in the resulting approximation in a log-log plot, i.e., use a logarithmic scale to plot the values of h on the x-axis, and a logarithmic scale to plot the errors between the finite difference approximation $f'_{\mathsf{num}}(x_0)$ and the exact value, which is (rounded) $f'(x_0) = 3.101766393836051$, on the y-axis. What do you observe as h becomes smaller, and for which h do you get the best approximation to the derivative? Discuss.

5. Consider $z = \sqrt{2}$

(a) What will z be rounded to in a FPA system with $\beta = 10$, $t = 3$, $e = \{-1, 0, 1, 2\}$? Compare the absolute and relative rounding errors to the maximal rounding errors you would expect.

(b) What will z be rounded to in a FPA system with $\beta = 2$, $t = 3$, $e = \{-1, 0, 1, 2\}$ Compare the absolute and relative rounding errors to the maximal rounding error you would expect.

Another question on the next page ...

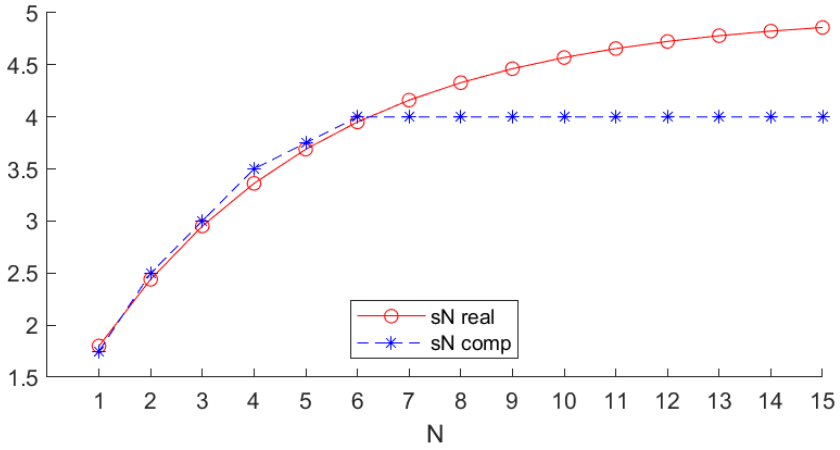

Figure 1

6. Für die geometrische Reihe gilt bekanntlich

$$s_N = \sum_{i=0}^{N} r^i = \frac{1 - r^{N+1}}{1 - r}.$$

Sie berechnen s_N rekursiv in einem FPA System mit $\beta = 2$, $t = 4$, $e_{\text{min}} = -1$, $e_{\text{max}} = 3$ für $r = 0.8$ und finden folgende Diskrepanz zwischen der berechneten Folge (sN comp) und den analytischen Werten (sN real), siehe Abbildung. Was genau ist passiert? Warum ändert sich das Verhalten genau ab $N = 6$ (*Hinweis:* Sie dürfen den Computer zu Hilfe nehmen)?