EXERCISE SET 2 – MATH-250 Advanced Numerical Analysis I

The exercise sheet is divided into two sections: quiz and exercises. The quiz will be discussed in the beginning of the lecture on Thursday, March 6. The exercises marked with ($\star$) are graded homework. The exercises marked with **(Python)** are implementation based and can be solved in the Jupyter notebooks which are available on Moodle/Noto. **The deadline for submitting your solutions to the homework is Friday, March 7 at 10h15.**

Quiz

(a) Consider the harmonic series $\sum_{k=1}^{\infty} \frac{1}{k}$, which is known to diverge. When attempting to compute the partial sum $1 + 1/2 + 1/3 + \ldots + 1/n$ (from the smallest to the largest) in double precision, what will happen as $n \to \infty$?

- ☐ The computed partial sums will overflow.
- ☐ The computed partial sums will stagnate ("converge") to ≈ 34.
- ☐ The computed partial sum will stagnate ("converge") to $\approx 2 \times 10^{16}$.
- ☐ The computed partial sum will stagnate ("converge") to $\approx 10^{300}$.

(b) Consider the same question for the alternating harmonic series $\sum_{k=1}^{\infty} (-1)^{k-1} \frac{1}{k}$, which is known to converge to $\log(2)$.

- ☐ The computed partial sums will overflow.
- ☐ The computed partial sums will stagnate ("converge") to $\approx \log(2)$.
- ☐ The computed partial sums will underflow.
- ☐ The computed partial sum will stagnate ("converge") to ≈ 0.

Exercises

Problem 1. Let $\mathbb{F}_1 = \mathbb{F}(2, 24, -126, 127)$ denote the set of single precision floating point numbers and let $\mathbb{F}_2 = \mathbb{F}(2, 53, -1022, 1023)$ denote the set of double precision floating point numbers. Consider an adjacent pair $x, y \in \mathbb{F}_1$ with $x, y \neq 0$, that is $x < y$ and $\nexists z \in \mathbb{F}_1$ such that $x < z < y$. How many distinct elements of $\mathbb{F}_2$ are between x and y?

Problem 2. Derive the smallest and largest positive elements in $\mathbb{F}(2, 8, -126, 127)$.[1]

Problem 3. Consider the set of floating point numbers with no constraints on the exponent, that is[2]

$$\mathbb{F} = \left\{ x = \pm \frac{m}{\beta^t} \beta^e : m \in \mathbb{N}, \beta^{t-1} \leq m \leq \beta^t - 1, e \in \mathbb{Z} \right\} \cup \{0\}$$

[1] $\mathbb{F}(2, 8, -126, 127)$ is known as bfloat16, which is used in, for example, Google cloud TPUs.

[2] By setting a limit on the exponent e you will create a finite subset of $\mathbb{F}$, which is then used in computers. By restricting $e_{\min} \leq e \leq e_{\max}$ you will get the set $\mathbb{F}(\beta, t, e_{\min}, e_{\max})$.

where β is the base and t the precision. Compute the smallest element in $\mathbb{N}$ not part of $\mathbb{F}$.

Problem 4. (Python) The sample variance s_n of n numbers $x_1, \ldots, x_n \in \mathbb{R}$ is given by

$$s_n^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2, \tag{1}$$

where $\overline{x}$ denotes the sample mean

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

Computing s_n^2 with formula (1) requires two passes through the data; one to compute $\overline{x}$ and the other to accumulate the sum of squares. A two-pass computation is undesirable for large data sets. An alternative formula, found in many statistics textbooks, requires only a single pass through the data:

$$s_n^2 = \frac{1}{n-1} \left(\sum_{i=1}^{n} x_i^2 - \frac{1}{n} \left(\sum_{i=1}^{n} x_i \right)^2 \right). \tag{2}$$

Compute the sample variance s_n of the data $x = [100'000'000, 100'000'001, 100'000'002]$ with both formulae. What do you notice? Explain.

Problem 5. Consider complex numbers $x = a + \mathrm{i}b$, $y = c + \mathrm{i}d$ with real floating point numbers $a, b, c, d \in r(\mathbb{F})$. Suppose that their product xy is computed according to the usual definition of complex multiplication. Show that in the standard model of rounding (Definition 1.14) it follows that the computed result $\mathrm{fl}(xy)$ satisfies

$$\mathrm{fl}(xy) = xy(1+\delta), \quad |\delta| \leq \sqrt{2} \cdot \gamma_2(\mathbb{F})$$

with $\gamma_2(\mathbb{F})$ defined as in Lemma 1.16.

($\star$) Problem 6. (Python) Consider quadratic polynomials of the form $q + px + x^2$ for $p, q \in \mathbb{R}$. The formula to compute the roots x_1 and x_2 of the polynomial is

$$x_{1,2} = -\frac{p}{2} \pm \sqrt{\left(\frac{p}{2}\right)^2 - q}. \tag{3}$$

(a) Using (3), write a Python function `roots(p, q)` which computes the roots

$$\hat{x}_1, \hat{x}_2 = \texttt{roots(p, q)}$$

given p and q.

(b) Test your `roots` function for the three polynomials

$$p_1(x) = 12 + 8\,x + x^2,$$
$$p_2(x) = 1 - 1000000000.000000001\,x + x^2, \text{ and}$$
$$p_3(x) = 1 + (2^{31} + 2^{-31})\,x + x^2$$

and clearly display the roots $\hat{x}_1$ and $\hat{x}_2$ for each polynomial.

Hint: Assure that you used the same order of summands as specified here. Otherwise, your results may differ.

(c) Evaluate the polynomials p_1, p_2, and p_3 for the roots $\hat{x}_1$ and $\hat{x}_2$ you computed using your `roots` function in (b) and clearly display the values you obtained.

(d) Given the exact roots for the polynomials p_1, p_2, and p_3 in Figure 1, write a Python function `rel_error` that computes

$$\varepsilon = \frac{|x - \hat{x}|}{|x|},$$

calculate the relative errors for $(\hat{x}_1, x_1)$ and $(\hat{x}_2, x_2)$ for each polynomial, and clearly display them.

	p_1	p_2	p_3
x_1	-2	10^9	-2^{-31}
x_2	-6	10^{-9}	-2^{31}

Figure 1: Exact roots for the polynomials p_1, p_2, and p_3

(e) (**This is for your understanding and will not be graded.**) Explain why for two of the polynomials one of their roots is not well approximated. Explain the role of the sign of p into determining which root will be *well-approximated* by the quadratic formula (3).

Remember to upload the completed Jupyter notebook `homework02.ipynb` corresponding to the homework to the submission panel on Moodle until Friday, March 7 at 10h15. To download your notebook from Noto, use `File > Download`. Only your submissions to Moodle will be considered for grading.