SOLUTION 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 sum will stagnate ("converge") to $\approx 2 \times 10^{16}$.
- ■ The computed partial sums will stagnate ("converge") to ≈ 34.
- ☐ 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 sum will underflow.
- ■ The computed partial sums will stagnate ("converge") to $\approx \log(2)$.
- ☐ The computed partial sum will stagnate ("converge") to ≈ 0.

Solution.

(a) Let $s_n = \sum_{k=1}^{n} \frac{1}{k}$. Then

$$s_n = \sum_{k=1}^{n} \frac{1}{k} = s_{n-1} + \frac{1}{n}.$$

At some point, s_{n-1} (which is larger than 1 and grows with n) will be so much larger than $\frac{1}{n}$, that the result of their sum suffers from a ronudoff error. Formally, for some $n^* \in \mathbb{N}$, all $n > n^*$ will satisfy

$$s_n = \mathrm{fl}(s_{n-1} + \frac{1}{n}) = s_{n-1}.$$

That is, the partial sums s_n will remain constant for all $n > n^*$.

Since $s_n \geq 1, \forall n$, in double precision this will at latest happen when $\frac{1}{n^*} \approx \varepsilon_M \implies n^* \approx 5 \times 10^{15}$. So

$$s_{n^*} = \sum_{k=1}^{n^*} \frac{1}{k} \ll \sum_{k=1}^{n^*} 1 = n^* \approx 5 \times 10^{15}.$$

This excludes all answer possibilities except one.

(b) In a similar manner to question (a), it can again be argued that the partial sums will remain constant after a certain $n^* \in \mathbb{N}$. Due to the convergence properties of the alternating harmonic series, the partial sum will already be close to the value $\log(2)$ once this happens.

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?

Solution. We consider a generic point x in $[\beta^e, \beta^{e+1})$, with $\beta = 2$, $e \in (-126, 127)$. The distance Δ between two adjacent points (also called *spacing*) is given by $\Delta = \beta^e(0.\underbrace{0 \cdots 0}_{t-1 \text{ zeros}} 1)_\beta = \beta^e \beta^{-t} = \beta^{e-t}$, where t is number of digits considered. In the case of $\mathbb{F}_1$, $\Delta_1 = \beta^{e-24}$, and in the case of $\mathbb{F}_2$, $\Delta_2 = \beta^{e-53}$. Since $\mathbb{F}_1 \subset \mathbb{F}_2$ for all $e \in (-126, 127)$, then both x and $x + \Delta_1 x$ belong to $\mathbb{F}_2$. So the number of elements n of $\mathbb{F}_2$ in the interval $(x, x + \Delta_1 x]$ is equal to

$$n\Delta_2 = \Delta_1 \Rightarrow n = 2^{53-24} = 2^{29}.$$

Since $x + \Delta_1 x$ belongs to $\mathbb{F}_1$, it should not be counted, and thus between an adjacent pair of non-zero elements in $\mathbb{F}_1$, there are $n - 1 = 2^{29} - 1$ non-zero elements of $\mathbb{F}_2$.

It is also possible to answer this question by taking a probabilistic point of view. Indeed, numbers if $\mathbb{F}_1$ have 24 digits (that may be either 0 or 1) while numbers in $\mathbb{F}_2$ have 53 digits. Consider x and $x + \Delta_1 x$, an adjacent pair of non-zero elements in $\mathbb{F}_1$. Both x and $x + \Delta_1 x$ also belong to $\mathbb{F}_2$ since we can write them with the same 24 first digits as in $\mathbb{F}_1$ and then we append $53 - 24 = 29$ zero digits to them. All the numbers of $\mathbb{F}_2$ in $[x, x + \Delta x)$ thus have the same first 24 digits as x, and any possible combination of the last 29 digits, which gives 2^{29} possible combinations (since a digit is, in this case, either 0 or 1). However, since $x \in \mathbb{F}_1$ and corresponds to the combination with all zeros, we do not want to count it. Consequently, there are $2^{29} - 1$ non-zero elements of $\mathbb{F}_2$ between x and $\Delta_1 x$, that is between an adjacent pair of non-zero elements in $\mathbb{F}_1$.

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

Solution. For this exercise, we refer to Lemma 1.8 of the lecture notes. A generic element in $\mathbb{F}(\beta, t, e_{min}, e_{max})$ can be written as $\pm\beta^e \sum_{i=1}^{t} \frac{d_i}{\beta^i}$, where $d_i \in 0, 1, ...\beta - 1, d_1 \neq 0$ and $e \in \{e_{min}, e_{min} + 1, \ldots, e_{max}\}$.

- The largest (in magnitude) element of the set is obtained when the exponent e equals

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

to e_{max} and when all the digits coincide with $\beta - 1$, that is

$$\text{largest} = \beta^{e_{max}} \left(0.(\beta-1)(\beta-1)...(\beta-1)\right)_\beta$$
$$= \beta^{e_{max}} \sum_{i=1}^{t} (\beta-1)\beta^{-i} = \beta^{e_{max}}(1-\beta^{-t}).$$

- The smallest element is instead obtained when the exponent e is equal to $e = e_{min}$ and when all the digits are 0 but the first one. The first digit must be 1 (since it has to be greater than 0). We thus have:

$$\text{smallest} = \beta^{e_{min}} \left(0.100...0\right)_\beta = \beta^{e_{min}-1}.$$

In our case we are considering $\mathbb{F}(2, 8, -126, 127)$. Therefore:

$$\text{largest} = \beta^{e_{max}}(1-\beta^{-t}) = 2^{127}(1-2^{-8}); \qquad \text{smallest} = \beta^{e_{min}-1} = 2^{-127}$$

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

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

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

Solution.

1. We notice that the spacing Δ between a generic point $x \in \mathbb{F} \cap [\beta^e, \beta^{e+1})$ and the next element in the set is given by $\Delta = \beta^{e-t}$ (see Problem 1). We observe that:

$$\Delta > 1 \iff e - t > 0 \iff e > t.$$

2. All the natural numbers up to β^t belong to the set, indeed:

$$m\beta^0 = m \in \mathbb{F}, \ \forall m \in \mathbb{N}, \ \beta^{t-1} \leq m \leq \beta^t - 1$$

   while β^t clearly belongs to the set.

3. We finally observe that $m\beta \in \mathbb{F}$ for $m \in \mathbb{N}$, $\beta^{t-1} \leq m \leq \beta^t - 1$, but in this interval the spacing between two adjacent elements is bigger than 1 by point 1. Thus, β^t belongs to this interval and the next elements is given by $\beta^t + \Delta > \beta^t + 1$. Therefore $\beta^t + 1$ does not belong to the set, since it is skipped, and by point 2, it is the smallest positive integer that does not belong to $\mathbb{F}$.

[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 $\mathbb{F}(\beta, t, e_{\min}, e_{\max})$.

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

Solution. Available in the Jupyter notebook `serie02-sol.ipynb` on Moodle.

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.

Solution. In the following, δ_i denotes a number bounded by $|\delta_i| \leq u$ and $|\theta_2|$, $|\theta_2'|$, $|\theta_2''|$, $|\theta_2''|\leq \gamma_2(\mathbb{F})$ (see Lemma 1.16).

$$\begin{aligned}\mathrm{fl}(xy) &= (ac(1+\delta_1) - bd(1+\delta_2))(1+\delta_3) + \mathrm{i}(ad(1+\delta_4) + bc(1+\delta_5))(1+\delta_6)\\ &= ac(1+\theta_2) - bd(1+\theta_2') + \mathrm{i}(ad(1+\theta_2'') + bc(1+\theta_2'''))\\ &= xy + e,\end{aligned}$$

where

$$\begin{aligned}|e|^2 &\leq \gamma_2(\mathbb{F})^2((|ac|+|bd|)^2 + (|ad|+|bc|)^2)\\ &\leq 2\gamma_2(\mathbb{F})^2(a^2+b^2)(c^2+d^2)\\ &= 2\gamma_2(\mathbb{F})^2|xy|^2\end{aligned}$$

which shows the claim.

($\star$) **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).

Solution. Available in the Jupyter notebook `homework02-sol.ipynb` on Moodle.

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.