

EXERCISE SET 1 – 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, February 27. 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, February 28 at 10h15.**

Quiz

(a) How many distinct elements are contained in $\mathbb{F}(2, 3, -1, 1)$?

☐ 17

☐ 49

☐ 25

☐ 129

(b) Let $\mathbb{F} = \mathbb{F}(\beta, t, e_{\min}, e_{\max})$ be a set of floating point numbers in the sense of Definition 1.6. Is the following statement true? If $a, b \in \mathbb{F}$, then $a + b \in \mathbb{F}$.

☐ True

☐ False

Exercises

Problem 1. (Python)

(a) Write a function `sequence` such that $x_n = \text{sequence}(n, C)$ for $n \in \mathbb{N}$, where

$$\begin{aligned} x_1 &= 2025 \\ x_{n+1} &= \frac{1}{2} \left(x_n + \frac{C}{x_n} \right) \end{aligned}$$

for some $C \in \mathbb{R}$.

(b) For $C = 2$, compute x_{20} . What do you notice?

Problem 2. (Python) Consider the function

$$f(x) = \frac{x^2}{2} \sin(x), \quad x \in [1, 20] \tag{1}$$

Write Python code to visualize the function $f(x)$ using 10, 20, and 100 evenly spaced points, respectively. Plot the three graphs inside the same figure using different colors. Which is the most appealing graphical representation of $f(x)$?

Hint: You can use the NumPy function `numpy.linspace/np.linspace` to generate evenly spaced points (see `numpy.linspace` in the NumPy documentation).

Problem 3. (Python) The Fibonacci numbers are recursively defined as:

$$\begin{aligned} f_0 &= 0, \\ f_1 &= 1, \\ f_n &= f_{n-1} + f_{n-2}, \quad n > 1. \end{aligned}$$

- (a) Write a function `fibonacci(n)` that, given a positive number n , returns a vector `f` containing the first $n + 1$ Fibonacci numbers f_0, f_1, \dots, f_n .
Program efficiently: it should, for example, not take longer than a couple of milliseconds to calculate `fibonacci(60)`.

- (b) Write a function `fiboquots(n)` that returns the vector `q` with the elements

$$\frac{f_2}{f_1}, \frac{f_3}{f_2}, \dots, \frac{f_n}{f_{n-1}}.$$

Hint: Do not use any loops. Instead, use `/` (look for `numpy.divide` in the NumPy documentation).

- (c) Using the function `matplotlib.pyplot.semilogy/plt.semilogy` in Python, and the previously constructed `fiboquots(n)`, plot the values

$$\left(n, \left| \frac{f_n}{f_{n-1}} - \frac{1 + \sqrt{5}}{2} \right| \right), \quad \text{for } n = 2, \dots, 60.$$

Based on the graph, try to understand what happens for large n .

Problem 4. (Python) Write a single function that computes the following three outputs for two vectors x and y having the same length:

1. The element-wise product of the two vectors x and y ;
2. The scalar product of x and y ;
3. A vector v for which the components are

$$v_1 = x_1 y_n, \quad v_2 = x_2 y_{n-1}, \quad \dots, \quad v_{n-1} = x_{n-1} y_2, \quad v_n = x_n y_1.$$

The function should be implemented as follows:

```

1 | def operations(x,y):
2 |     """
3 |     ElProd is the elementwise product of two vectors x and y
4 |     NOTE: x and y can be row or column vectors.
5 |
6 |     ScalProd is the scalar product of x and y
7 |
8 |     v is the vector defined as:
9 |     v(1) = x(1)y(n)
10 |    v(2) = x(2)y(n-1)
11 |    ....

```

```

12 |     v(N) = x(N)y(1)
13 |     """
14 |     ### YOUR CODE HERE
15 |
16 |
17 |
18 |     return ElProd, ScalProd, v

```

Problem 5. (Python) The Taylor series of the exponential is defined as

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

- (a) Create a function `myexp(x)` which, given a real number x , returns an approximate value for e^x by summing the Taylor series until the ratio between the next element in the sequence and the partial sum falls below 10^{-16} . Compute `myexp(30)`, `myexp(10)`, `myexp(1)`, `myexp(-20)`, `myexp(-40)`. Compare these values with the values returned by the function `numpy.exp/np.exp` for the same arguments. What do you see? Can you explain why the differences occur?

Hint: $n!$ can be computed with the function `math.factorial(n)`.

- (b) Modify your function `myexp` so that you save the values of the partial sums of the Taylor series in a vector `p`. For $x = 1$, plot the approximation error $|p_n - e^x|$, $n = 0, 1, \dots$ throughout the iterations using the function `matplotlib.pyplot.semilogy/plt.semilogy`.

(★) **Problem 6. (Python)** Newton's method is an algorithm which can be used to approximate a root (or zero) of a real-valued function. Given an initial guess x_0 , the approximation is generated by the sequence

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 1, 2, \dots$$

The method stops when

$$|f(x_n)| < \text{tol}.$$

Consider the function

$$f(x) = \sin\left(\frac{1}{e^x + 1}\right) - \frac{\log(x+1) - \pi}{10}.$$

- (a) Write a Python function `func(x)` such that $f(x) = \text{func}(x)$. Compute and print the vector

$$\mathbf{a} = [f(0.5) \quad f(1.7) \quad f(2.1) \quad f(4.5)]$$

- (b) Differentiate f to get f' . Write a Python function `dfunc(x)` such that $f'(x) = \text{dfunc}(x)$. Compute and print the vector

$$\mathbf{b} = [f'(0.5) \quad f'(1.7) \quad f'(2.1) \quad f'(4.5)].$$

- (c) Plot the value of $f(x)$ at 1000 evenly spaced values of x in the interval $[0, 100]$.
- (d) Write a Python function `newton(func, dfunc, x0, tol, nmax)` which implements Newton's method and returns the approximation of the root of the function `func` with derivative `dfunc`. `x0` is the initial guess, `tol` is the tolerance, and `nmax` the maximum number of iterations.

If after `nmax` iterations we don't have $|f(x_n)| < \text{tol}$ or if at any point $x_n \notin (-1, 100)$, terminate the algorithm, return the current value x_n , and print "no convergence".

Fix `tol = 1e-6` and `nmax=500` and write Python code which prints the output of `newton(func, dfunc, x0, tol=1e-6, nmax=500)` for `x0 = 1, 2, ..., 6`.

Remember to upload the completed Jupyter notebook `homework1.ipynb` corresponding to the homework to the submission panel on Moodle until Friday, February 28 at 10h15. To download your notebook from Noto, use File > Download. Only your submissions to Moodle will be considered for grading.