MAT101 Programming – Homework 8

Deadline: Monday, 27.11.2023, 22:00 PM

Login to `https://w3.math.uzh.ch/my` with your UZH credentials to submit your solved exercises for grading. You can find more information on how to upload/submit your exercises on `https://wiki.math.uzh.ch/public/studentUpload`.

For submission, please upload **at most 1** Python file **per exercise**. You could even just upload 1 Python file for the whole exercise sheet. You can use comments and/or print statements to answer non-programming tasks.

This exercise sheet is intended to make you familiar with `Matplotlib`, so if you have not so already, install `Matplotlib` using pip or any python package handler of your choice.

Exercise 1. **25** P.

To make a plot, we need something to plot. For this exercise we are going to implement two ways to approximate e^x and then visualise them in a plot.

a) Write a function `approx_e_limit(x, n)` which approximates e^x using the limit: **5** P.

$$e^x = \lim_{k\to\infty}\left(1+\frac{x}{k}\right)^k \approx \left(1+\frac{x}{n}\right)^n$$

b) Write a function `approx_e_sum(x, n)` which approximates e^x using the sum: **5** P.

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

Now we want to plot the approximations we get when using the two functions we just defined.

c) Write a function `plot_e_approximations(x, k)` which creates a plot displaying the approximations given by `approx_e_limit` and `approx_e_sum` for all n in the range [0, k]. For reference add a vertical line at the height of e^x calculated for example with `numpy.exp(x)` or `math.exp(x)`. Show and save the plot. **10** P.

d) To make clear what the plot is showing, make sure to include the following:

i) A title. **1** P.

ii) Labels for the x-axis and y-axis. **2** P.

iii) A legend stating what the different lines show. **2** P.

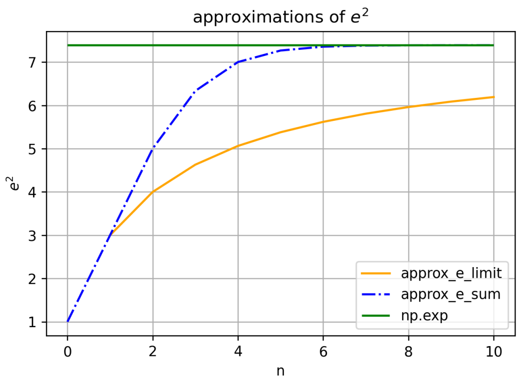

Figure 1: Example plot for `plot_e_approximations(2, 10)`

Note: If you do not manage to get a working implementation for a) and/or b), you can still get full credit for c) by using another “data set” to plot instead.

Exercise 2. **15** P.

a) Write a function `plot_subplots(x_min, x_max)` which creates a figure with four subplots arranged in a square. In each subplot plot e^x in the range [x_min, x_max], however: the upper left plot should have linearly scaled axes; the upper right plot should have a linear y-axis but a logarithmic x-axis; the lower left plot should have a linear x-axis but a logarithmic y-axis; and the lower right plot should have both axes logarithmic.
Make sure to change the color for each subplot. Also, add axis labels to all subplots, add a title to the whole figure, and save and show the figure. **10** P.

b) Add two new arguments to `plot_subplots`, `grid` and `function`. Change `plot_subplots` such that passing `True` for `grid` adds a grid to all four subplots and the function that is plotted is `function`.
However, one should still be able to call the function as `plot_subplots(x_min, x_max)` which should behave as `plot_subplots(x_min, x_max, False, np.exp)`. **5** P.

Note: It is enough to change `plot_subplots` such that `np.sin` and `np.cos` can be given for `function`. Regarding the title of the figure, you can use `function.__name__` to get the name of `function`.

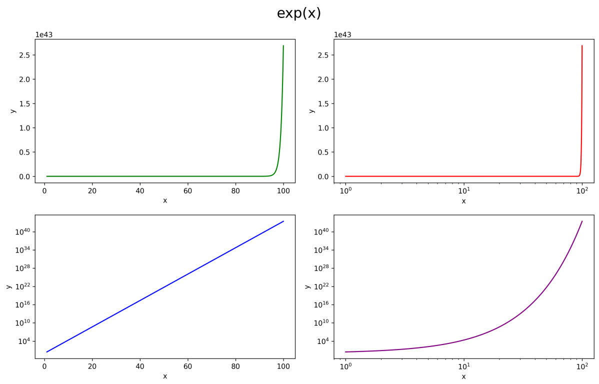

Figure 2: Example plot for `plot_subplots(1, 100)`

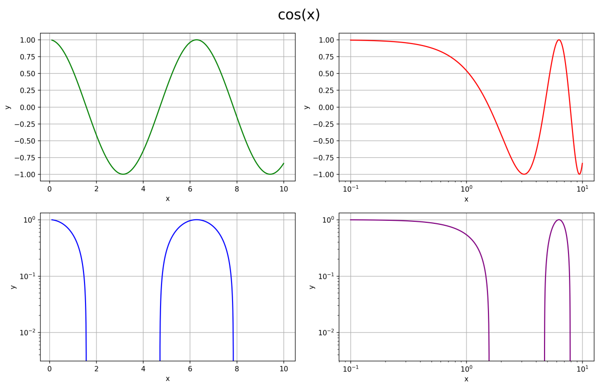

Figure 3: Example plot for `plot_subplots(0.1, 10, True, np.cos)`