MA124 Maths by Computer: Assignment 1

A. Taylor series approximations (12 marks)

One can approximate $\sin(x)$ using a finite number of terms of a Taylor series:

$$\sin(x) \simeq \sum_{n=0}^{N} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

One can also approximate the natural logarithm $\ln(1+x)$ using a finite number of terms of a Taylor series:

$$\log(1+x) \simeq \sum_{n=1}^{N} (-1)^{n+1} \frac{x^n}{n}$$

Note that for a given N, there are $N+1$ terms of the sine series and the highest power of x is x^{2N+1}, while there are N terms for the $\ln(1+x)$ series and the highest power is x^N.

Assignment: Explore how these approximations compare with the true sine and logarithm functions.

Details:

- Write two Python functions, with names of your choice, that compute the approximations to $\sin(x)$ and $\ln(1+x)$. Each function should have two arguments: `x` and `N`, where `x` is an np.array and `N` is an integer. The functions should return an np.array with the series approximation evaluated at `x`.
- Write Python code to plot $\sin(x)$ for $x \in [-3\pi, 3\pi]$ and **on the same graph** plot the series approximations for all values of N from 0 to some maximum value between 5 and 8. You can decide what you think looks nice.
- Write Python code to plot $\log(1+x)$ for $x \in [-0.9, 2]$ and **on the same graph** plot the series approximations for all values of N from 1 to some maximum value between 5 and 8. You can decide what you think looks nice and you do not need to use the same choice as for the sine function.

For both plots you will want to play with line weights, styles and/or colours to distinguish the true functions and approximate curves. You will also need to adjust the vertical plot limits to have informative plots. You may plot a grid or not as you think best. You are **not required to include a legend**. You can assume that the reader can understand the ordering of the curves with N from their behaviour.

Hints

- The Wikipedia page on Taylor series contains plots that are what you are aiming for (although you are encouraged to produce plots in your own style). It is highly recommended that you skim that page for relevant material and in particular look at the section on approximations.
- The NumPy and Matplotlib notebooks contain a worked example for the Taylor series approximation to of $\exp(x)$. You should first fully understand this example and then use it as a starting point.

Challenge section (worth 4 of the 12 marks)

Attempt this only after you have completed all other tasks.

Write Python code to make two plots showing the error in the Taylor series approximation to $\sin(x)$. The plots are as follows:

- Plot the absolute error $|\sin(x) - f_N(x)|$ as a function of x for $x \in [-3\pi, 3\pi]$, where $f_N(x)$ is the series approximation. The plot should contain multiple curves corresponding to the values of N you used above.
- Produce a log-log plot of the absolute error $|\sin(x) - f_N(x)|$ as a function of x for $x \in [0.2, 2\pi]$ for $N = 0, 1, 2, 3$.
- Briefly interpret the two plots, especially the log-log plot.

B. Parametric roller coaster (10 marks)

You will design and plot a roller coaster. For the purposes of this question, a roller coaster is a closed, embedded (non-self-intersecting) curve in $\mathbb{R}^3$, that has one "fun feature". These roller coasters do not have to make mechanical sense. For example, they can have sharp corners. However, they must be closed so that the passengers can get off where they got on. They cannot self-intersect for passenger safety.

Assignment: Design and plot representative views of a roller coaster.

Details:

- You are going to design your roller coaster in sections using parameterised curves. The design requirements are that the roller coaster must have 3 sections minimum and 6 maximum. There can be at most one straight section. One section must be a "fun feature", meaning that it looks fun, so at a minimum this section does not lie in a plane.
- Your submission should contain a mathematical description of the sections of your roller coaster in the form:

$$x_1(t) = \cdots, y_1(t) = \cdots, z_1(t) = \cdots, \quad t \in [\cdots]$$
$$x_2(t) = \cdots, \quad \cdots, \quad \cdots,$$
$$\vdots$$

followed by plots of the full roller coaster from various viewpoints.

- Your Python code must print the coordinates of the first and last point of each section of your roller coaster. This way it will be clear to the marker that the last point of section n is the same as the first point of section $n + 1$. Do not print all the points on your parameterised curves, only the first and last.

Marks will be awarded for creativity, both in design and presentation.

Hints:

- First review and understand the plotting of the helix in the Matplotlib notebook. This is a useful starting point even if you choose not to use the helix in your roller coaster.

- We suggest getting a basic roller coaster with just 3 sections first before trying anything fancy.

- Python will be your friend in designing your roller coaster. Work one section at a time. Since you are required to print the first and last points of each segment, it will be obvious from the output when sections to line up end to end. Continue to work section by section and make sure the end point of the last section matches the first point of the first section.

- After you have perfected your roller coaster in Python, it should be straightforward to state mathematically the parameterisation of each section.

- Finally, you want to plot it the whole roller coaster. (Do not produce numerous plots of individual sections.) Use subplots to show more than one view. Consider this sketch of a chair and Google "orthographic views". You may explore making 3D plots in Python. However, for consistency in the submissions, you should only submit 2D plots.

Note, because computers work with only a finite number of digits, even when the points match, you may obverve tiny differences in printed values, e.g., in my model solution the point $(\pi, 0, 0)$ is printed first as:

```
6.283185307179586 0.0 0.0
```

and then as

```
6.283185307179586 -3.8473413874435795e-16 0.0
```

The y coordinate of the second is of size 10^{-16} and so approximately zero. Ignore such tiny differences.

Further 3 marks

A further 3 marks will be awarded for each assignment based on overall quality and clarity of the submitted notebook.

Submission

You will submit a single Jupyter notebook.

- The last thing you should do before submitting the notebook is to Restart Kernel and Run All Cells. You should then save the notebook and submit the .ipynb file. **You will lose one mark if you submit a notebook that has not been run.**

- A template is provided to get you started. Please remove all red text from the template before submission.

- A good guide for what the final notebook should be like is: if the notebook is run and all code cells are collapsed, the notebook should be readable as a **short** report, primarily consisting of a short intro to each section followed by figures and descriptions of the figures. Think of the Markdown content as extended figure captions. In later assignments you may need to discuss numerical methods, but not in this first assignment.

- Each code cell should begin with a comment line or lines concisely stating what the cell is for. Functions should have comments or docstrings describing what they do. One assumes the reader understands Python. Add comments to set off blocks of code or to note anything tricky. In most cases Python code explains itself and does not need comments.

Warnings

- Do not wait until the last minute to work on this assignment!

- You should download Template1.ipynb from the Moodle page, edit it in JupyterLab to at least to put in your student number, and save the edited version (maybe also rename the notebook). **Make certain that you can do all these things now, while there is time to resolve any issues during the computer labs. This is especially true if you are running JupyterLab on Azure**. Help is available, but not if you wait until the last minute.

- It is your responsibility to backup your work and submit the assignment by the deadline.
