MA124 Maths by Computer: Assignment 1

A. Perspective view of Euclid's orchard. (4 marks)

Thomae's function on $(0,1)$ is given by

$$f(x) = \begin{cases} \frac{1}{q} & \text{if } x \in \mathbb{Q} \text{ and } x = \frac{p}{q}, \text{ where } p \in \mathbb{N} \text{ and } q \in \mathbb{N} \text{ are coprime} \\ 0 & \text{if } x \text{ is irrational.} \end{cases}$$

Typical computer plots.svg) of this function show points at nonzero values of the function for $q = 2, 3, \cdots$ up to some maximum cutoff value.

A perspective view of Euclid's orchard is essentially the same thing except that, rather than plotting points, one plots vertical lines from $y = f(x)$ to $y = 0$. (The lines correspond to trees whose variation in height is due to distance from the observer.) You can enhance the look of the orchard by plotting points (the Thomae function) as well as lines. By varying the marker size (point size) you have give the orchard some added sense of depth.

Assignment: Write Python code to make a perspective plot of Euclid's orchard.

Details:

- Hint: choose y first, then work out x and plot immediately with `plt.plot` and not `plt.scatter`.
- First work out how to plot Thomae's function. After that runs correctly, modify it to add the lines of Euclid's orchard. After than runs correctly, vary the point marker sizes.
- You are free to choose colours of the lines and points, and choose the marker style.
- You are free to choose the maximum value of q as long as it is less than 40.
- All Python code for this part should be in a single code cell and should produce a single, attractive plot.

B. Taylor series approximations to sin(x) and cos(x). (6 marks)

One can approximate $\sin(x)$ using the first $N+1$ terms of a Taylor series:

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

There is a similar approximation for $\cos(x)$ using $N+1$ terms of a Taylor series.

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

Details:

- Obtain the first $N+1$ terms in the Taylor series for $\cos(x)$. (This is best done by differentiating the terms in the sine series.)

- Write two Python functions, with names of your choice, that compute the approximations to $\sin(x)$ and $\cos(x)$. 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 the values in array `x`. Both functions should be in a single Python cell containing no other code other than comments, but there should be comments.

- Write Python code to make two plots, one for $\sin(x)$ and a separate one for $\cos(x)$. For each, plot the true trigonometric function for $x \in [-2\pi, 2\pi]$ and on the same graph plot the series approximations for all values of N from 0 to some maximum value between 5 and 9. You can decide what you think looks nice. You will want to play with line weights and line styles to distinguish the true and approximate curves. You will also need to adjust the plot limits. You do not need to include legends. You will want to comment on this in your description of the figure, but you can assume that the reader can understand the ordering of the curves with N from their behaviour.

- Write Python code to make two error plots, one for $\sin(x)$ and a separate one for $\cos(x)$. For each, plot the absolute difference $|f(x) - f_N(x)|$ as a function of x, where $f(x)$ is one of $\sin(x)$ or $\cos(x)$, and $f_N(x)$ is the corresponding series approximation. Each plot should contain multiple curves corresponding to the same values of N in the previous item. You may want to adjust the plot limits. You do not need to include legends.

Further details:

- You should not attempt to plot curves for different N's with different line styles. This will lead to undesirable, complicated code.

- For simplicity of discussion, you were told that the argument `x` is an np.array. However, your function should work equally well if it is called with a float `x`.

C. Parametric roller coaster. (10 marks)

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 7 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.

- Python will be very helpful in designing your roller coaster.

    - We suggest starting with the parameterisation of a helix, even if you do not want to use the helix in your roller coaster. Figure out how you want to plot it. Use subplots to show more than one view. Consider this sketch of a chair and Google "orthographic views".
    - Once you are able to plot and view the helix, start generating and plotting different parameterised curves for sections of the roller coaster. This is easy in Python once you have one section. Work at getting sections to line up end to end, so that you have a closed roller coaster.
    - Once you have perfected your roller coaster in Python, it should be straightforward to state mathematically the parameterisation of each section.
- Your submission should contain a mathematical description of the sections of your roller coaster in the form:

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

  followed by plots from various viewpoints together with a brief description of the roller coaster.

Marks will be awarded for creativity, both in design and presentation, but we suggest getting a basic roller coaster with just 3 sections first before trying anything fancy.

Further 5 marks

A further 5 marks will be awarded for each assignment based on overall quality and clarity of the submission; the level of understanding demonstrated; originality, creativity and engagement.

Submission

You will submit a single Jupyter notebook. A template is provided to get you started. Your submission should be such that:

- If the notebook is run and all code cells are collapsed, the notebook should be readable as a **short** report, primarily consisting of 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.

Warnings

- Do not wait until the last minute to work on this assignment!
- You should immediately download Template1.ipynb from the Moodle page, bring it into JupyterLab, edit it at least to put in your student number, and save the edited version (maybe also rename the notebook). It is especially important that you work through these procedures if you are using the online version of JupyterLab. Make certain that you know how to download a copy of your work back on your own hardware so that you will be able to submit it. **Make certain that you can do all of these things now, while there is time to resolve any issues during the online computer labs.**
- If you are using the online version of JupyterLab, please download your work frequently! Do not work for hours without downloading!
- It is your responsibility to backup your work and submit the assignment by the deadline.