MTH1003 Mathematical Modelling

Project I (individual project): Interpolation and root finding

Submission deadline: 12:00 noon on Tuesday 3rd December 2024 — see below for details on how to submit.

In this project you will write some Python code to do cubic interpolation and root finding. The project will test your understanding of these numerical methods as well as your knowledge of Python functions, loops, arrays, and plotting.

You will work individually on this project. Your attention is drawn to the appropriate guidelines on collaboration and plagiarism, which are available from the *Academic Misconduct web page*.

This assessment is **AI-prohibited**. This is because you will demonstrate that you have achieved the intended learning outcomes only if you complete the assessment without using GenAI tools.

To make everyone's task distinct, the calculations done by each student will depend on their student ID. However, codes that are different by the student ID, but identical or very similar otherwise, will not be considered distinct and will be taken as evidence of academic misconduct, with corresponding disciplinary consequences.

This project contributes 10% of the overall module mark.

The function `mydata`

You are provided with a Python file `functions.py` containing the function `mydata`. It takes one input argument, your student ID, and returns two row vectors of length 6 containing some data for you to work with. For example, if your student ID is `123456789`, then to obtain your data you could use the commands

```
from functions import mydata
studentid = 123456789
x, y = mydata(studentid)
```

Task 1: The function `myID` (5%)

In `functions.py`, write a Python function called `myID` that takes no input arguments and returns your student ID as its output argument. (You can find your student ID number on your student card. It will be a 9 digit number, usually beginning with a 7.) This function will allow the marker to recreate your calculations.

Task 2: Plotting (15%)

Write a Python script called `plotmydata.py`. This script will need produce a single plot, saved as a `png` file. In this task you will plot your data points. In the next two tasks you will add more to your plot.

In your script you should import and call the function `mydata` to obtain your data vectors `x` and `y` and produce a plot of `y` versus `x`. Produce a scatter plot (i.e. plot the individual points, not a line). Include axis labels, a title and a legend. Make sure all the labelling, including axis tick values, are of suitable size. Save your plot as a png file — take a careful look at any file you generate to make sure it does contain a readable and well-labelled plot.

Task 3: Cubic interpolation (35%)

In `functions.py` write a function called `cubicfit` to do cubic interpolation. It should take as input (i) a 4-element row vector of x values, (ii) a 4-element row vector of y values, and (iii) a scalar value $\hat{x}$. It should fit a cubic polynomial through the given data and evaluate that polynomial at $\hat{x}$ to produce a single scalar output value (which you might call $\hat{y}$). You should use Lagrange's interpolation formula. Thus, it should be possible to call your function with, for example,

```
studentid = 123456789          # or studentid = myID()
x, y = mydata(studentid)
xhat = 3.4                     # or any other value
yhat = cubicfit(x[1:5], y[1:5], xhat)
```

You should think of ways to test your function to convince yourself that it is working correctly, as the markers will read and test it.

Return to your `plotmydata.py` script and add some code to import your `cubicfit` function and use it to add a line to your plot that interpolates the 2nd, 3rd, 4th and 5th data points. Make sure that this line is added to the legend. You may find it useful to refer to the solutions for the second tutorial sheet for an example.

Task 4: Root finding (45%)

Consider the cubic curve defined by fitting through the four data points `x[1:5]`, `y[1:5]`, as in the above example. We wish to determine where this cubic passes through $y = 0$ between `x[2]` and `x[3]`. In `functions.py`, write a Python function called `findroot` that uses the **regula falsi method** (a bracketing method described in the lecture notes) to approximate the root.

The function `findroot` should take as input a pair of x values, x_{l0} and x_{r0}, as a first guess for the bracketing interval, and provide as output a single value of x, the approximate root. Your function

should stop and return its result, say x_*, when it finds a value of the cubic that is smaller in magnitude than 10^{-5}, that is $|f(x_*)| < 10^{-5}$. It should output the corresponding value of x_*.

Thus, it should be possible to call your function with, for example,

```
studentid = 123456789          # or studentid = myID()
x, y = mydata(studentid)
xl0 = x[2]
xr0 = x[3]                     # or other pair of values
xstar = findroot(xl0, xr0)
```

Your function `findroot` will need to call `mydata` to obtain the `x` and `y` values for the cubic fit, and also import and call your earlier function `cubicfit` to obtain values of the cubic as needed.

You might want to base your function `findroot` initially on the code given in the lecture notes for the bisection method. Note, however, that the stopping criterion should be based on the value of the cubic function, not the size of the bracketing interval. Once you are satisfied it is working you can convert it to regula falsi by a relatively small change.

Again you should think of ways to test your function to convince yourself that it is working correctly.

Return to your `plotmydata.py` script and add some code to import your `findroot` function and use it to add a single point to your plot to indicate the root you have found. Make sure that this is added to the legend. You may also like to use the `plt.axhline` command to add the line $y = 0$ to your plot.

Marks

Your code will be tested in various ways by the marker to check that it gives correct results. It is important that you submit your code in the form of Python functions and scripts, exactly as instructed, since we will partly automate the testing of your code. Pay careful attention to the input and output arguments of your functions, including their order, as well as the function names.

Marks will be awarded for well structured and commented code, as well as correct answers when tested.

Important note: This is an individual project and while you are welcome to discuss it with your peers and with tutors, **your work and coding must be your own and not copied from each other, nor from other sources, such as the internet**. If you do find web-based resources and code, then of course you can study these and use them to inspire how you write your own scripts and functions, but you must not copy them (or copy and then make changes, or similar). You can however take Python scripts or functions from the lecture notes and related materials on ELE, and use and modify these freely.

Submission

Submit only the following files:

- your Python file `functions.py`, containing the Python functions `mydata`, `myID`, `cubicfit` and `findroot` as described above,
- your Python script `plotmydata.py`, containing code to produce your plot, saved as a `png` file,
- your plot of $y = f(x)$ in png format `yplot.png`.

The files, and the functions within them, should be correctly named. The files should be submitted via ELE. **Note**: ELE will only allow you to submit a single file, so please `zip` your files into a single file before submission as follows:

- **On a Windows system:** Put the three files (and nothing else) in the same folder. Right click on the folder and select `send to ... compressed folder`. The resulting folder will have a similar name to your original folder but with .zip on the end. You can submit the zipped folder to ELE.
- **On a Mac:** Put the three files (and nothing else) in the same folder. From a Finder window, ctrl-click on the folder and select `Compress ...`. The resulting folder will have a similar name to your original folder but with .zip on the end. You can submit the zipped folder to ELE.

The deadline is **12:00 noon on Tuesday 3rd December 2024** (week 11 of term 1).