Project 1

How many correct digits can we find when root-finding with double precision floats?

We will apply the Bisection Method in Python to find the root of $f(x) = 0$ for three different versions of $f(x)$. The three versions are:

$$\begin{aligned}
\text{(A)} \quad f(x) &= 24e^{5x^3} - (625x^{12} - 1000x^9 + 750x^6 - 240x^3 + 31)e^3 \\
\text{(B)} \quad f(x) &= 24e^{5x^3} - (625x^{12} - 1000x^9 + 750x^6 - 240x^3 + 32)e^3 \\
\text{(C)} \quad f(x) &= 24e^{5x^3} - (625x^{12} - 1000x^9 + 750x^6 - 240x^3 + 33)e^3
\end{aligned}$$

Important: Each of the functions has *exactly one* root between 0 and 1. That's the root we are trying to find. For each of (A), (B), and (C), your goal is to find that root correct to at least 8 decimal places, using the Bisection Method. This sounds easy since Python floats compute every addition, multiplication, etc. with approximately 16 digits of accuracy, and we have an algorithm which in theory can compute to arbitrary precision if we wait long enough. However, I believe you will find this impossible for *one* of the versions. This fact will lead us eventually to a discussion of the limitations of finite precision computing.

Write Python code to use the Bisection Method, and for each of the three examples, use the starting interval $[0, 1]$ after you check that $f(0)f(1) < 0$. Run enough Bisection Method steps to get an approximate root with (hopefully) 8 correct places of accuracy.

Next, we want to investigate whether we trust that the computed roots have 8 correct decimal places (after the decimal point). To do this, run the Bisection Method with some other starting intervals $[a, b]$ that contain the root, and compare the first 8 decimal places. Be sure to "randomize" your choices of a and b a bit to make sure you're not repeating essentially the same computation. (That means, for example, that using $[0, 2]$ as the starting interval will not be helpful, because you will end up repeating $[0, 1]$ after one step of bisection!) If the results agree, with several different starting intervals, one can be reasonably sure the digits are right.

A few details: When you're checking with alternative starting intervals $[a, b]$, be sure to first check $f(a)f(b) < 0$. You may want to use numbers a and b that are close to 0 and 1, respectively, to help meet this requirement. Import `numpy` and use the library function `np.exp(x)` for e^x. For example, use `np.exp(5*x**3)` for e^{5x^3} and `np.exp(3)` for e^3. Be sure to view plenty of digits in your output.

Run enough steps of Bisection to get the first eight digits to stop changing. If using the textbook's code, choose a tolerance `TOL` that will guarantee at least 8 correct digits. More steps are fine if you think it will help. For each version (A) - (C), report your root r **rounded to 8 digits** after the decimal point. If you feel that you cannot get 8 correct digits, how many decimal places are you confident are correct?

Advice for all projects this semester: Submit a report on Canvas that presents and explains your results. The report doesn't have to be extremely formal, but be sure to present any solutions asked for, and answer all questions.

Begin your report by stating your conclusions about the three versions above, including your best guess for the root (you can stop at 8 digits) in each version. Explain your reasoning. Identify which version is the "difficult" one. Save the python code used and your output, and include these with your report. You can just cut and paste the code and output from the python session into the end of your report. Include code that you used, or least enough example code so I can see how you arrived at your results. Please upload **one file only, as a pdf** – they work best for feedback on Canvas. For example, if you like to use Word, just save or export the file as a pdf.

Due: Thurs., Jan. 29