CSC 395 Homework 1 Spring 2024

Due: Tuesday, February 13

In this homework, you will explore the behavior of some classic root finding methods: when they work well, when they fail, real world uses, and error calculations. This assignment is split into these four parts, and for each you will be expected to submit an individual file showcasing your exploration of these topics. You will submit these files through gradescope.

Make sure that the code you submit is cleanly written and well commented. Whenever questions are asked in the assignment, you should include your answers as comments in your code that are placed in an appropriate location within your file.

Part 1. **Common Root Finding Methods**

In class, we introduced three root finding methods: the Bisection Method, Newton's Method, and the Secant Method. Your first task is to implement these three methods directly in a python file titled `hw1p1.py`. For this part, please ensure that your functions take as input functions `f` and `df`, which represent the function $f(x)$ and its derivative $f'(x)$ respectively. You will need to take additional inputs, but the details are left to you. You should make sure your functions employ both of the common stopping conditions discussed in class: limiting the number of iterations, and using an error tolerance as a stopping threshold (i.e., stop when the difference between two consecutive guesses is within the specified tolerance).

In this part of the assignment, you will be using these three methods to perform some simple root finding. All of the assigned problem in this part should work as intended (assuming you start with reasonable initial guesses for the methods). Use these simple problems as test cases for your functions to ensure that they are working as intended (because we know that these cases should all work well).

Using all three methods, perform the following tasks in the `main` block of your file (code included within the conditional `if __name__ == __main__:`).

1. Consider the function $f(x) = x^3 - 3x + 1$.
   (a) Find the root of this function in the interval [1,2].
   (b) Find the root of this function in the interval [0,1].

2. A function of the form $f(x) = x^2 - b^2$ has a root at $x = b$. Utilize this fact to:
   (a) Approximate the value of $\sqrt{2}$.
   (b) Approximate the value of $\sqrt{3}$.

3. Note that the zero of $f(x) = g(x) - h(x)$ is the point where the curves $g(x)$ and $h(x)$ intersect. Use this fact to find the value $\bar{x}$ such that $\cos(\bar{x}) = \bar{x}$.

Part 2. **Pathological Cases**

Now that we've had a chance to see these methods in action, we are going to take a moment to explore some situations where they can sometimes fail us in various ways. Not all mechanisms of failure will manifest themselves in the same way, and it is important to explore the different behaviors you might encounter when using these methods in the wild.

For this part, we will focus exclusively on Newton's method, though the Secant Method would likely face many of the same challenges (and the Bisection Method has its own problems as was discussed in class). Below you will be presented with another series of tasks similar to those in Part 1, except now undesirable things might occur: you may get wrong answers, and your code may even through errors! That's ok, the whole point of this section is to examine these failure cases. Though please make sure to comment out any lines of code that actually produce errors before submitting (and leave a comment next to the line explaining what error occurs when you attempt it).

When attempting to explain the behaviors you are seeing, it will help to actually plot the functions in question and try to visualize the behavior of Newton's Method that you are seeing. It may also help to add some print statements to show the series of guesses that Newton's Method is taking (though please make sure to comment out those print statements in the submitted version of your code).

In a new file titled `hw1p2.py`, again define Newton's method as you did in Part 1. Then, in the main block of this new file, attempt the following tasks.

1. Again consider the function $f(x) = x^3 - 3x + 1$, but this time use a starting guess of $x_0 = -0.8$. What happens? Which zero do you find? Is this what you were expecting? Why or why not? Make sure to include your answer as a comment in the code.

2. Consider the function $f(x) = x^3 - 2x + 2$ with an initial guess of $x_0 = 0$. What happens to your code? Why do you think this is?

3. Consider the sigmoidal function

$$f(x) = \frac{e^x}{1 + e^x} - \frac{1}{2}.$$

Try to find this function's zero starting with an initial guess of $x_0 = -3$. Then try $x_0 = -2$. What happens when you try these initial guesses? What do you think is happening?

4. Attempt to find the value $\bar{x}$ such that $\sin(\bar{x}) = \bar{x}$. What happens? Why could this case be different from the cos case from before?

5. Consider functions of the form $f(x) = |x|^\alpha$. Using an initial guess of $x_0 = 1$, attempt to find the zero of this function for each of the following values of α:

   (a) $\alpha = 1/3$
   (b) $\alpha = 2/3$
   (c) $\alpha = 4/3$

   Note that your `df` function may have to be defined piecewise, since the derivative will change sign based on the sign of x.

Part 3. **Real World Problem: Maximum Infection Levels in the SIR Model**

In class, we introduced the SIR model for disease propagation, where S are the susceptible individuals, I are infected/infectious, and R are the removed individuals (who cannot be re-infected). The model is given below, where R_0 is called the Basic Reproduction Number of the disease, and represents the average number of new infections caused by each person who gets infected.

$$\begin{aligned} \frac{\partial S}{\partial t} &= -R_0 SI, \\ \frac{\partial I}{\partial t} &= R_0 SI - I, \\ \frac{\partial R}{\partial t} &= I. \end{aligned}$$

Note that these equations have been *nondimentionalized* so that the S, I, and R components all correspond to fractions of the total population, and t is scaled so that one unit of the time represents an average recovery window for the disease.

It turns out that epidemics described by this model all tend to have the same shape: an initial period of exponential growth of the infected group, following by a curved peak, and a drop to 0 as the infection burns out. In the real world, it can be very useful to try to predict where the peak of the epidemic will occur: both when it will occur as well as the size of the peak. That way policy makers can better allocate resources as the disease progresses.

While there are a few complicated analytic tricks we could use to obtain information about this peak, another common approach is to numerically approximate the value. It turns out that we can do this using Newton's Method!

Our first step is to formally define our problem: we are attempting to find the maximum of the curve $I(t)$. Note that $I(t)$ will be maximized when $I'(t) = 0$, so we just need to find the zero of $I'(t) = R_0 \cdot S(t) \cdot I(t) - I(t)$ (as defined by our model). Since

we are finding zeros of $I'(t)$, we will also need an expression for the second derivative to use Newton's Method, which we can obtain through direct differentiation while recalling that our original model gives us the expressions for $S'(t)$ and $I'(t)$ that we will need. This gives us $I''(t) = R_0^2 SI^2 + (R_0 S - 1)^2 I$.

We are almost ready to use Newton's Method to find that peak of the infections. But, we have one problem: we do not have access to closed form solutions for the function $S(t)$ and $I(t)$ that we need to actually evaluate $I'(t)$ and $I''(t)$ as Newton's Method requires. But, we can use numerical integration to approximate the solution to the differential equations, and provide us with the values we need.

You have been provided with a file `sir.py` which contains several functions for numerically solving these ODEs (using the odeint function from `scipy`), as well as a few examples using these function in the main block. These provided functions are:

- `sirFunc`: this computes the RHS of the ODE, and is used internally by other provided functions. You will not interact with this function.
- `sir`: this function takes in the initial values for S, I, and R (represented as a list of the three elements), a final time T to solve until, and the value of R_0. It will return the approximated values of $S(T)$, $I(T)$, and $R(T)$. This is the function you will primarily use for Newton's Method.
- `sirForPlot`: a variation of the `sir` function which returns values for use in plotting. You will not interact with this function.
- `plotSIR`: this function will plot a given scenario for you to better showcase what is happening with the disease. It is not required that you use this function, but it could be helpful to look at the curves in question as you work on finding the peak infections.

With this provided code, you are now ready to find the peak infections. Create a new file `hw1p3.py` and import the provided code (or you may copy-paste it into your file if desired). In this file, your task is to use Newton's Method to find both the time of the peak infections as well as the size of the peak given the value of R_0 for a disease.

You should implement this method, then test it on a few examples in the main block of your code. You may re-use your code from previous parts of this assignment, or you may write a modified version of the method that is more specific to this problem (which may be easier than trying to get your previous implementations to work with this complicated scenario).

Part 4. **Convergence Tests**

In class, we discussed the behavior of the error of our three root finding methods, and found the following results regarding the value $|\epsilon_n|$, the error of the approximation at iteration n:

- For the Bisection Method, $|\epsilon_{n+1}| = k \cdot |\epsilon_n|$.
- For Newton's Method, $|\epsilon_{n+1}| = k \cdot |\epsilon_n|^2$.
- For the Secant Method, $|\epsilon_{n+1}| = k \cdot |\epsilon_n|^\phi$, where $\phi = \frac{1+\sqrt{5}}{2} \sim 1.618....$

We then made an interesting observation. Consider an error that behaves as follows: $|\epsilon_{n+1}| = k \cdot |\epsilon_n|^\gamma$. If we take the log of this equation, we obtain the expression

$$\log |\epsilon_{n+1}| = \gamma \log |\epsilon_n| + \log k.$$

So if we plot all $(|\epsilon_n|, \log |\epsilon_{n+1}|)$ pairs for each iteration on a log-log scale, they should all lie on a straight line with a slope of γ.

In this part of the assignment, you will examine the convergence of these three methods and generate plots showing their rates of convergence. Create a new file `hw1p4.py` to contain your solutions to the following tasks.

Modify your implementations of the three methods so that they return a full list of all iterations rather than simply the final guess. Use these modified functions to perform the following root finding tasks. For each task and each method, compute the absolute value of the error at each iteration (you know the actual root) and plot consecutive pairs of errors on a log-log scale.

1. Consider $f(x) = (x-1)(x-2)(x-3)$ and find the root at $\bar{x} = 1$ (for Newton's Method, use an initial guess of $x_0 = 0.1$, and use reasonable initial guesses of your choice for the other two methods). This case should be easily handled by all three methods, and you should see the desired convergence results.
2. Consider $f(x) = (x-1)^3$ and again find the root at $\bar{x} = 1$ (using the same starting points as before). This function is a pathological example because its derivative is 0 at the root, which can cause problems for our convergence. You should see that the rate of convergence changes for Newton's Method.

As a final task, you should return to your `hw1p3.py` file from Part 3 and modify it to perform a convergence study. For the problems above, you knew the exact location of the zero, so you were able to directly compute the error of the methods. But now, you only have your approximation for the peak infection time, so you cannot directly evaluate the error. There are two common strategies to deal with this. Pick one of the following strategies and utilize it to general the convergence plot for Newton's Method when finding the peak infections.

- Once Newton's Method has converged, the answer it gives at its final iteration should be a good approximation for the value of the zero. So you can approximate the error of each iteration by comparing it to the guess at the final iteration, i.e., set $|\epsilon_n| = |x_{final} - x_n|$.
- You can compute the error as the difference between consecutive guesses, which should also converge at the same rate, i.e., set $|\epsilon_n| = |x_{n+1} - x_n|$.