Exercise #6 - Solving Non-linear Equations

Computational Physics Lab

Instructor: Prof. Sean Dobbs

March 5, 2020

Due by 11 PM, March 13th

Submission

For each problem, you will write one or more python programs. These programs should follow the Python coding and formatting conventions outlined for the course. Many problems will have additional questions to answer, or will ask for a record of the terminal showing the output when the program is run with various arguements. These answers can be given either in block comments in the prologue of the corresponding assignment, or in clearly labeled text files. The terminal output should contain some information on the username and machine you are running on (this will often be in the command prompt).

You will be evaluated based on the files contained in your remote GitHub repository at the due date. You should and commit all the files you created to your local repository on a regular basis. You should do this by adding any new or modified files using "git add", and then finalizing changes using "git commit". Remember that "git status" will give you information on which files in your repository. have been changed. Remember to add short, but useful comments when performing a commit. When you are finished with the exercise, push the current status of your local repository to the remote repository on GitHub using the command "git push -u". You are encouraged to push your local files to the remote repository periodically before you are done, and certainly well before the deadline, if possible.

1. Wien’s Displacement Constant

Planck's radiation law tells us that the intensity of radiation per unit area and per unit wavelength λ from a black body at temperature T is

$$I(\lambda) = \frac{2\pi h c^2 \lambda^{-5}}{e^{hc/\lambda k_B T} - 1} \,,$$

where h is Planck's constant, c~is the speed of light, and k_B is Boltzmann's constant.

a) Show by differentiating that the wavelength λ at which the emitted radiation is strongest is the solution of the equation

$$5e^{-hc/\lambda k_B T} + \frac{hc}{\lambda k_B T} - 5 = 0.$$

Make the substitution $x = hc/\lambda k_B T$ and hence show that the wavelength of maximum radiation obeys the **Wien displacement law**:

$$\lambda = \frac{b}{T},$$

where the so-called **Wien displacement constant** is $b = hc/k_B x$, and x is the solution to the nonlinear equation

$$5e^{-x} + x - 5 = 0.$$

b) Write a program to solve this equation to an accuracy of $\epsilon = 10^{-6}$ using the binary search method, and hence find a value for the displacement constant.
Use the `scipy` module to obtain values for the parameters: h, c, k_B. See " `pydoc scipy.constants` ".

c) The displacement law is the basis for the method of *optical pyrometry*, a method for measuring the temperatures of objects by observing the color of the thermal radiation they emit. The method is commonly used to estimate the surface temperatures of astronomical bodies, such as the Sun. The wavelength peak in the Sun's emitted radiation falls at $\lambda = 502\,\mathrm{nm}$. From the equations above and your value of the displacement constant, estimate the surface temperature of the Sun.

For full credit in your derivation for part (a), your finished program, and your results for in parts (b) and (c).

2. The Roots of a Polynomial

Consider the sixth-order polynomial

$$P(x) = 924x^6 - 2772x^5 + 3150x^4 - 1680x^3 + 420x^2 - 42x + 1.$$

There is no general formula for the roots of a sixth-order polynomial, but one can find them easily enough using a computer.

a) Write a program which obtains a polynomial function, its derivative, and a plotting range from the user. The program should then plot the given polynomial function. The program should then ask the user to input an initial value, then determine and print out the root of the polynomial, repeating this process for additional root finding until the user decides to quit. The program should accurately solve for the root value to at least ten decimal places using Newton's method. The purpose of the plot is to aid the user in determining an initial input value for finding the appropriate root(s) of the polynomial.

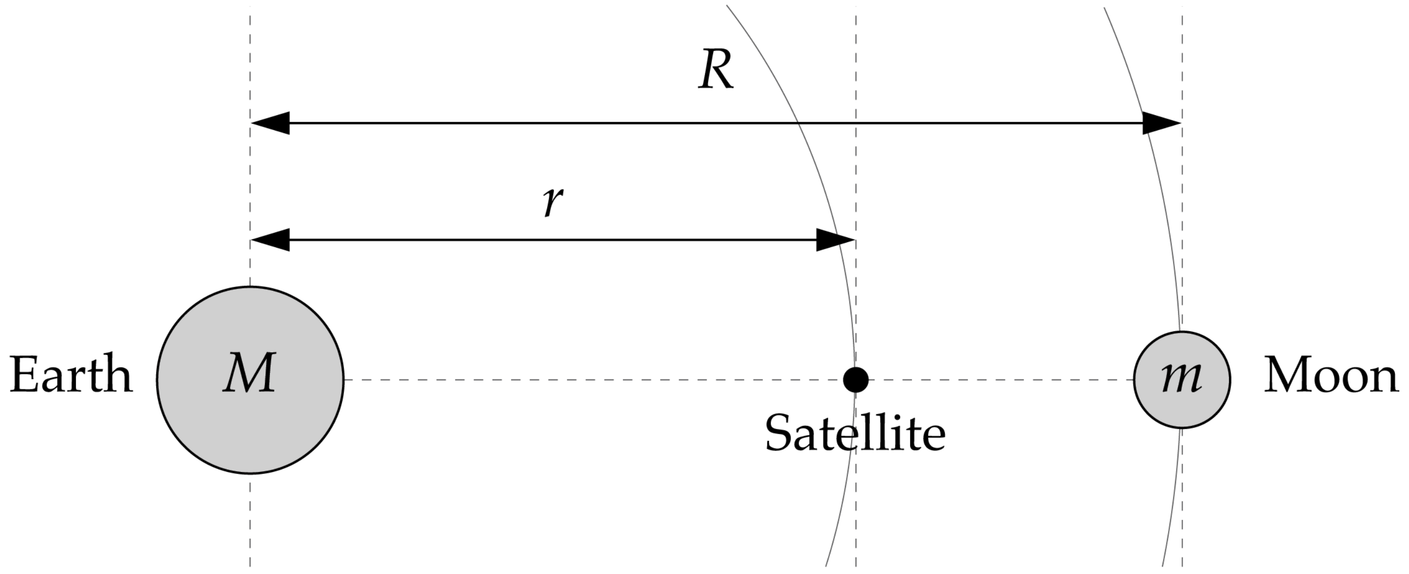
b) Use your program to find the six roots of the polynomial $P(x)$ in the range $x = (0, 1)$.

Note that the polynomial $P(x)$ in this example is the sixth Legendre polynomial mapped onto the interval from zero to one.

For full credit turn in your program, a print out of your graph, and the output from this program.

3. The Lagrange Point

There is a magical point between the Earth and the Moon, called the L1 Lagrange point, at which a satellite will orbit the Earth in perfect synchrony with the Moon, staying always in between the two. This works because the inward pull of the Earth and the outward pull of the Moon combine to create exactly the needed centripetal force that keeps the satellite in its orbit. Here's the setup:



a) Assuming circular orbits, and assuming that the Earth is much more massive than either the Moon or the satellite, show that the distance r from the center of the Earth to the L_1 point satisfies

$$\frac{GM}{r^2} - \frac{Gm}{(R-r)^2} = \omega^2 r,$$

where M and m are the Earth and Moon masses, G is Newton's gravitational constant, and ω is the angular velocity of both the Moon and the satellite. Obtain the value of G from the `scipy` module.

b) The equation above is a fifth-order polynomial equation in r (also called a quintic equation). Such equations cannot be solved exactly in closed form, but it's straightforward to solve them numerically. Write a program that uses either Newton's method or the secant method to solve for the distance r from the Earth to the L_1 point. Compute a solution accurate to at least four significant figures.

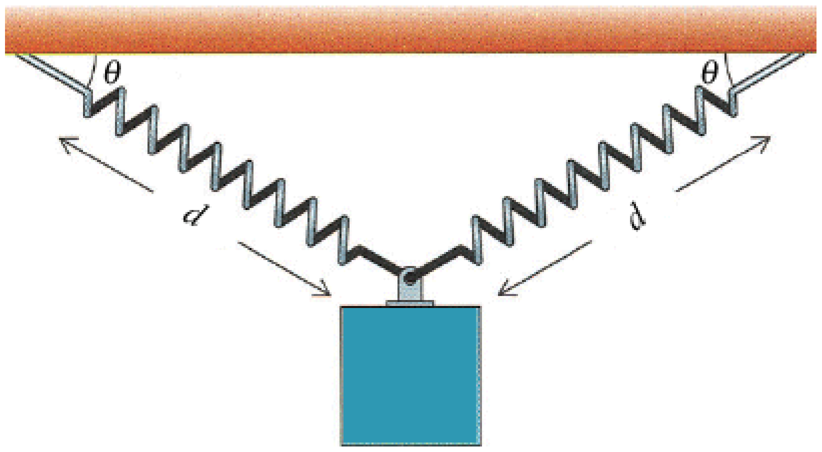
The values of the various parameters are: $G = 6.674 \times 10^{-11}\ \mathrm{m^3 kg^{-1} s^{-2}}$, $M = 5.974 \times 10^{24}$ kg, $m = 7.348 \times 10^{22}$ kg, $R = 3.844 \times 10^{8}$ m.

You will also need to choose a suitable starting value for r, or two starting values if you use the secant method.

For full credit turn in your program, your derivation for part (a), and your results for part(b).

4. Mass on Two Springs

As a simple example of the usefulness of the root finding techniques we will attempt to solve a deceptively simple problem that does not have an analytic solution, namely that of a mass suspended between two springs as shown below.



The basic problem is to find the value of θ at which this system will be in equilibrium. For our present purposes we will use the following values:

- $m = 5$ kg, mass of the object
- $L_0 = 0.3$ m, half the distance between the two supports & spring rest length
- $k = 1000$ N/m, spring constant
- $g = 9.81$ m/s^2, acceleration due to gravity

Given the information above, derive the following equation that can be solved to find the equilibrium angle θ:

$$\tan(\theta) - \sin(\theta) - \frac{mg}{2kL_0} = 0$$

The physical problem is to find the angle of the spring when the above system is in equilibrium. If we assume that we can determine the angle to an accuracy of 1/1000 of a degree, then we would like a numerical technique that provides us with at least that much precision.

The physical problem is to find the angle of the spring when the above system is in equilibrium. If we assume that we can determine the angle to an accuracy of 1/1000 of a degree, then we would like a numerical technique that provides us with at least that much precision

One way to accomplish this is to search for the zero of the function in equation above. We can accomplish this using any one of the methods described in class. Furthermore if we examine the progression of the estimates obtained from our root finding methods and continue to iterate until the change in consecutive estimates is below the desired precision, we can get a result that has a known accuracy.

a) Write a program which utilizes the Bisection Method to solve this problem. Implement the Bisection Method as a function which returns a list containing the root value and the number of iterations to determine the root to a desired accuracy. You should obtain a value near 30 degrees.

b) Once you are satisfied with your program from part (a), implement additional root finding functions for the Newton Raphston Method, False Position Method, and Secant Method. Modify your program to solves the two spring problem using all four methods: Bisectional, False Position, Newton-Raphson, and Secant Methods. Compare the results given by all methods. Do they obtain the same roots? Do they have the same rate of convergence to the answer?

For full credit turn in your program, the derivation of the equation $f(\theta)$, and your results from parts (b).