PHYS 381 – Computational Physics I (Winter 2025)

Assignment #1: Finding Minima of Functions

Due date: February 3, 2025

Group members:

Member #1: Jared Crebo (30085839

Authors' contributions:

This assignment was completed solo. All work presented in this document and the related code was completed by Jared Crebo.

Abstract (0.5 points):

This assignment aims to provide a practical application of minima finding algorithms, which are commonly used in numerical methods and computational sciences. The two main algorithms in use are the Bisection method and the Newton-Raphson method. Both methods will be implemented to solve for the roots of basic second-order parabolic functions. The Newton-Raphson method will also be used to find the minimum value of an exponential decay function. These methods were programmed using the Python programming language in Jupyter Notebook on Visual Studio Code. They were implemented using functions to allow for repetitive testing for robustness across various initial guesses for *x*. Both methods are compared in terms of their efficiency and robustness for different initial guesses. The main takeaway from this assignment is a deeper understanding of these numerical methods and how they can be applied to solve problems.

Introduction (0.5 points):

Numerical methods are essential in the modern age of engineering and science to solve complex mathematical problems without known analytical solutions. Among many methods and algorithms available, the bisection and Newton-Raphson methods are some of the simplest and most useful tools. The bisection method makes use of the Intermediate Value Theorem to gradually narrow down the range of values that could contain the root until it converges on the root within a specified tolerance. This method requires two initial guesses that must straddle the root, otherwise it will find no solution. The Newton-Raphson method only requires one initial guess and makes use of tangent line approximations to find a solution. This method will find a solution, as long as a real solution exists and that the gradient of the function at the initial guess is in the direction of the solution. These methods will be compared in terms of efficiency and robustness.

Some basic examples of the bisection and Newton-Raphson methods will be examined and then applied to a physics problem involving locating the minima of a potential energy function ruling the interaction of two ions, Na^+ and Cl^-. For a simple, one-dimensional case, the potential energy with respect to the distance between the two ions, *x*, can be described as:

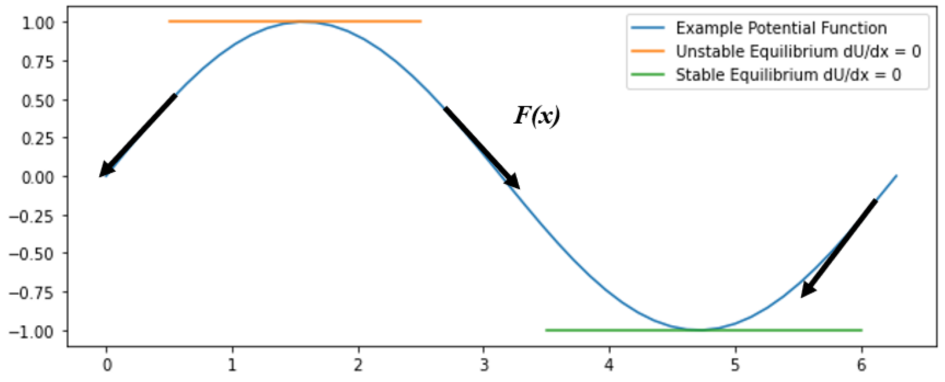
$$U(x) = A\exp\left(-x/p\right) - \frac{e^2}{4\pi\epsilon_0}\frac{1}{x}$$

The first term in the equation is based on the short-range, Pauli repulsion of electron clouds. The second term represents the long-range electrostatic attraction of oppositely charged ions. The force acting on the particles is therefore:

$$F(x) = -\frac{dU(x)}{dx}$$

The negative sign represents the force acting in opposition to the direction of motion, bringing the state to a point of equilibrium at $\frac{dU(x)}{dx} = 0$. On *U(x)*, this is located at the bottom of the potential well, or the minimum of the potential energy function. This state also exists at the maximum of the potential energy function, but this state of equilibrium is considered unstable because any slight deviation from the exact value of *x* that satisfies $\frac{dU(x)}{dx} = 0$ will cause the force to exacerbate its deviation from equilibrium. Adversely, the minimum is a stable state of equilibrium for the same reason.

Figure 1: Stable and unstable equilibrium states on an arbitrary potential energy function



Methods (1 point):

The bisection method uses the Intermediate Value Theorem to narrow the range of possible values of the root until to converges to an approximate solution. This method is very robust in terms of its initial guesses, but relatively slow in efficiency. It will converge as long as the function is continuous on $[x_1,x_3]$ and $f(x_1)*f(x_3) < 0$. Mathematically, the algorithm is as follows:

1) Select a domain $[x_1, x_3]$ such that $f(x_1) < 0$ and $f(x_3) > 0$. On a plot of the function f(x), the function between the initial points $(x_1, f(x_1))$ and $(x_3, f(x_3))$ should cross the x-axis at some point, otherwise no solution will be found.

2) Select a convergence criterion, ζ, to define when the approximate solution has been found. For this assignment, $\zeta = 1e - 4$.

3) Compute the midpoint between $[x_1, x_3]$ such that $x_2 = \frac{x_1+x_3}{2}$ and compute the corresponding $f(x_2)$

4) Check if $f(x_2)$ is within the tolerance of convergence. If $|f(x_2)| > \zeta$, then continue to the next steps. If If $|f(x_2)| \leq \zeta$, **then $(x_2, f(x_2))$ is the solution**.

5) Check if $f(x_2) > 0$, then replace $(x_2, f(x_2)) = (x_3, f(x_3))$. If $f(x_2) < 0$, then replace $(x_2, f(x_2)) = (x_1, f(x_1))$.

6) Repeat steps 3 – 5 until the solution is found.

The Newton-Raphson method employs the Taylor series expansion to approximate the root of a function f(x). It uses the derivative of the function to find the tangent line at x_n, and the tangent line's intersection with the x-axis becomes x_{n+1}. It has very fast convergence if the initial guess is close to the root, however it requires knowledge of the analytical form of f'(x), which may not always be available. Without this analytical component, f'(x) would also need to be approximated which would increase computational time immensely. It is also less robust in terms of its initial guess, where a poor initial guess will cause the solution to diverge. Mathematically, the algorithm is as follows:

1) Select an initial guess x_n and compute $f(x_n)$ and $f'(x_n)$.

2) Select a convergence criterion, ζ, to define when the approximate solution has been found. For this assignment, $\zeta = 1e - 4$.

3) If $|f(x_n)| > \zeta$, then continue to the next step. If $|f(x_n)| < \zeta$, **then (x_n, $f(x_n)$) is the solution.**

4) Compute $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ and $f(x_{n+1})$. For the next iteration, $(x_n, f(x_n)) = (x_{n+1}, f(x_{n+1}))$.

5) Repeat steps 2 – 4 until the solution is found.

Code workflow (1 point):

The code in question 5e works by first defining the functions for $U(x)$, $U'(x) = \frac{dU}{dx}(x)$, and $U''(x) = \frac{d^2U}{dx^2}(x)$. This will make subsequent code more concise and easier to read which functions are being used. The main function, *newton_raphson_energy(x)*, takes in the parameter *x* and first calculates $U'(x)$ and $U''(x)$. This problem aims to find the root of $U'(x)$, rather than $U(x)$ itself. Therefore, the altered Newton-Raphson method uses $x_{n+1} = x_n - \frac{\boldsymbol{U'(x)}}{\boldsymbol{U''(x)}}$ to find x_n where $U'(x_n) = 0$. This gives the minimum or maximum of the function *U(x)*.

For this assignment, I have also selected a relaxation factor, $\alpha = 0.5$, to be applied. This will prevent each iteration from taking large steps and prevent overshooting the solution by applying this factor to $x_{n+1} = x_n - \boldsymbol{\alpha}\frac{f(x_n)}{f'(x_n)}$. With this relaxation factor, the initial guess can be anywhere in the range of x_n = [0.05, 0.95[and this method will produce the correct minimum of *U(x)*. The relaxation factor was defined as $\alpha = 0.5$ through trial and error of testing robustness for a range of initial guesses for x_n. The drawback of applying this factor is that it takes many more iterations, and therefore more computing time to converge to the solution. The variable *nsteps* is defined as 0, which will increase by 1 each iteration to count how many iterations of the method it takes to converge to the solution. An array, *tolerance*, is defined to store *U'(x)* at each iteration to plot against *nsteps* to visualize the convergence. The convergence criterion, ζ, was defined as $\zeta = 1e - 4$.

Within the *while* loop, if the variable *converged* remains *False,* then the loop will continue to iterate. If $U'(x_n) > 0$, then $x_{n+1} = x_n + \alpha\frac{U'(x)}{U''(x)}$ and if $U'(x_n) < 0$, then $x_{n+1} = x_n - \alpha\frac{U'(x_n)}{U''(x_n)}$. This is necessary to drive the converging solution towards the root of *U'(x)*, which is in the positive direction if $U'(x) > 0$ and vice versa. This is a drawback of this code since it is only applicable to this case of *U(x)* and may not apply to a different function. With x_{n+1} updated, $U'(x_{n+1})$ and $U''(x_{n+1})$ are computed and the tolerance is checked. If $|U'(x_{n+1})| < 1e\text{-}4$, then the variable *converged* is assigned to be *True*, the *while* loop ends, and the subsequent solution is outputted to the console along with some plots that will be discussed in the following section.

Results and analysis (1 point):

Figure 2: Plot of bisection method solution to parabolic function

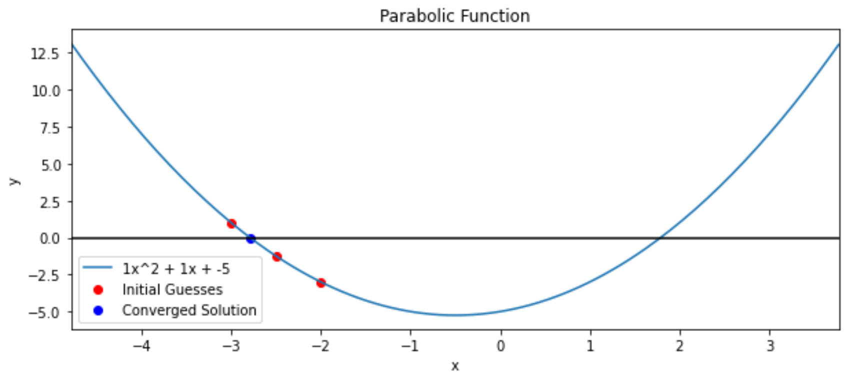


Figure 3: Tolerance vs iteration of bisection method

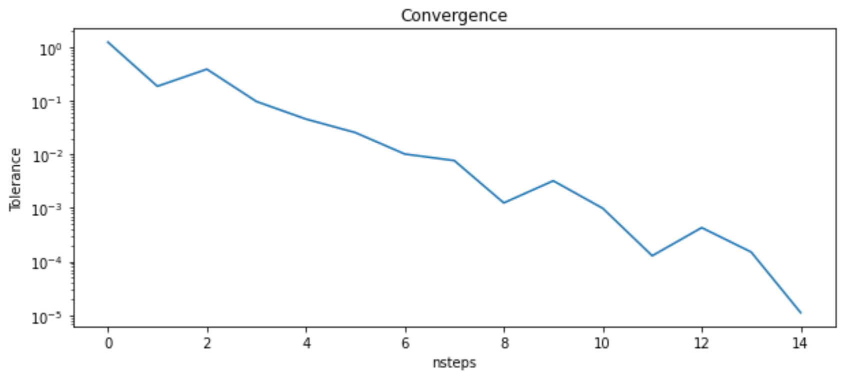


Figure 2 shows the bisection method being used to solve for the root of the function $y = x^2 + x - 5$ with an initial range of $x = [-3, -2]$. This method converged in 15 iterations as shown in Figure 3.

Figure 4: Plot of Newton-Raphson method solution to parabolic function

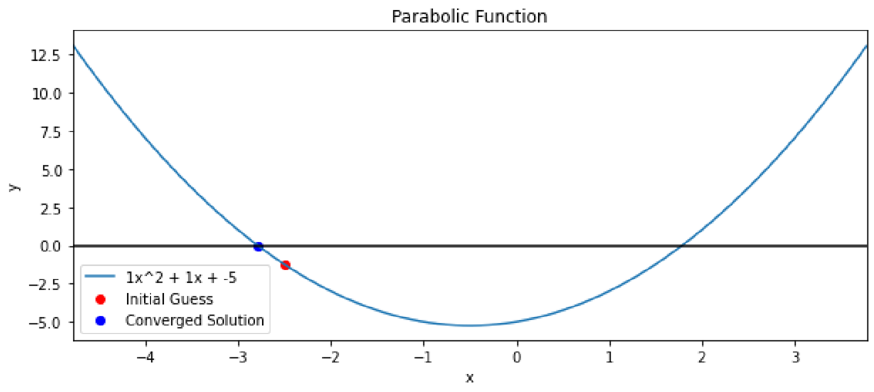


Figure 5: Tolerance vs iteration of Newton-Raphson method

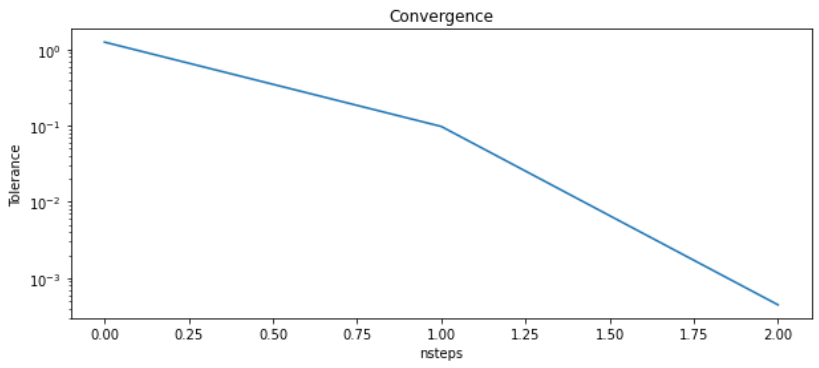


Figure 4 shows the solution to the same function as in Figure 2, $y = x^2 + x - 5$, but this time using the Newton-Raphson method. This was done purposefully to compare efficiency. Both initial guesses are also equivalent, since the average of the initial guesses in the bisection method is roughly -2.5 which was used as the initial guess for the Newton-Raphson method.

Figure 5 shows that the Newton-Raphson method converged within 3 iterations, while the bisection method converged in 15 iterations for the same problem. This demonstrates the superiority of the Newton-Raphson method, as long as the analytical derivative of f(x) is known.

Figure 6: Potential Energy Function, U(x)

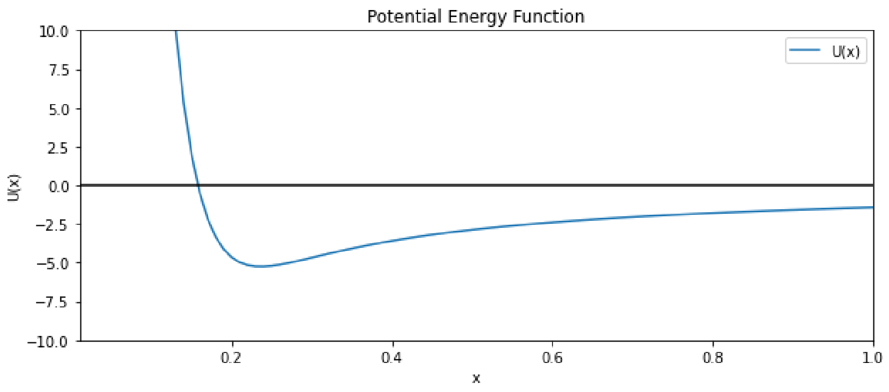
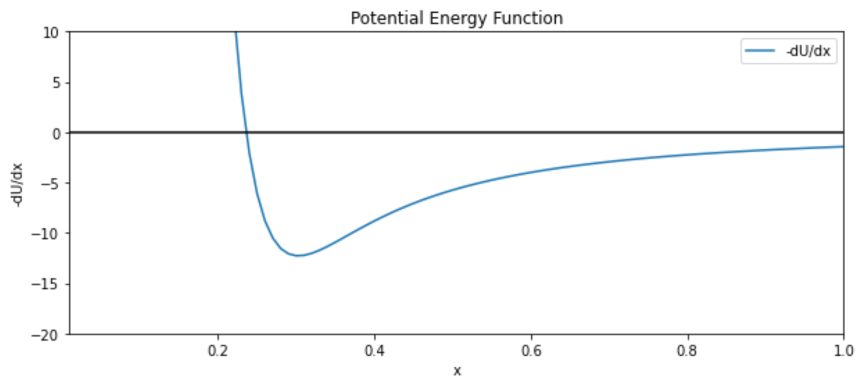


Figure 7: Derivative of Potential Energy Function, -U'(x)



Figures 6 and 7 show the potential energy function U(x) and -U'(x), respectively.

$$U(x) = Aexp\left(\frac{-x}{p}\right) - \frac{e^2}{4\pi\epsilon_0}\frac{1}{x}$$

$$-\frac{dU}{dx}(x) = \frac{A}{p}exp\left(\frac{-x}{p}\right) + \frac{e^2}{4\pi\epsilon_0}\frac{1}{x^2}$$

Physically, $F(x) = -\frac{dU}{dx}$ which equals zero at the minimum of U(x). This means that at the minimum of U(x), there is no force acting on the ions and they are in stable equilibrium. To find where U'(x) = 0, the formula will also require U''(x).

$$\frac{d^2U}{dx^2}(x) = \frac{A}{p^2}\exp\left(\frac{-x}{p}\right) - \frac{2e^2}{4\pi\epsilon_0}\frac{1}{x^3}$$

With the methodology outlined in the **Code workflow**, the modified Newton-Raphson method found the following minimum from an initial guess of 0.2.

Figure 8: Minimum of U(x) using modified Newton-Raphson method

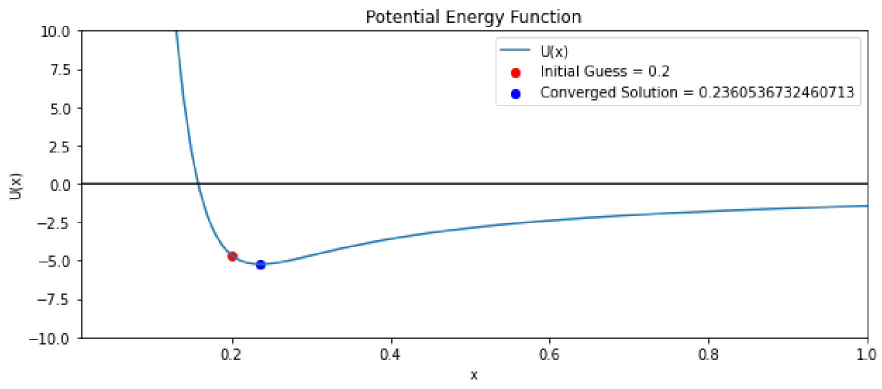
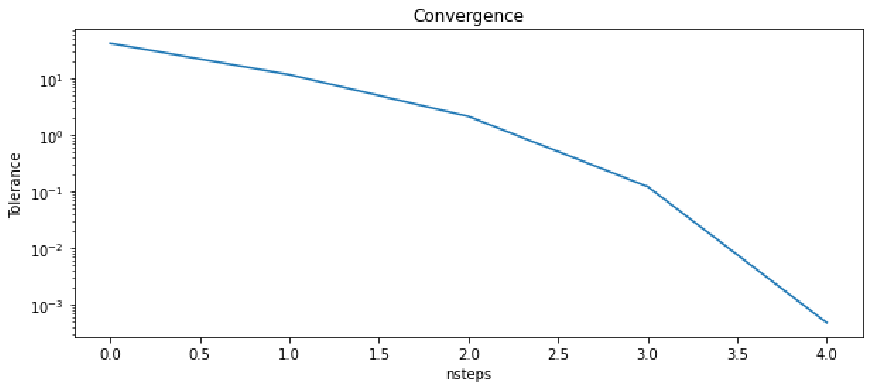
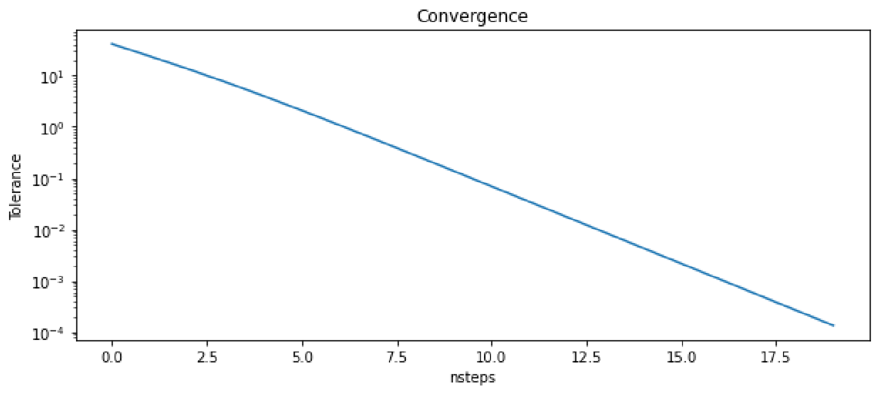


Figure 9: Tolerance vs iteration of modified Newton-Raphson method



With no relaxation factors, this converges in 5 iterations as shown in Figure 9. However, this only works with an initial guess that is very close to the minimum. Testing the same parameters with an initial guess of 0.6 results in the solution diverging, even though it is a relatively close guess to the actual solution. To ensure the robustness of this method, a relaxation factor of 0.5 was added as mentioned in the **Code workflow** and tested for a range of initial guesses between [0.05, 0.95].

Figure 10: Tolerance vs iteration of modified Newton-Raphson method, with relaxation factor



This causes the number of iterations to increase on the order of 2-4x, but it will give the correct answer for a larger range of initial guesses. And given the simplicity of the problem, this decrease in efficiency is not noticeable. These parameters can be adjusted to trade robustness for efficiency, or vice versa, depending on the goals of the user.

Conclusions (0.5 points):

Overall, this assignment investigates the efficacy and efficiency of two numerical methods to be employed in finding the roots and minima of functions. The bisection method and the Newton-Raphson methods are first used to find the roots of parabolic functions, and their efficiencies are compared. The results show that the Newton-Raphson method is more robust and efficient than the bisection method, since it only requires one initial guess and converges in less iterations. But it relies on knowing the analytical derivative of the function. If this is not known, then the computational time to approximate it would increase. The Newton-Raphson method was then employed to find the minimum of an exponential decay function that represents the potential energy between two ions. Physically, the minimum of the potential energy represents a state of stable equilibrium where the force acting on the system is zero. A relaxation factor was applied to the Newton-Raphson method to allow for increased robustness with a decrease in efficiency to find the minimum. This parameter is variable and can be changed depending on the goals of the user.

References:

- PHYS 381 Assignment 1: Finding the Minima of Functions, *PHYS 381: Computational Physics I,* Department of Physics and Astronomy, University of Calgary, Winter 2025

Other (0.5 points):

This page is to be filled by the instructor or TAs ONLY.

Remaining 0.5 points are granted for following the template and overall quality of the report.

Was the assignment submitted before the due date (mark one option)? YES NO