Franklin W. Olin College of Engineering

MTH3199 Applied Math for Engineers – Fall 2025
Assignment 1

Assigned: Tuesday, September 2, 2025.
Lab Report Due: Monday, September 15, 2025 (11:59 PM EST).

Online Resources

- Descriptions of the bisection method: Wikipedia, Berkley, Jeffrey Chasnov
- Descriptions of Newton's method: Wikipedia, UT Austin, Libretexts
- Descriptions of the secant method: Wikipedia, Oscar Veliz, Geeks for geeks
- Order/rate of convergence: Roger D. Peng (definition), (bisection), (Newton's), Libretexts (secant)

Overview

This assignment serves as an introduction to root-finding algorithms. A root, or zero, of a function $f(x)$, is an input x_0 that satisfies the condition $f(x_0) = 0$. As such, a root-finding algorithm is an algorithm (i.e. a sequence of mathematical instructions) for computing the roots of a function. In this module, we will be taking a look at three different numerical methods for root-finding: the **bisection method**, **Newton's method**, and the **secant method**.

The bulk of this assignment will involve implementing and testing these three root-finding methods in order to better understand how each works. When it comes to numerical methods, there is usually no free lunch: there is no perfect algorithm for a given type of problem. Each algorithm comes with its own set of strengths and weaknesses. During and after implementation and testing, you will be asked to think about when it would be preferable to use one of these algorithms instead of another.

Finally, at the end of the assignment, you will be challenged to use a combination of these methods to solve a difficult numerical problem in order to give you some practice thinking about how to apply these new tools to solve a problem that you haven't seen before. One of the deliverables for this part will be an animation of the system being modeled. This assignment will walk you through the necessary steps of generating the animation in MATLAB and then converting it to a video, which is something that you will do several times throughout this course.

Instructions

Day 1 (Tuesday, September 2nd) activity. Please complete before class on Friday, September 5th.

Teaming and Github

Before you begin, you are encouraged to form groups of two or three. Groups will jointly submit a single set of deliverables on Canvas (and will be graded as a single unit). Once you have found teammates (or if you are choosing to work alone), add yourself (and your teammates if you have them) to the same **assignment 01** group on Canvas. You will also need to create a Github account; download and install Github desktop; and create an assignment 01 repository (each team will create and submit a single repository). There are two minor deliverable on Canvas to check to make sure that you have created an account and a code repository. Be sure to submit them by the stated due date (see Canvas). We will go through how to use Github over the next few sessions.

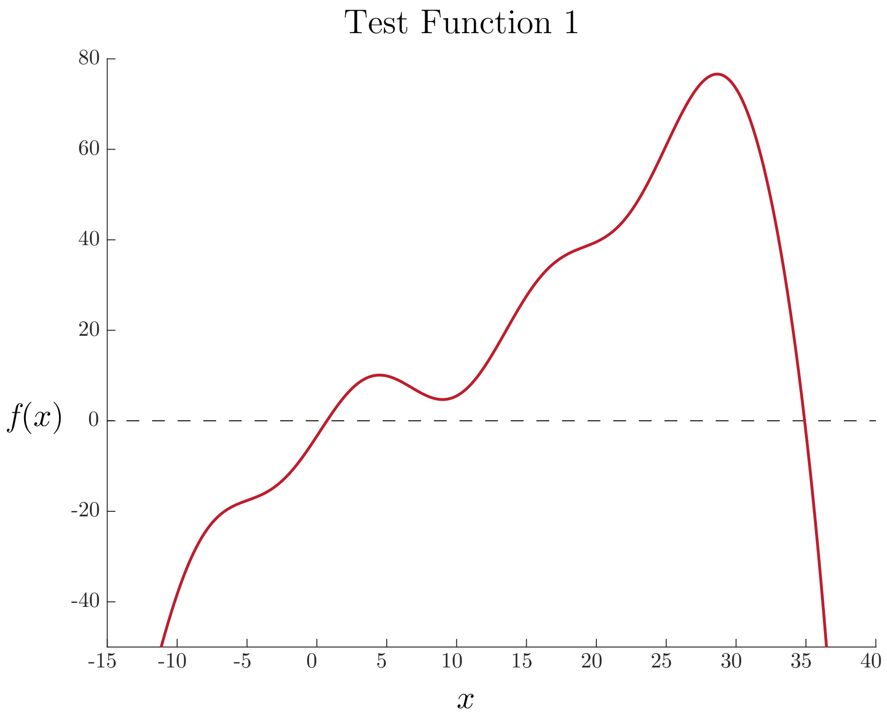
Initial Test Function

We will use the following function as an initial test of our implementations of the three algorithms to make sure that our code is working properly:

$$f(x) = \frac{x^3}{100} - \frac{x^2}{8} + 2x + 6\sin\left(\frac{x}{2}+6\right) - .7 - e^{\frac{x}{6}} \tag{1}$$

$$f'(x) = \frac{3x^2}{100} - \frac{x}{4} + 2 + 3\cos\left(\frac{x}{2}+6\right) - \frac{e^{\frac{x}{6}}}{6} \tag{2}$$

A plot of $f(x)$ is shown on the next page. Finding the roots of this function analytically would be a very difficult challenge (probably impossible). However, we can get a computer to find an extremely precise (though not exact) solution in fractions of a second.



The code defining the test function and its derivative has been provided below for you to copy and paste:.

```
%Definition of the test function and its derivative
test_func01 = @(x)  (x.^3)/100 - (x.^2)/8 + 2*x + 6*sin(x/2+6) -.7 - exp(x/6);
test_derivative01 = @(x)  3*(x.^2)/100 - 2*x/8 + 2 +(6/2)*cos(x/2+6) - exp(x/6)/6;
```

Note that both of these functions are anonymous (they are stored as a variable, instead of as part of a file).

Part 1: Basic Implementation

Let's start by trying to make a small piece of code that works, which we will then generalize into something more useful. In MATLAB, use the bisection method to find the two roots of $f(x)$ shown in the plot above. For reference, the update step for the bisection method is provided below:

$$\text{Bisection Method:}(L_{n+1}, R_{n+1}) = \begin{cases} (L_n, M_n) \text{ if: } f(L_n) > 0 > f(M_n) \text{ or } f(L_n) < 0 < f(M_n) \\ (M_n, R_n) \text{ if: } f(M_n) > 0 > f(R_n) \text{ or } f(M_n) < 0 < f(R_n) \\ \text{where: } M_n = \frac{L_n+R_n}{2} \end{cases} \tag{3}$$

Here, (L_n, R_n) are the endpoints of the interval of the current iteration, and M_n is the midpoint of that interval. Which root the bisection method converges to will depending upon your choice for the initial interval, (L_0, R_0). To validate your results, generate a plot of $f(x)$ overlaid with the horizontal line $y = 0$ and the root points $(x_r, f(x_r))$.

Now, do the same for Newton's Method and the secant method. There is no need to generate new plots, just make sure that the roots that you find are consistent with the ones computed via the bisection method. The update step for these two algorithms have been provided below:

$$\text{Newton's Method:} \quad x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4}$$

$$\text{Secant Method:} \quad x_n = \frac{x_{n-2}f(x_{n-1}) - x_{n-1}f(x_{n-2})}{f(x_{n-1}) - f(x_{n-2})} = x_{n-1} - f(x_{n-1})\left(\frac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})}\right) \tag{5}$$

Part 2: Generalization

At this point, you should have some working code that implements the bisection method, Newton's method, and the secant method to find the root of the test function provided at the start of the exercise. However, these implementations would have a lot more utility if they were not specifically hard-coded for this specific test function! As such, your task is to now generalize the code that you have written to work for any input functions (and initial guess). To do so, refactor the solver portion of your code so that it is in the form of a function that takes a mathematical function and an initial guess as input arguments:

```
function x = bisection_solver(fun,x_left,x_right)
    %your code here
end

%Note that fun(x) should output [f,dfdx], where dfdx is the derivative of f
function x = newton_solver(fun,x0)
    %your code here
end

function x = secant_solver(fun,x0, x1)
    %your code here
end
```

Additionally, edit the rest of your code so that it queries these solver functions to find the root. Essentially, you are tasked with creating a simpler version of MATLAB's fzero function (which uses a combination of these three algorithms plus some more sophisticated methods to do root-finding).

Finally, let's do some work to make the solvers a bit more sophisticated. Please include the following:

- **Early termination conditions:** There is no need for the solver to continue to iterate if the solution is sufficiently correct, or if additional iterations will not change the solution by much. As such, at every step, your solvers should terminate if:

$$|\Delta x| = |x_{n+1} - x_n| < A_{thresh}, \qquad |f(x_n)| < B_{thresh} \tag{6}$$

  where A_{thresh} and B_{thresh} are a pair of very small threshold values (I recommend 10^{-14}).

- **Division safeguards:** Newton's method and the secant method both involve a division operation during the update step. If the denominator is zero, or very close to zero, this will result in a massive or undefined step size, which is really bad. In these cases, I would recommend terminating the loop instead of letting everything break. As such, before computing the next step, add a check to see if the denominator in the update step is zero (which would result in an undefined step size), or if the updated step size $|x_{n+1} - x_n|$ would be huge (also bad), in which case the program should terminate early.

- **Bisection safeguards:** The bisection method does not work if the initial interval does not contain a zero crossing. Add a check to make sure that the initial guess contains a zero crossing:

$$f(x_l) < 0 < f(x_r) \quad \text{or} \quad f(x_l) > 0 > f(x_r) \tag{7}$$

  and terminate the program if this is not the case.

- **Exit flags (not required):** An exit flag is an additional output to a function that indicates whether or not the function completed successfully, or (if it was unsuccessful) how exactly it failed. MATLAB's exit flags are usually integers, whose value corresponds to success or the type of failure mode.

Deliverables and Submission Guidelines

Day 4 (Friday, September 12th) will be dedicated to wrapping up any loose ends and writing your lab reports. Please make sure to submit these deliverables by 11:59 PM on Monday, September 15th.

Your primary deliverable will be in the form of a lab report that documents what you have accomplished and learned. How you write the lab report is up to you, but there are a few items that you should make sure to include, and a few submission guidelines:

Submission Guidelines:

- Remember that each team will be submitting a single assignment. You will need to add yourself to your team's 'assignment01' group on Canvas. Make sure to include the names of you and your teammates at the top of your lab report
- The lab report should be saved as a pdf.
- At the start of the lab report, please approximate how much time each teammate spent working on the assignment (you won't be judged on time spent, but I need it to gauge the assignment difficulty).
- Code will be submitted separately as a link to your team's Github repository for the assignment.
- Please make sure that your repository includes all of the code that you have written over the course of this module. This code should be readable and adequately commented. Code should be submitted as several '.m' files (no livescripts!).

Lab Report Deliverables: This list will be populated as additional components are added to the assignment.

- For each of the three root-finding algorithms that you have implemented, describe one of its strengths and one of its weaknesses.