AMATH 581 Homework #0

Tyler Shakibai

1. Root-finding is the task of finding zeros of a function. The two most common ways to do this are with the Bisection Method and Newton's method.

   Consider the function $f(x) = -x - \cos(x)$. We will try to find the root near $x = -0.74$. In other words, we want to find $\bar{x}$ such that $f(\bar{x}) = 0$. First, use Newton's Method with the initial guess $x_0 = -3$ to find the root near $x = -0.74$ such that $f(\bar{x})$ is within a tolerance of 10^{-6}. Keep track of the iterations required to converge, noting that the first guess of $x_0 = -3$ counts as an iteration. Next use the Bisection Method with the interval endpoints $x_a = -3$ and $x_b = 1$, *i.e.,* the initial interval is $[x_a, x_b]$. Keep track of the calculated midpoint values and the number of midpoints calculated (iterations) until an accuracy of 10^{-6} (on the value of the function) is achieved.

   Deliverables: Save the iterations from Newton's Method to the variable `A2` as a column vector. Save the iterations (calculated midpoints) from the Midpoint Method to the variable `A3` as a row vector. Finally, save the variable `A4` which is a 1×2 vector with the number of iterations for the Newton and Bisection Methods respectively as the two components.

   To practice demonstrating mastery of the "Discussing results from a mathematical perspective" presentation skill, discuss the results you see and what this implies about using either of the two methods for this problem.

   Solutions: In this problem we compared two different numerical methods for finding the root of a simple trigonometric equation, Newton's Method and the Bisection Method. For both methods, we used a tolerance of 10^{-6} on the *value of* $f(\bar{x})$, where $\bar{x}$ is the approximate root. *i.e.,* we should have $|f(\bar{x})| < \epsilon$ for some small $\epsilon > 0$, here $\epsilon = 10^{-6}$. For Newton's Method, we used an initial guess of $x_0 = -3$ and for the Bisection Method we used $[-3, 1]$ as the initial search interval.

   Both methods correctly found the root, within the desired tolerance bounds. Newton's Method converged in 6 iterations whereas the Bisection Method converged in 22 iterations, significantly more than Newton's method. Assuming that each

method uses approximately the same number of operations per iteration, this would imply that Newton's method converges more rapidly than the Bisection Method. That being said, this experiment alone is not enough to make that conclusion for all problems, further analysis is required to draw that conclusion. In future work it would be ideal to calculate the number of floating-point operations used for Bisection and Newton's Method, to determine if Newton's Method is faster on average.

2. Polynomial curve fitting is the act of finding a (usually) low-degree polynomial to fit a number of data points. The most common polynomial curve fit is a line of best fit (or "least-squares line"). Consider the datasets

```
x = [1, 3, 4, 8, 9]
y = [3, 4, 5, 7, 12]
```

Use a built-in tool (`polyfit` in MATLAB or `numpy.polyfit` in python) to find the line of best fit assuming the data represents some function $y = f(x)$.

Deliverables: Save the slope of the best-fit line to the variable `A5`.

To practice demonstrating mastery of the "2D plotting" presentation skill, create a plot of the data and the line of best fit.

Solution:

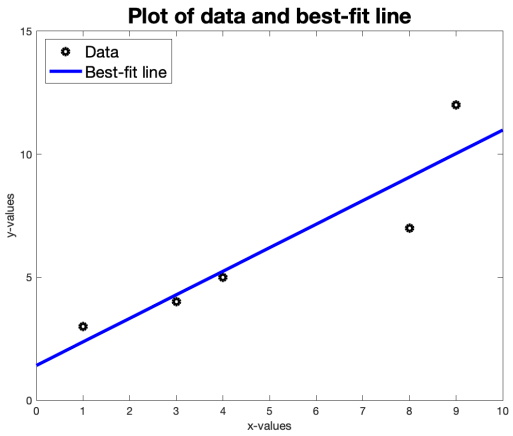

Figure 1: The data (black circles) and best-fit line (blue solid line) created using MATLAB. The positive correlation between the x- and y-data is seen by the slope of the line being positive.

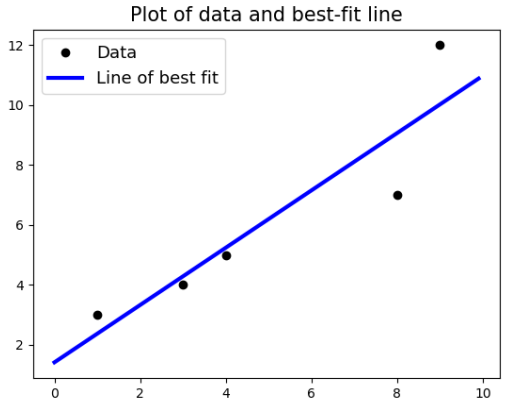

Figure 2: The data (black circles) and best-fit line (blue solid line) created using python. The positive correlation between the x- and y-data is seen by the slope of the line being positive.

Code:

```
x = np.array([1, 3, 4, 8, 9])
y = np.array([3, 4, 5, 7, 12])
z = np.polyfit(x, y, 1)
A5 = z[0]

## 2D Plot

fig, ax = plt.subplots()
ax.plot(x, y, "ko", linewidth=3, label="Data")
xplotting = np.arange(0, 10, 0.1)
ax.plot(xplotting, np.polyval(z, xplotting), "b", linewidth=3, \
label="Line of best fit")
ax.legend(loc="upper left", fontsize=13 )
plt.title("Plot of data and best-fit line", fontsize=15)
```