# Problem 2

Aerospace engineers sometimes compute the trajectories of projectiles such as rockets. A related problem deals with the trajectory of a thrown ball. The trajectory of a ball thrown by a right fielder is defined by the (x, y) coordinates as displayed in Fig. 1. The trajectory can be model

A number and a line

Description automatically generated with medium confidence

A diagram of a curve

Description automatically generated

1. Find the appropriate initial angle θ0, if v0 = 30 m/s, and the distance to the catcher is 90 m. Note that the throw leaves the right fielder’s hand at an elevation of 1.8 m and the catcher receives it at 1 m. Use 𝜀𝑠 = 0.01%

Given:

v0 = 30/ms

y0 = 1.8m

y = 1m

g = 9.81 m/s2

We can use the following code to find our first initial angle for θ0:

A computer screen shot of a code

Description automatically generated

A computer code with black text

Description automatically generated

Using these as our θ0:

[xL, xu] = [0.661967, 0.663540]

For the Bisection method, we get the following:

A screenshot of a computer

Description automatically generated

We can se here that both the root and the error are able to converge.

For the False Position method, we get the following: A screenshot of a computer

Description automatically generated

For the fixed-point iteration method, we have to derive our g(x) first before using the method. The following code is used for the g(x) of the equation:

A screenshot of a computer

Description automatically generated

This equation will then be used for the fixed-point iteration method, along side the same initial value for x0:

A screenshot of a computer

Description automatically generated

Moving on, for the newton method, we first need to find the derivative of the equation. After that, we can plug it into code like so:

A white background with black text

Description automatically generated

Then, we just pass the same initial guess as always:

A screenshot of a computer

Description automatically generated

Using the secant method:

A screenshot of a computer

Description automatically generated

Using Modified Secant:

A screenshot of a computer

Description automatically generated

Using `fzero` from Octave:

A screenshot of a computer

Description automatically generated

Lastly, the Muller method:

A screenshot of a computer

Description automatically generated

1. Repeat part a using different initial guesses (3 different values where applicable).

Using a different initial guess:

A screenshot of a computer

Description automatically generated

Using a second value closer to the first initial guess using in (a):

A screenshot of a computer

Description automatically generated

1. Plot a graph of the approximation percentage error for all the used algorithms in part a.

All plots in (a) and (b) include Root Estimation plot and Error Approximation plot.

1. Which algorithm is the fastest?

The fastest algorithm is the Newton algorithm as it is the fastest to reduce the error to approximate a value, and the one that takes the least amount of iterations.

# Problem 3

A total charge *Q* is uniformly distributed around a ring-shaped conductor with radius *a*. A charge *q* is located at a distance *x* from the center of the ring (Fig. 2). The force exerted on the charge by the ring is given by:

Where . Find the distance where the force is 1.25 N if and are C for a ring with a radius of 0.85 m.

You need to run all codes when possible.

A circle with a line and arrows

Description automatically generated with medium confidence

We begin similarly with Problem 2 by finding possible roots with the following code:

A white background with colorful text

Description automatically generated

Which yields the following possible roots:

A white background with black text

Description automatically generated

[x1, x2] = [0.240240, 0.242242]

[x1, x2] = [1.289289, 1.291921]

Using all numerical methods at once, passing various inputs needed for each piece of code, but specific initial guesses as the ones found, we can see that all solutions converge onto 0.2410~:

A screenshot of a computer

Description automatically generated

# Problem 4

For fluid flow in pipes, friction is described by a dimensionless number, the Fanning friction factor . The Fanning friction factor is dependent on a number of parameters related to the size of the pipe and the fluid, which can all be represented by another dimensionless quantity, the Reynolds number . A formula that predicts given is the von Karman equation:

Typical values for the Reynolds number for turbulent flow are 10,000 to 500,000 and for the Fanning friction factor are 0.001 to 0.01. Develop a function to solve for given a user-supplied value of between 2,500 and 1,000,000. Design the function so that it ensures that the absolute error in the result is E < 0.000005.

For this problem, we had to modify our usual steps to find possible brackets:

A screenshot of a computer code

Description automatically generated

This code, would yield several possible brackets perfect for our methods:

A screenshot of a computer

Description automatically generated

These values are:

Re = 10000

[x\_1, x\_2] = [0.007721, 0.007730]

Re = 64444

[x\_1, x\_2] = [0.004937, 0.004946]

Re = 118889

[x\_1, x\_2] = [0.004333, 0.004342]

Re = 173333

[x\_1, x\_2] = [0.004018, 0.004027]

Re = 227778

[x\_1, x\_2] = [0.003811, 0.003820]

Re = 282222

[x\_1, x\_2] = [0.003658, 0.003667]

Re = 336667

[x\_1, x\_2] = [0.003532, 0.003541]

Re = 391111

[x\_1, x\_2] = [0.003441, 0.003450]

Re = 445556

[x\_1, x\_2] = [0.003360, 0.003369]

Re = 500000

[x\_1, x\_2] = [0.003288, 0.003297]

With these values, we can go over to plug them into our numerical methods functions: