Math 6019 Assignment 1.

Problem 1:

Implement Univariate scan method which would output an interval with the minimum bracketed.

The function is defined in the python file named *UnivariateMethod.py.* This script contains two functions viz 'univariatescan' and 'univariate_plot' to scan an interval that contains a local minimum of an objective function given an initial guess and to plot the objective function along with the bracketed values respectively.

The function 'univariatescan' takes the following input values:

1. 'F': It is the objective function whose local minimum is being calculated. The function is defined as any mathematical equation using 'Anonymous (lambda) function' in python.
2. 'x0': It is the initial guess that user inputs and has a default value of '0.0'.
3. 'tol': It is the tolerance allowed in the error calculation and has a default value of 10^{-8}.
4. 'alp': It is the step size used in this calculation and has a default value of 10^{-2}.
5. 'maxiter': It is the maximum # of iterations defined for stopping criteria and has a default value of 10^{3}.

The 'univariatescan' function returns the values 'a' (Lower bracketed value), 'b' (Upper bracketed value) and 'k' (# of iterations the function took to find the bracketed values).

The function 'univariate_plot' takes the following inputs:

1. 'a': It is the lower bracketed value.
2. 'b': It is the upper bracketed value.
3. 'f': It is the objective function defined by the user.

The 'univariate_plot' returns a figure in a new window showing the plot of the objective function and the scanned values for the local minimum for the given objective function.

Running the script:

1. At the end of the script, I am defining an equation in 'x' using python's anonymous function and storing it in the variable 'Func'.
2. Inputting the value for initial guess and assigning it to the variable 'X0' (**The input value is a float value**).
3. Calling the function 'univariatescan' and passing the values, 'Func' and 'X0' to scan the values containing the local minimum of the given objective function. Assigning the returned values of 'a', 'b', and 'k' to 'A', 'B' and 'K' respectively.
4. Calling the function 'univariate_plot' and passing the values 'A', 'B' and 'Func' to show the figure containing objective function plot and the scanned values containing the local minimum of the objective function.
5. Displaying the scanned values and the number of iterations 'k' to the console.

The python code is appropriately commented defining the process.

Some of the compiled results:

i. $F(x) = x^3-3x^2$ and X0 = 2.5

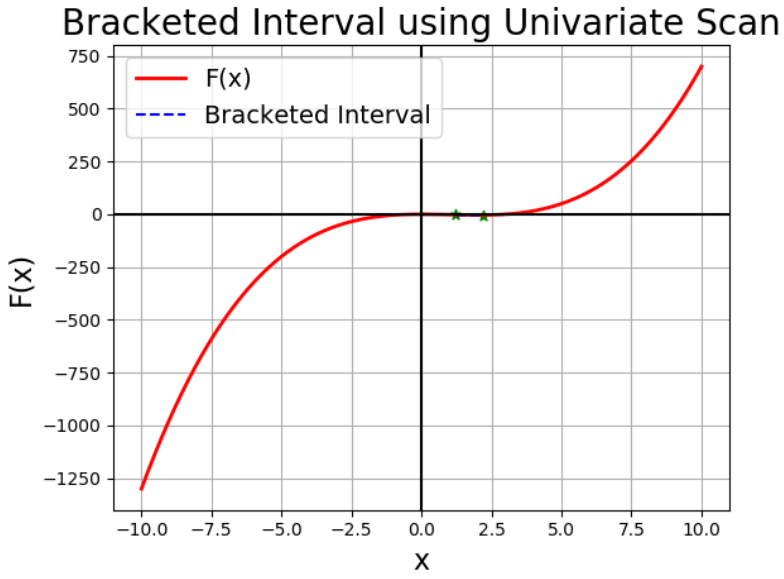

Figure 1: Objective function plot with the bracketed local minimum.

Output values:

'a' = 1.23; 'b' = 2.19; 'k' = 6

Problem 2:

Implement Bisection method which would output a minimum value of the objective function given two guesses that contain the minimum.

The function is defined in the python file named ***BisectionMethod.py.*** This script contains two functions viz 'bisection' and 'bisection_plot' to find the local minimum of the objective function given an interval that contains a local minimum and to plot the objective function along with the minimum value respectively.

The function 'bisection' takes the following input values:

1. 'f': It is the objective function whose local minimum is being calculated. The function is defined as any mathematical equation using 'Anonymous (lambda) function' in python.
2. 'a': It is the lower bracketed value from univariate scan.
3. 'b': It is the upper bracketed value from univariate scan.
4. 'tol': It is the tolerance allowed in the error calculation and has a default value of 10^{-8}.
5. 'maxiter': It is the maximum # of iterations defined for stopping criteria and has a default value of 10^{3}.

The 'bisection' function returns the values 'c' (Local minimum of the objective function), and 'k' (# of iterations the function took to find the local minimum).

The function 'bisection_plot' takes the following inputs:

1. 'c': It is the local minimum of the objective function.
2. 'f': It is the objective function defined by the user.

The 'bisection_plot' returns a figure in a new window showing the plot of the objective function and the local minimum for the given objective function.

Running the script:

1. At the end of the script, I am defining an equation in 'x' using python's anonymous function (same function used in problem 1) and storing it in the variable 'Func'.
2. Inputting the value for 'a' and 'b' and assigning them to the variables, 'A' & 'B' (**The input values are float values obtained from problem 1**).
3. Defining the tolerance value and the maximum iterations for stopping criteria.
4. Calling the function 'bisection' and passing the values, 'Func', 'A' and 'B' to find the local minimum of the given objective function. Assigning the returned values of 'c' and 'k' to 'C' and 'K' respectively.
5. Calling the function 'bisection_plot' and passing the values 'C' and 'Func' to show the figure containing objective function plot and the local minimum of the objective function.
6. Displaying the local minimum value and the number of iterations 'k' to the console.

The python code is appropriately commented defining the process.

Some of the compiled results:

i. $F(x) = x^3-3x^2$; ‘A’ = 1.23; ‘B’ = 2.19;

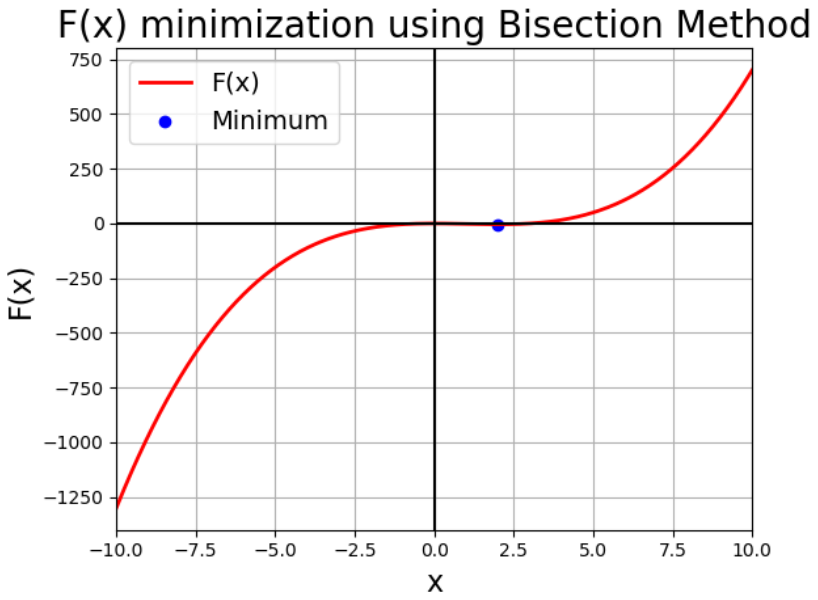

Figure 2: Objective function plot with the local minimum using bisection method.

Output values:

‘c’ = 2.0; ‘F(x=2.0) = -4.0; ‘k’ = 48

Problem 3:

Implement Newton – Raphson method which would output a minimum value of the objective function given an initial guess.

The function is defined in the python file named ***NewtonRaphson.py.*** This script contains two functions viz 'newton_raphson' and 'newtonraphson_plot' to find the local minimum of an objective function given an initial guess and to plot the objective function along with the local minimum value respectively.

The function 'newton_raphson' takes the following input values:

1. 'f': It is the objective function whose local minimum is being calculated. The function is defined as any mathematical equation using 'Anonymous (lambda) function' in python.
2. 'x0': It is the initial guess that user inputs and has a default value of '0.0'.
3. 'tol': It is the tolerance allowed in the error calculation and has a default value of 10^{-8}.
4. 'alp': It is the step size used in this calculation and has a default value of 10^{-2}.
5. 'maxiter': It is the maximum # of iterations defined for stopping criteria and has a default value of 10^{3}.

The 'newton_raphson' function returns the values 'xn' (Local minimum of the objective function), and 'k' (# of iterations the function took to find the local minimum).

The function 'univariate_plot' takes the following inputs:

1. 'xn': It is the local minimum of the objective function.
2. 'f': It is the objective function defined by the user.

The 'newtonraphson_plot' returns a figure in a new window showing the plot of the objective function and the local minimum for the given objective function.

Running the script:

1. At the end of the script, I am defining an equation in 'x' using python's anonymous function and storing it in the variable 'Func'.
2. Inputting the value for initial guess and assigning it to the variable 'X0' (**The input value is a float value**).
3. Calling the function 'newton_raphson' and passing the values, 'Func' and 'X0' to scan the values containing the local minimum of the given objective function. Assigning the returned values of 'xn' and 'k' to 'Xn' and 'K' respectively.
4. Calling the function 'newtonraphson_plot' and passing the values 'Xn' and 'Func' to show the figure containing objective function plot and the local minimum of the objective function.
5. Displaying the local minimum value and the number of iterations 'k' to the console.

The python code is appropriately commented defining the process.

Some of the compiled results:

i. $F(x) = x^3-3x^2$ and X0 = 2.5

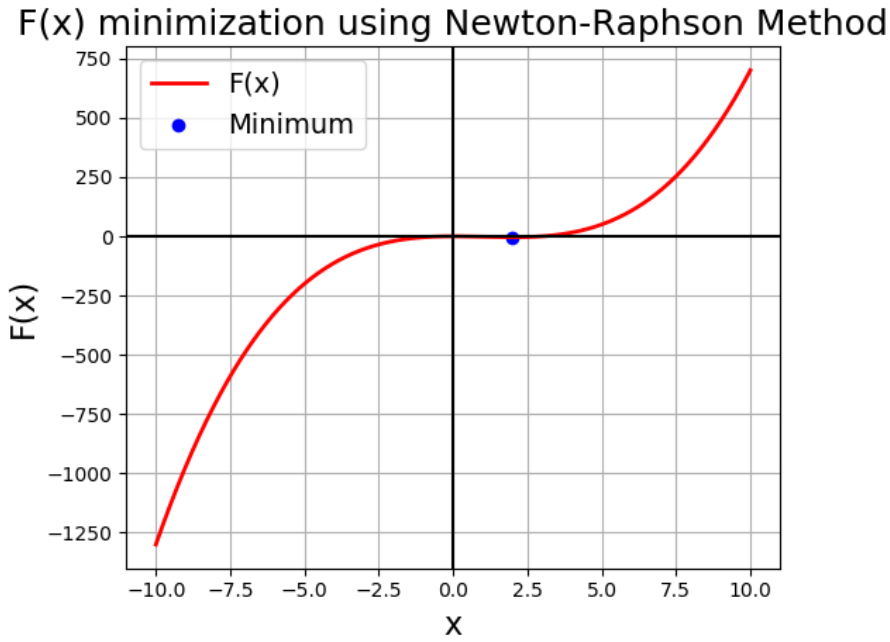

Figure 3: Objective function plot with the local minimum using newton-raphson method.

Output values:

'xn' = 1.995; 'F(x=2.0) = -4.0; 'k' = 4

Problem 4:

Create a function that fits given data to a line through the origin using total least squares.

The function is defined in the python file named ***Problem4.py.*** This script contains two functions viz 'total_least_squares' and 'prob4_plot' to find the local minimum of an objective function given an initial guess and to plot the objective function along with the local minimum value respectively.

The function 'total_least_squares' takes the following input values:

1. 'x0': It is the initial guess that user inputs and has a default value of '0.1'.
2. 'm': It is the slope of the line 'y' given and has a default value of 0.5.
3. 'n': It is the # of points used to create 2xN vector and has a default value of 10^3.
4. 'd_range': It is the range in which N points are defined and has a default range of (0, 5).
5. 'b': It is the intercept of the line 'y' and has a default value of 0.0.
6. 'eps': It is the magnitude of the noise and has a default value of 10^{-2}.

The 'total_least_squares' function returns the values 'xn' (Local minimum of the objective function), 'X' (Array of X values), 'Y' (Array of Y values corresponding to X values)and 'F_a'(Objective function array corresponding to X & Y values).

The function 'prob4_plot' takes the following inputs:

1. 'rnge': It is the range in which N points are defined.
2. 'Minimum': It is the minimum of the objective function.
3. 'X': It is the array containing the X values.
4. 'Y': It is the array containing the Y values corresponding to X values.

The 'newtonraphson_plot' returns a figure in a new window showing the plot of the objective function and the local minimum for the given objective function.

The python code is appropriately commented defining the process.

Input values:

‘X0’ = 0.45; ‘slope’ = 0.5; ‘N’ = 10; ‘D_range’ = (0, 5); ‘y_intercept’ = 0.0; ‘noise’ = 1e-3

Output values:

Minimized ‘a’ = 0.4952; Minimized ‘F(a)’ = 0.0017

Best Line Fit plot:

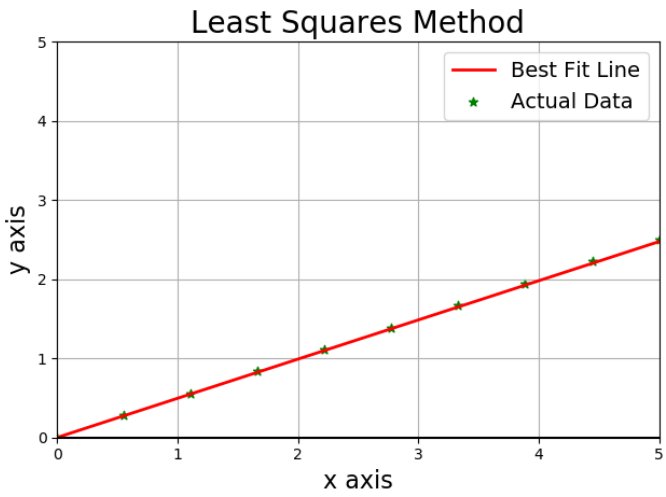

Figure 4: Best Line Fit plot for N=10

Objective Function plot:

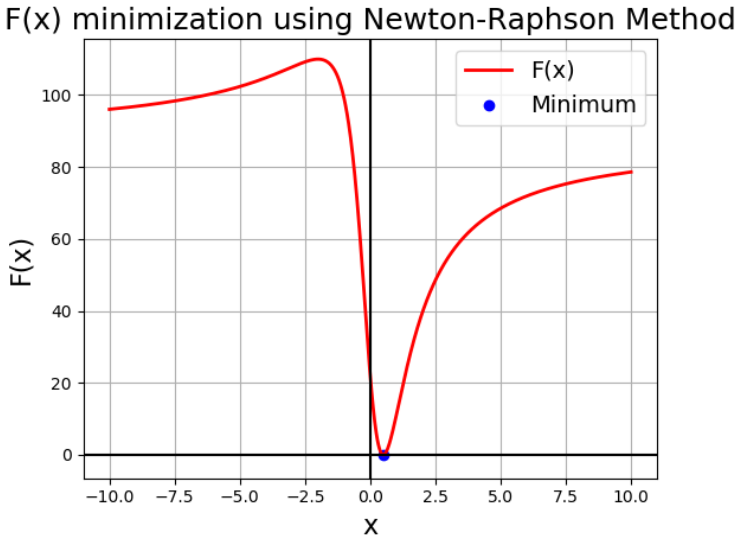

Figure 5: Objective Function plot with the minimum for N=10

Input values:

‘X0’ = 0.45; ‘slope’ = 0.5; ‘N’ = 100; ‘D_range’ = (0, 5); ‘y_intercept’ = 0.0; ‘noise’ = 1e-3

Output values:

Minimized ‘a’ = 0.4948; Minimized ‘F(a)’ = 0.0171

Best Line Fit plot:

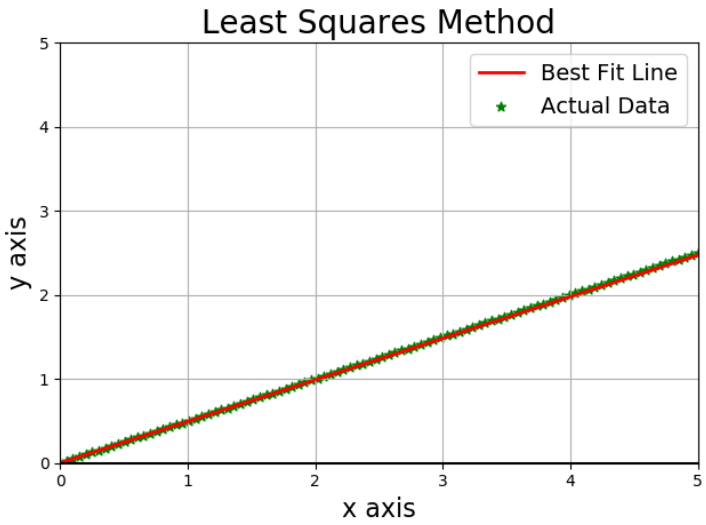

Figure 6: Best Line Fit plot for N=100

Objective Function plot:

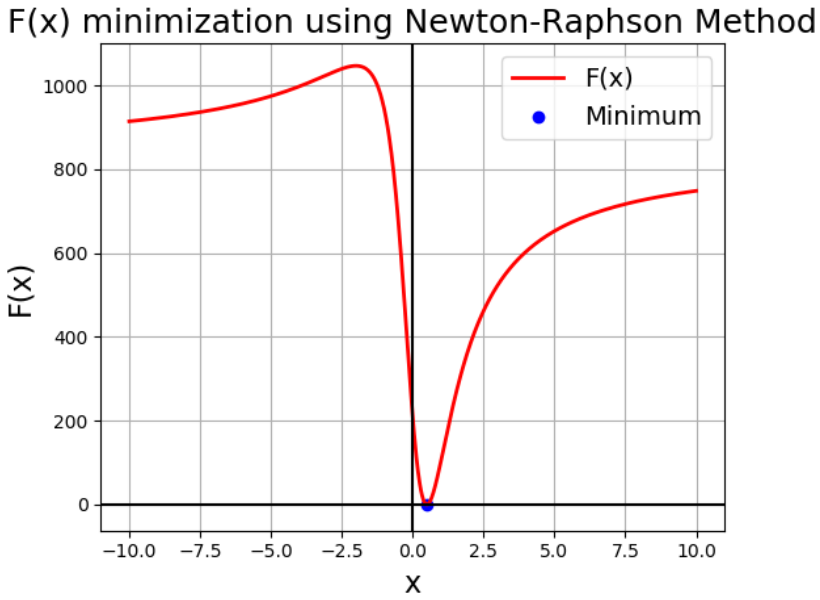

Figure 7: Objective Function plot with the minimum for N=100

Input values:

‘X0’ = 0.45; ‘slope’ = 0.5; ‘N’ = 1000; ‘D_range’ = (0, 5); ‘y_intercept’ = 0.0; ‘noise’ = 1e-3

Output values:

Minimized ‘a’ = 0.4948; Minimized ‘F(a)’ = 0.1683

Best Line Fit plot:

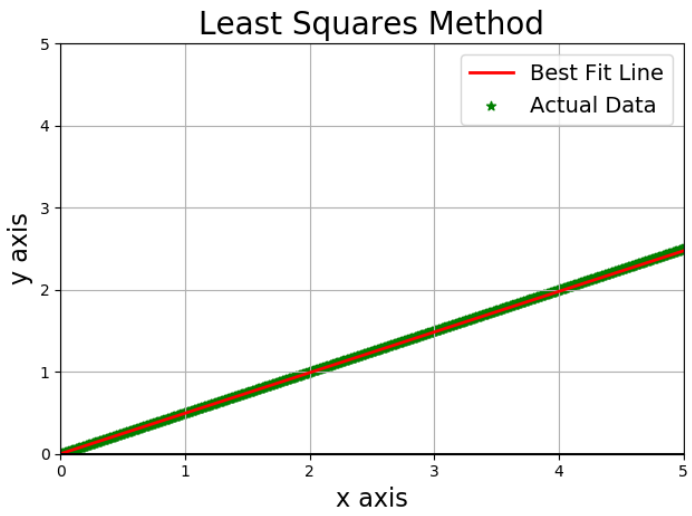

Figure 8: Best Line Fit plot for N=1000

Objective Function plot:

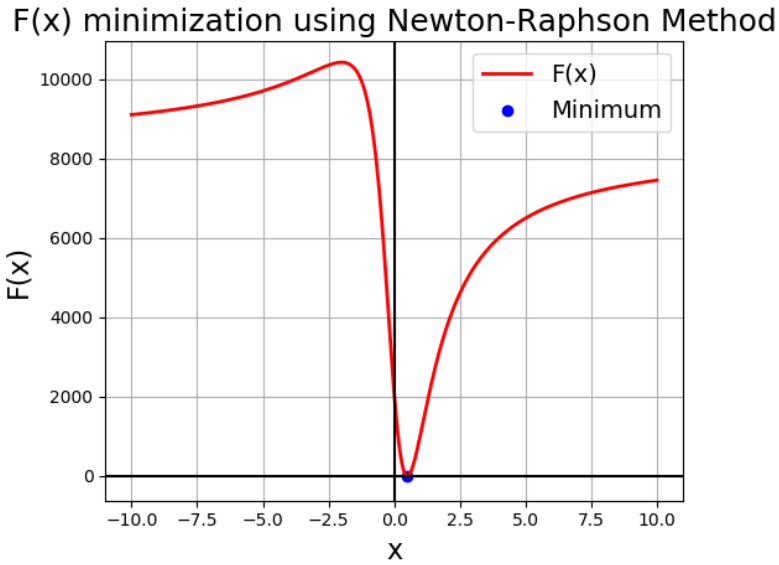

Figure 9: Objective Function plot with the minimum for N=1000

Problem 5:

The collective compiled results for a, b, c for all 3 cases are shown in following figures.

1. P = (5, 3), C = (2, 2), r = 1

Interval containing the minimum. For initial guess value x0 = 0.5

Output a = 0.81, b = 1.77

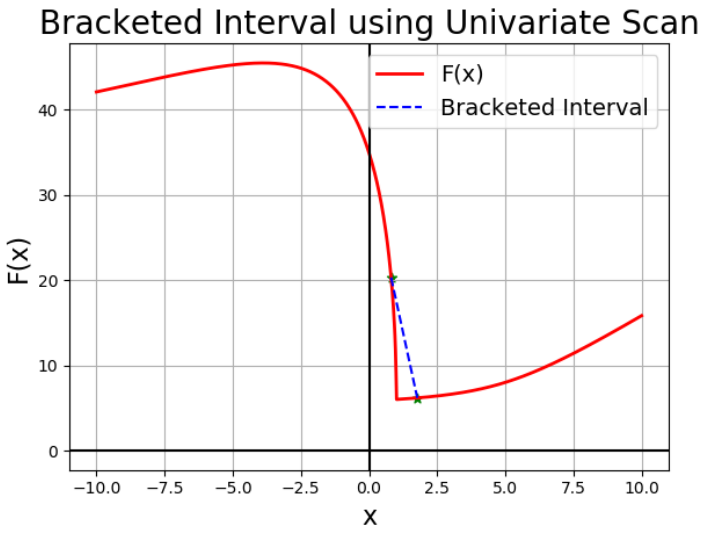

Figure 10: Objective function plot with the bracketed local minimum.

Objective function with minimum point.

Output minimum = 1.0 and F(xmin) = 6.0

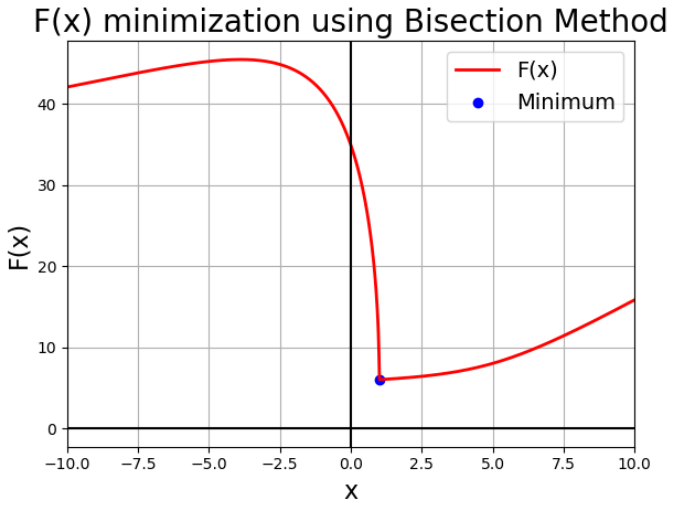

Figure 11: Objective function plot with the local minimum using bisection method.

Optimal Path for Robot. The robot must travel until x = 1.0 along x-axis to avoid the circle while reaching the point P.

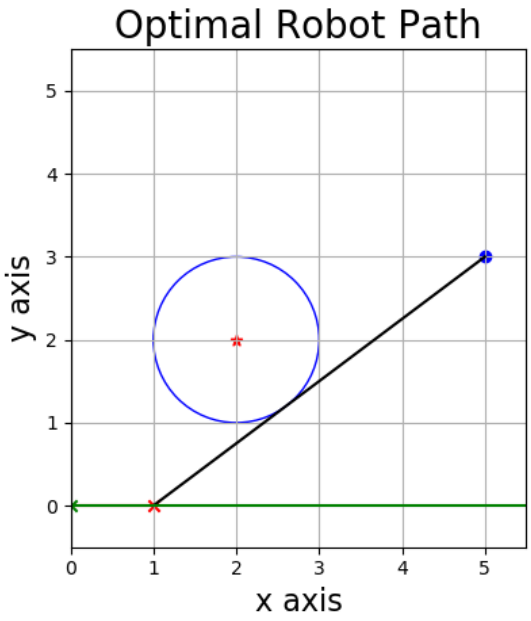

Figure 12: Plot showing the robot path to the given point while avoiding the circle.

2. P = (2, 4), C = (2, 2), r = 1

Interval containing the minimum. For initial guess value x0 = 0.5

Output: a = 0.51, b = 5.0

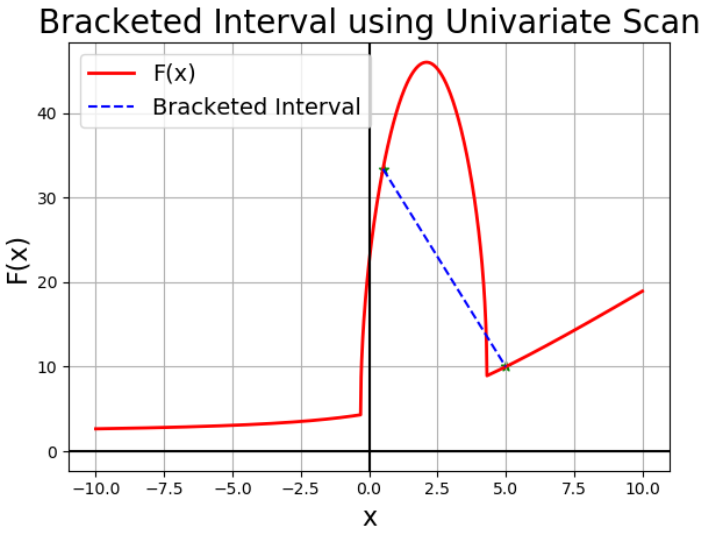

Figure 13: Objective function plot with the bracketed local minimum.

Objective function with minimum point.

Output: minimum = 4.3094 and F(xmin) = 8.9617

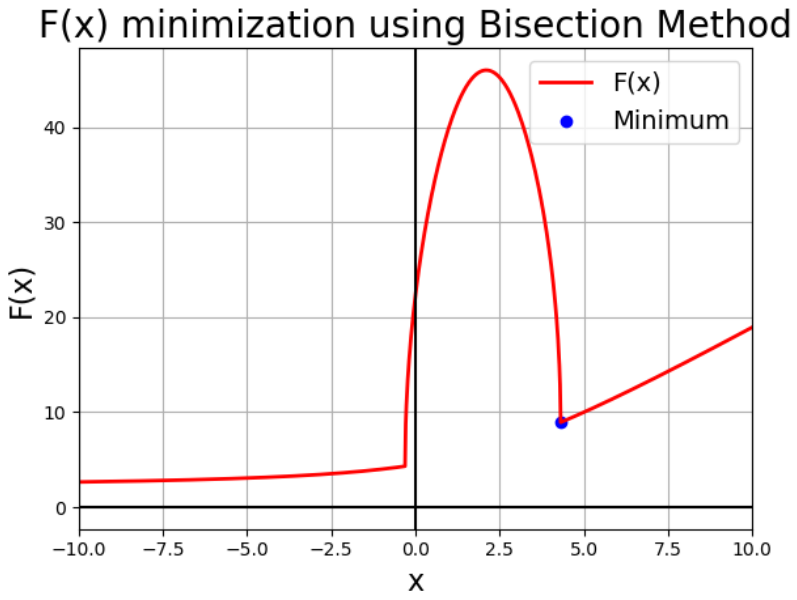

Figure 14: Objective function plot with the local minimum using bisection method.

Optimal Path for Robot: The robot must travel until x = 4.3094 along x-axis to avoid the circle while reaching the point P.

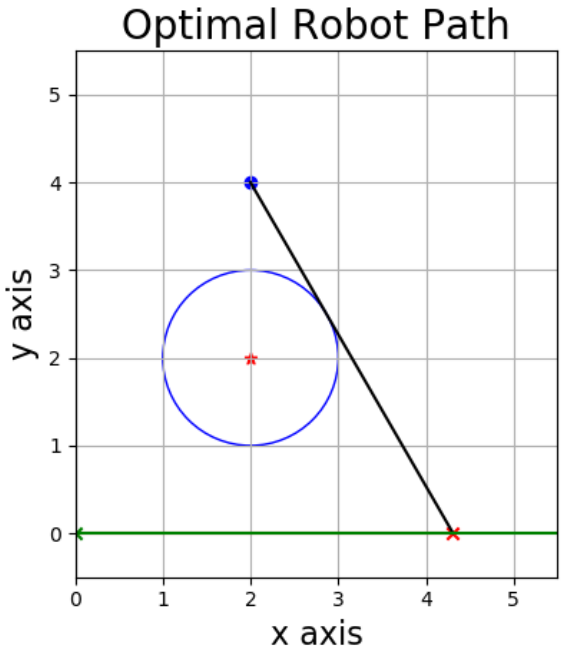

Figure 15: Plot showing the robot path to the given point while avoiding the circle.

3. P = (2, 5), C = (2, 2), r = 1

Interval containing the minimum. For initial guess value x0 = 0.5

Output: a = 0.51, b = 5.0

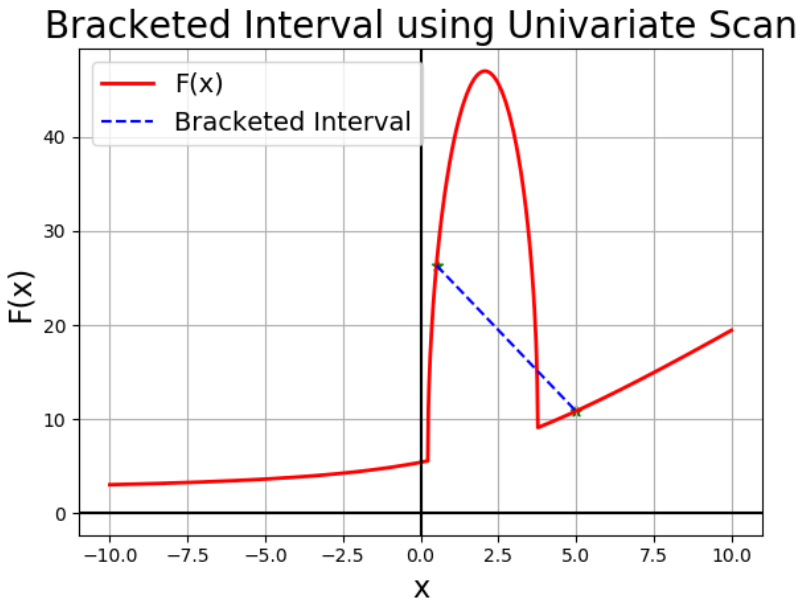

Figure 16: Objective function plot with the bracketed local minimum.

Objective function with minimum point.

Output: minimum = 3.7678 and F(xmin) = 9.0711

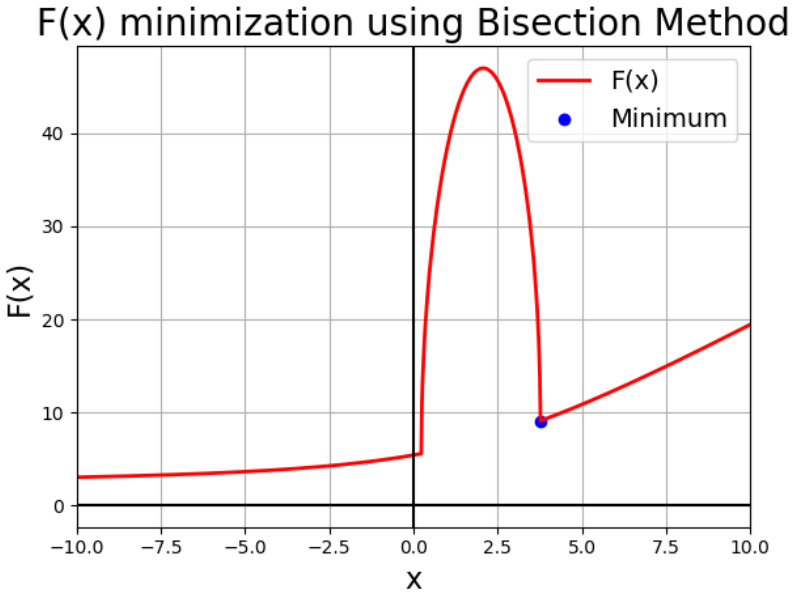

Figure 17: Objective function plot with the local minimum using bisection method.

Optimal Path for Robot: The robot must travel until x = 3.7678 along x-axis to avoid the circle while reaching the point P.

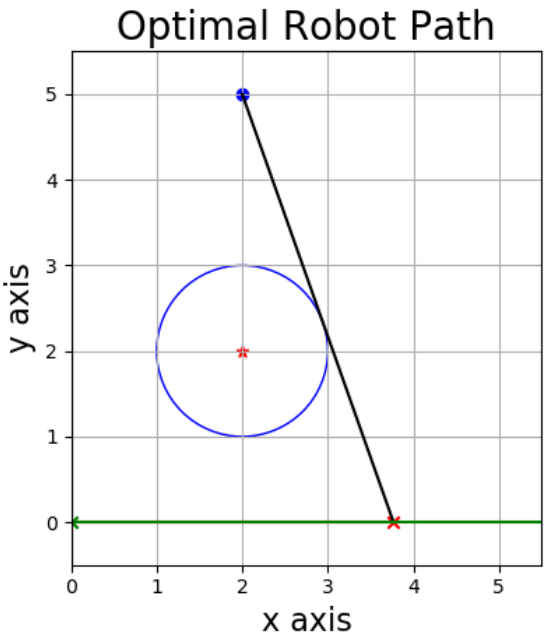

Figure 18: Plot showing the robot path to the given point while avoiding the circle.