Programming Assignment – 1

Name: Saumya Gupta

Roll No. : 210944

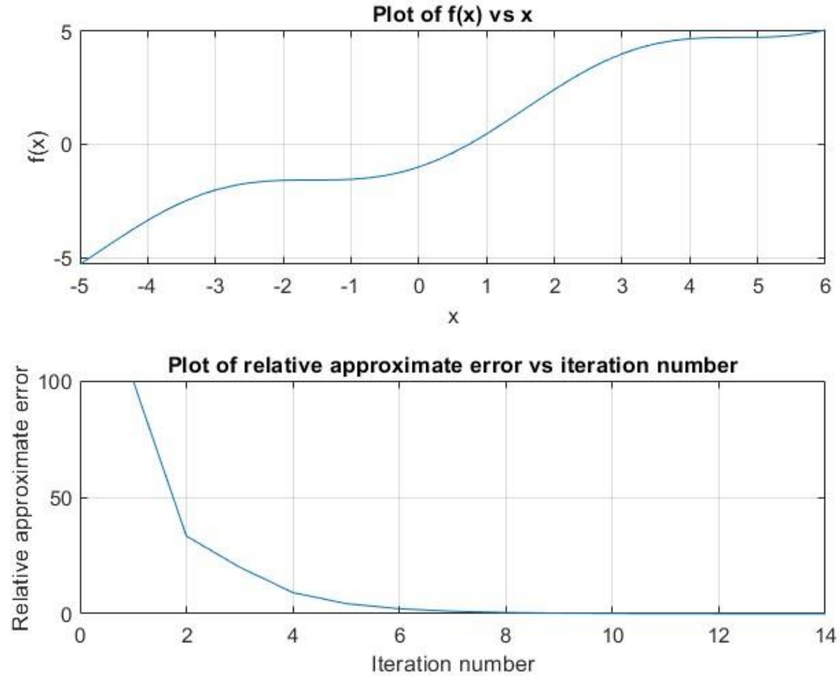
Ans 1

(a) Bisection Method:

(1) f(x) = x – cos(x)

Output: The root is 0.739075

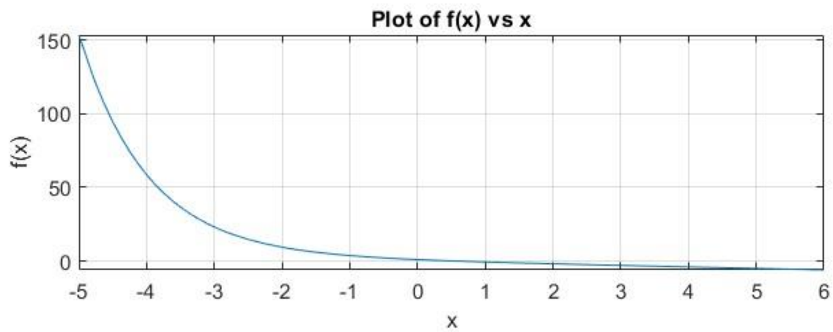
```
Command Window
>> Main
Choose method to find the root. Type:
 1 for Bisection Method
 2 for False-Position Method
 3 for Fixed-Point Method
 4 for Newton-Raphson Method
 5 for Secant Method
 So which method do you like to use: 1
Enter a function in x: x-cos(x)
Enter starting point 1: 0
Enter starting point 2: 1
Enter maximum no. of iterations: 50
Enter maximum error: 0.01
fx The root is 0.739075>> |
```

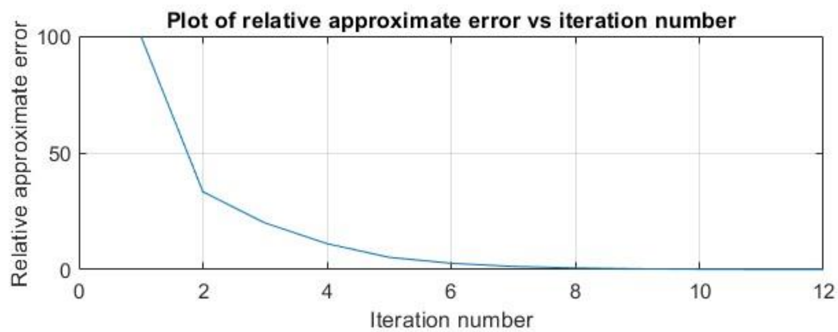


(2) f(x) = exp(-x) - x

Output: The root is 0.567139

```
Command Window
>> Main
Choose method to find the root. Type:
 1 for Bisection Method
 2 for False-Position Method
 3 for Fixed-Point Method
 4 for Newton-Raphson Method
 5 for Secant Method
 So which method do you like to use: 1
Enter a function in x: exp(-x)-x
Enter starting point 1: 0
Enter starting point 2: 1
Enter maximum no. of iterations: 50
Enter maximum error: 0.05
The root is 0.567139>>
```





<u>Convergence of Bisection Method</u>:

The rate of convergence of Bisection Method is linear, i.e., it is very slow.

<u>Stability of Bisection Method</u>:

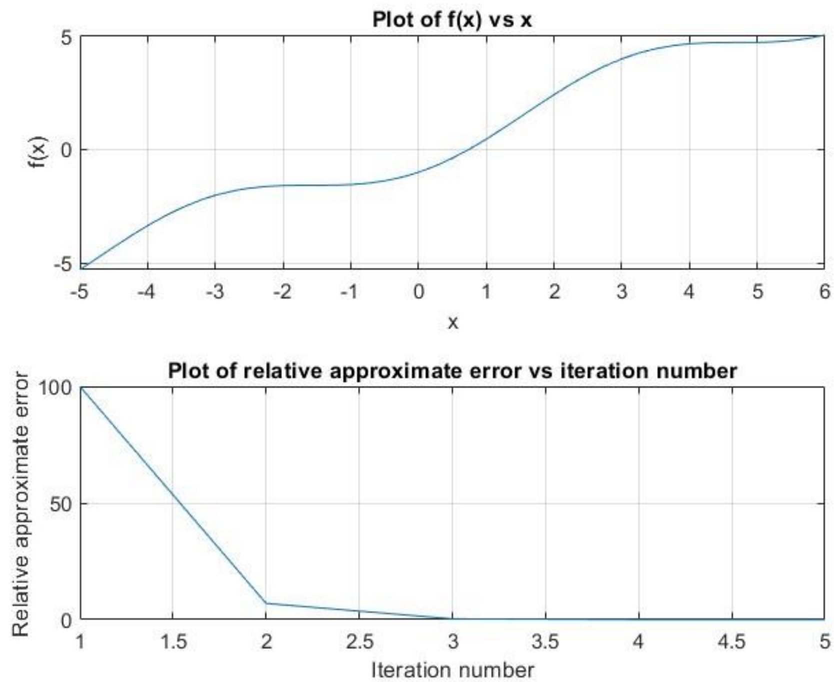
Bisection Method is always stable and the value of approximate percentage error decreases at each iteration as we are going nearer to the root in each iteration.

(b) False-position Method:

(1) f(x) = x – cos(x)

Output: The root is 0.739085

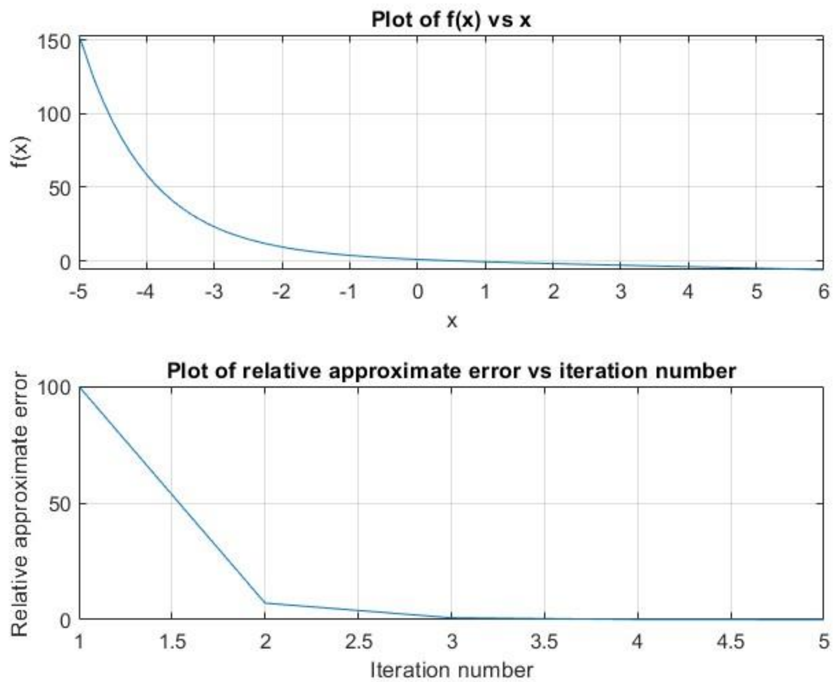
```
Command Window
>> Main
Choose method to find the root. Type:
 1 for Bisection Method
 2 for False-Position Method
 3 for Fixed-Point Method
 4 for Newton-Raphson Method
 5 for Secant Method
 So which method do you like to use: 2
Enter a function in x: x - cos(x)
Enter starting point 1: 0
Enter starting point 2: 1
Enter maximum no. of iterations: 50
Enter maximum error: 0.01
fx The root is 0.739085>>
```



(2) f(x) = exp(-x) - x

Output: The root is 0.567150

```
Command Window
>> Main
Choose method to find the root. Type:
 1 for Bisection Method
 2 for False-Position Method
 3 for Fixed-Point Method
 4 for Newton-Raphson Method
 5 for Secant Method
 So which method do you like to use: 2
Enter a function in x: exp(-x)-x
Enter starting point 1: 0
Enter starting point 2: 1
Enter maximum no. of iterations: 50
Enter maximum error: 0.05
fx The root is 0.567150>>
```



<u>Convergence of False-Position Method</u>:

The rate of convergence of False-Position Method is between linear and quadratic, i.e., it is usually faster than Bisection Method.

<u>Stability of False-Position Method</u>:

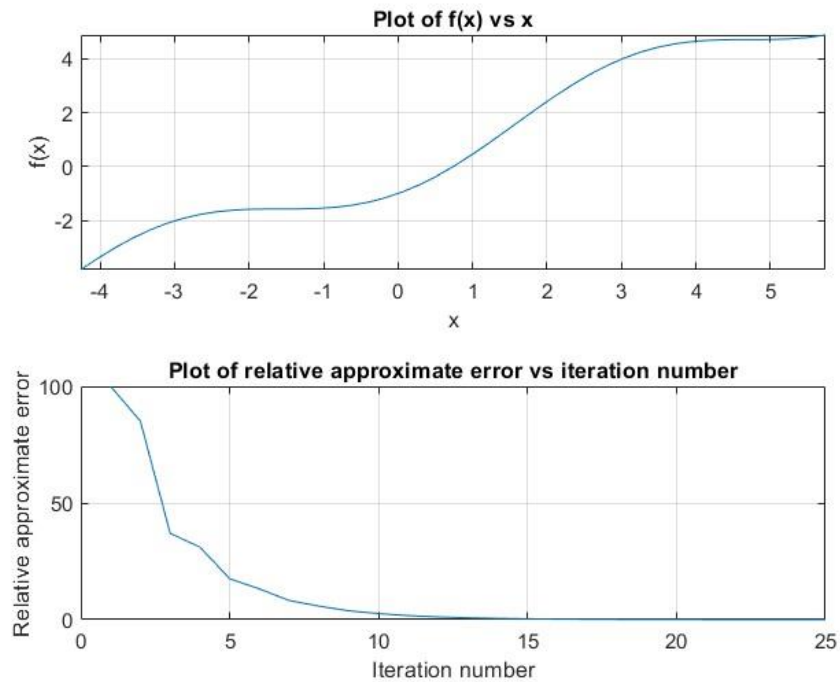
False-Position Method is always stable and the value of approximate percentage error decreases at each iteration as we are going nearer to the root in each iteration.

(c) Fixed-Point Method:

(1) f(x) = x – cos(x)

Output: The root is 0.739106

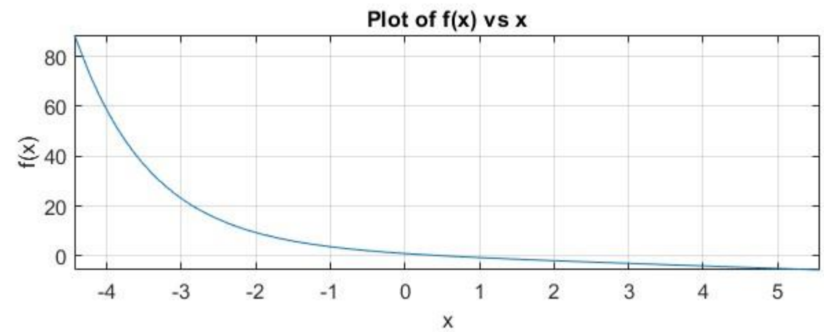
```
Command Window
>> Main
Choose method to find the root. Type:
 1 for Bisection Method
 2 for False-Position Method
 3 for Fixed-Point Method
 4 for Newton-Raphson Method
 5 for Secant Method
 So which method do you like to use: 3
Enter a function in x: x-cos(x)
Enter a function such that x=g(x): cos(x)
Enter starting point: 0
Enter maximum no. of iterations: 50
Enter maximum error: 0.01
fx The root is 0.739106>>
```

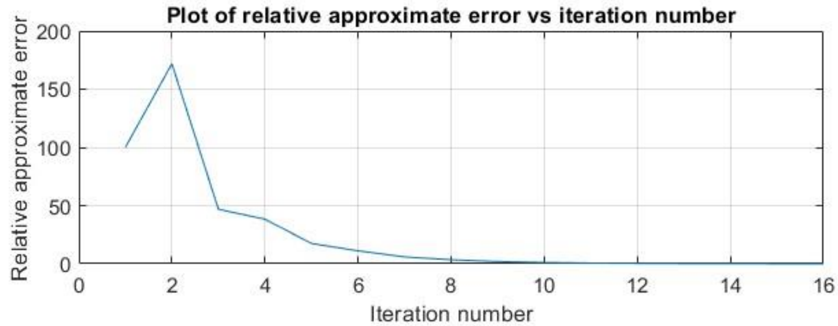


(2) f(x) = exp(-x) - x

Output: The root is 0.567068

```
Command Window
>> Main
Choose method to find the root. Type:
 1 for Bisection Method
 2 for False-Position Method
 3 for Fixed-Point Method
 4 for Newton-Raphson Method
 5 for Secant Method
 So which method do you like to use: 3
Enter a function in x: exp(-x)-x
Enter a function such that x=g(x): exp(-x)
Enter starting point: 0
Enter maximum no. of iterations: 50
Enter maximum error: 0.05
fx The root is 0.567068>>
```





<u>Convergence of Fixed-Point Method</u>:

Fixed-point method has a broad spectrum of convergence but at worst, it is linear. As seen in our second test case, the Fixed-Point Method initially diverges but later become convergent.

<u>Stability of Fixed-Point Method</u>:

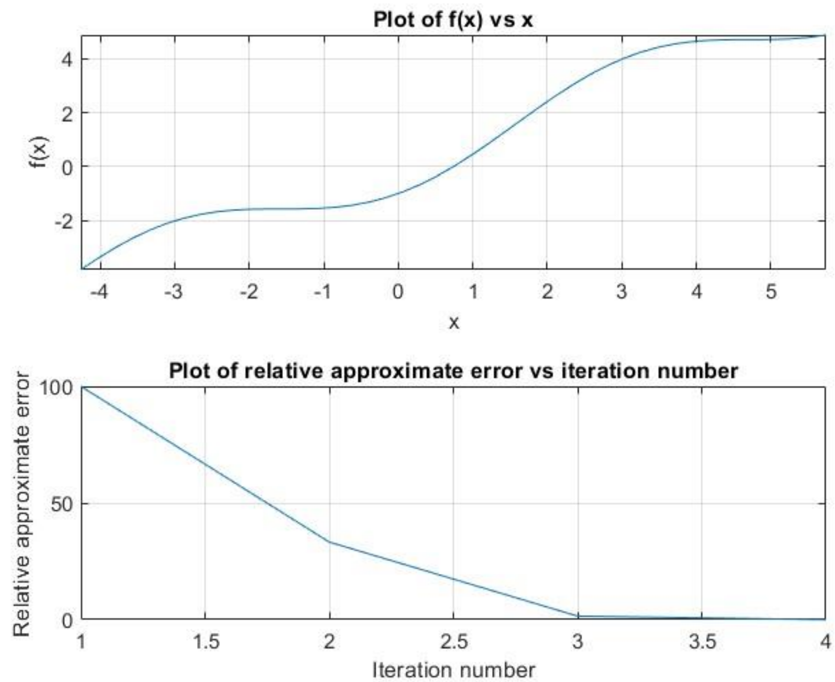
In our second test case, the Fixed-Point Method was not stable as the approximate error percentage value initially increases.

(d) Newton-Raphson Method:

(1) f(x) = x – cos(x)

Output: The root is 0.739085

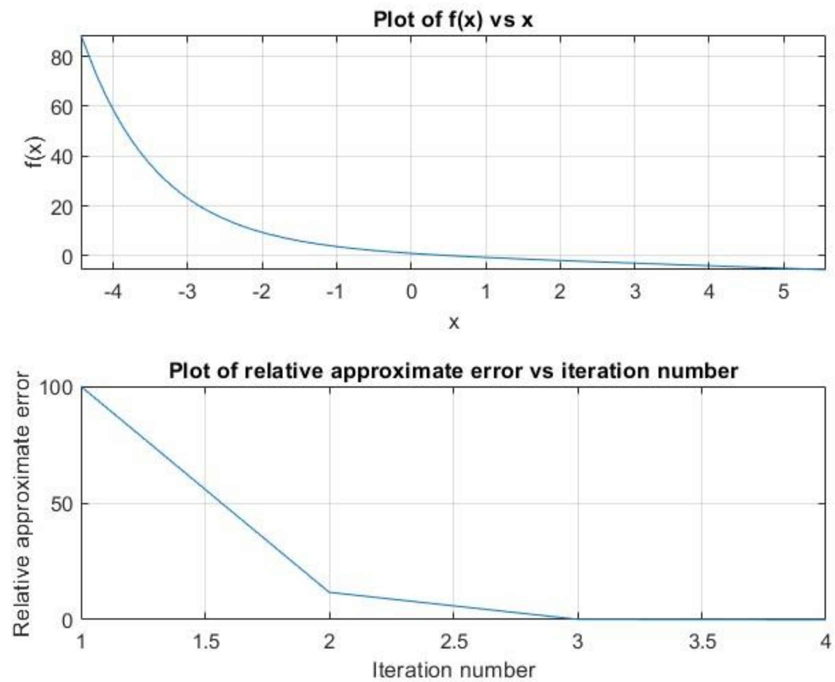
```
Command Window
>> Main
Choose method to find the root. Type:
 1 for Bisection Method
 2 for False-Position Method
 3 for Fixed-Point Method
 4 for Newton-Raphson Method
 5 for Secant Method
 So which method do you like to use: 4
Enter a function in x: x-cos(x)
Enter derivative of function: 1+sin(x)
Enter starting point: 0
Enter maximum no. of iterations: 50
Enter maximum error: 0.01
fx The root is 0.739085>>
```



(2) f(x) = exp(-x) - x

Output: The root is 0.567143

```
Command Window
>> Main
Choose method to find the root. Type:
 1 for Bisection Method
 2 for False-Position Method
 3 for Fixed-Point Method
 4 for Newton-Raphson Method
 5 for Secant Method
 So which method do you like to use: 4
Enter a function in x: exp(-x)-x
Enter derivative of function: -exp(-x)-1
Enter starting point: 0
Enter maximum no. of iterations: 50
Enter maximum error: 0.05
fx The root is 0.567143>>
```



Convergence of Newton-Raphson Method:

The rate of convergence of Newton-Raphson Method is generally between linear and quadratic but as we reach closer to the root, it tends to becoming quadratic.

Stability of Newton-Raphson Method:

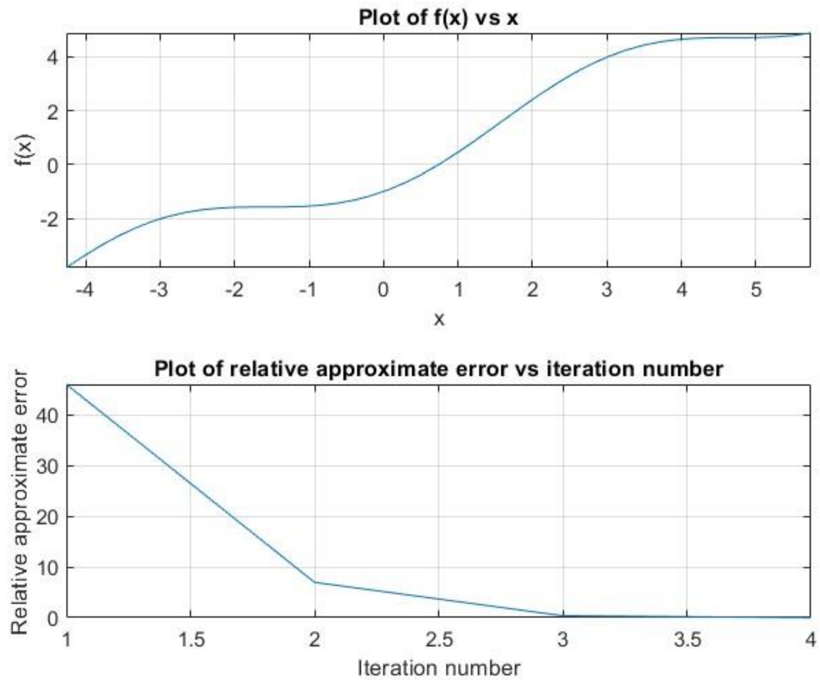
Newton-Raphson Method is always stable.

(e) Secant Method:

(1) f(x) = x – cos(x)

Output: The root is 0.739119

```
Command Window
>> Main
Choose method to find the root. Type:
 1 for Bisection Method
 2 for False-Position Method
 3 for Fixed-Point Method
 4 for Newton-Raphson Method
 5 for Secant Method
 So which method do you like to use: 5
Enter a function in x: x-cos(x)
Enter starting point1: 0
Enter starting point2: 1
Enter maximum no. of iterations: 50
Enter maximum error: 0.01
fx The root is 0.739119>>
```

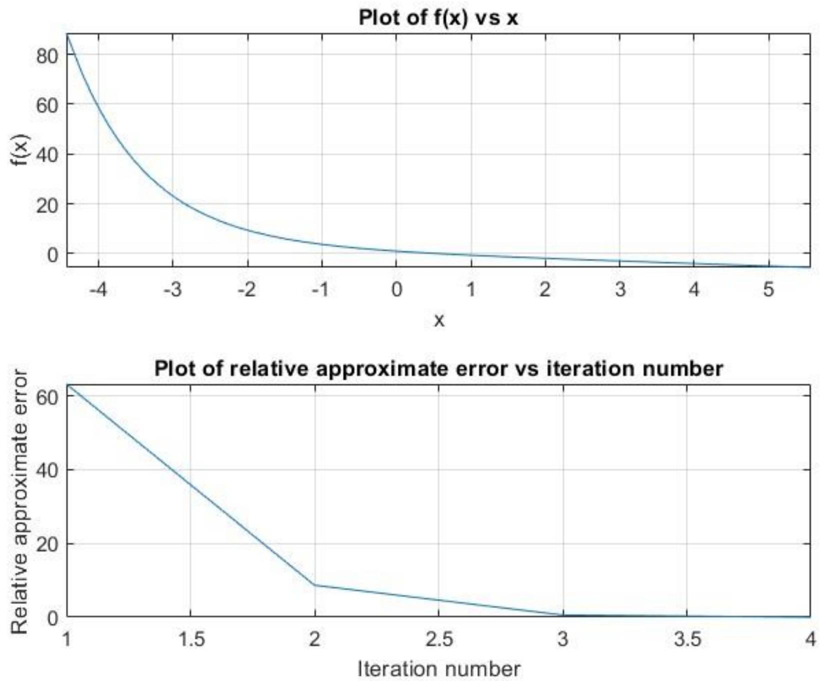


(2) f(x) = exp(-x) - x

Output: The root is 0.567170

Command Window

```
>> Main
Choose method to find the root. Type:
 1 for Bisection Method
 2 for False-Position Method
 3 for Fixed-Point Method
 4 for Newton-Raphson Method
 5 for Secant Method
 So which method do you like to use: 5
Enter a function in x: exp(-x)-x
Enter starting point1: 0
Enter starting point2: 1
Enter maximum no. of iterations: 50
Enter maximum error: 0.05
fx The root is 0.567170>>
```



<u>Convergence of Secant Method</u>:

The rate of convergence of Secant Method is quadratic but if the multiplicity of the root is larger than one, the convergence of the secant method becomes linear.

<u>Stability of Secant Method</u>:

Secant Method is usually stable and the approximate error percentage value decreases regularly.

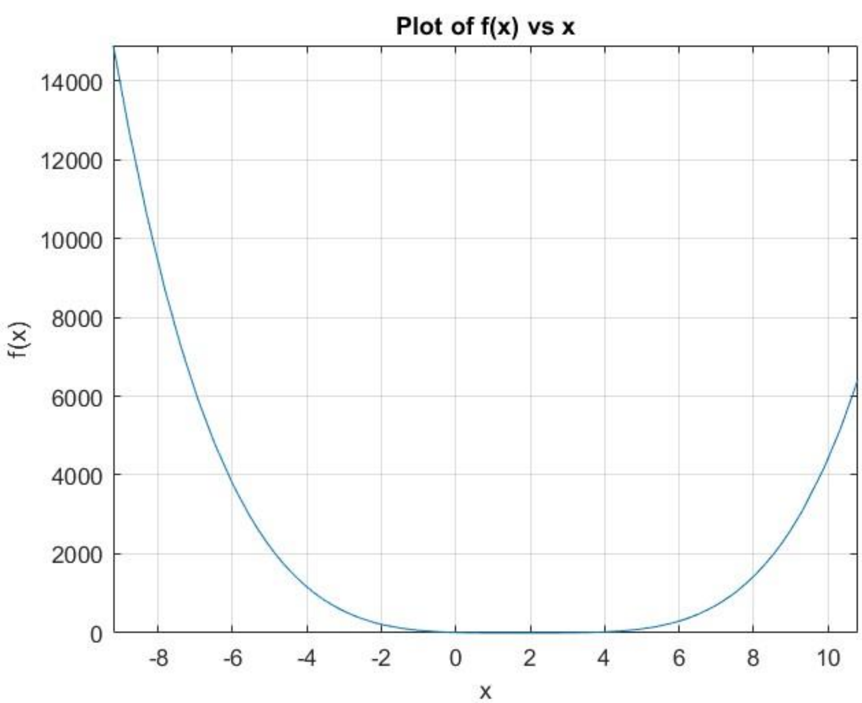
Ans2

(a) Muller Method:

(1) Starting Values : (-1,0,1)

Output: The root is 0.800000

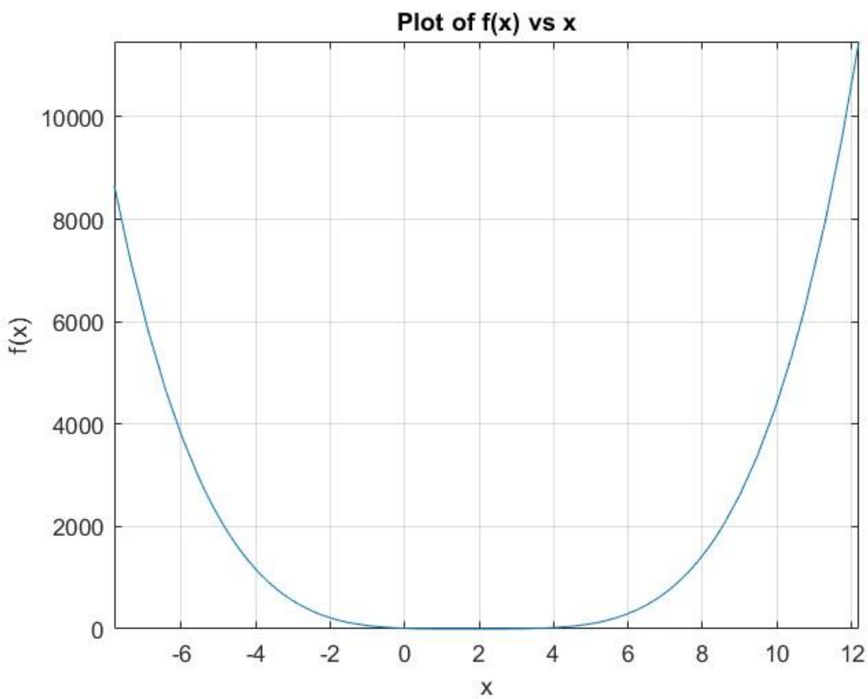
```
Command Window
>> Main
Choose method to find the root. Type:
 1 for Muller Method
 2 for Bairstrow Method
 So which method do you like to use: 1
Enter a polynomial in x: x.^4 - 7.4*x.^3 + 20.44*x.^2 - 24.184*x + 9.6448
Enter starting point1: -1
Enter starting point2: 0
Enter starting point3: 1
Enter maximum no. of iterations: 50
Enter maximum error: 0.01
fx The root is 0.800000>>
```



(2) Starting Values : (0,1,2)

Output: The root is 2.200000

```
Command Window
>> Main
Choose method to find the root. Type:
 1 for Muller Method
 2 for Bairstow Method
 So which method do you like to use: 1
Enter a polynomial in x: x.^4  - 7.4*x.^3 + 20.44*x.^2 - 24.184*x + 9.6448
Enter starting point1: 0
Enter starting point2: 1
Enter starting point3: 2
Enter maximum no. of iterations: 50
Enter maximum error: 0.01
fx The root is 2.200000>>
```



<u>Convergence of Muller Method</u>:

The rate of convergence of Muller Method is between linear and quadratic.

<u>Stability of Muller Method</u>:

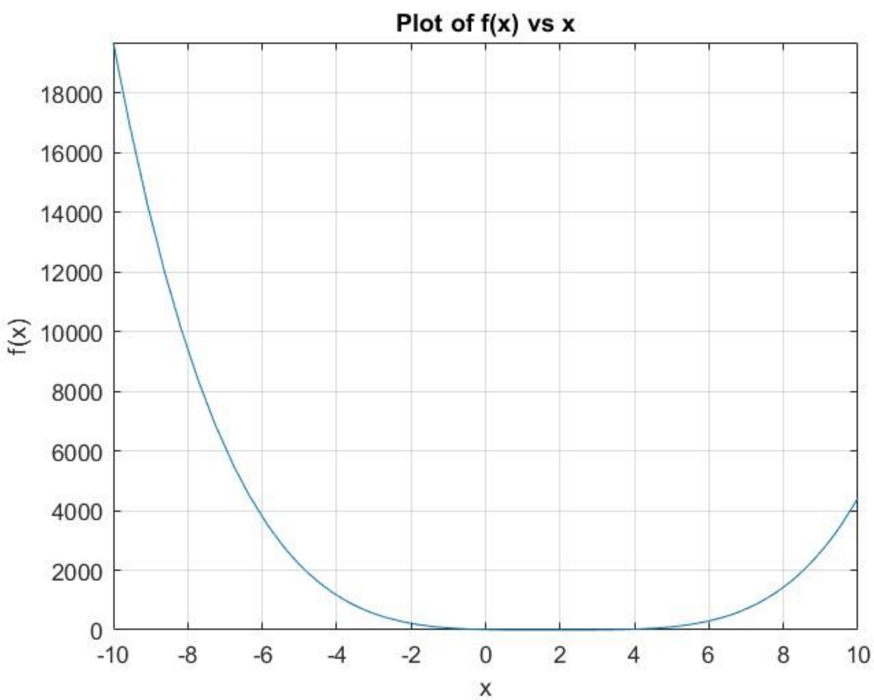
Muller Method is usually stable and the approximate error percentage value decreases regularly.

(b) Bairstow Method:

(1) Starting Values : (-5,4)

Output: The roots are 2.200000 and 0.800000

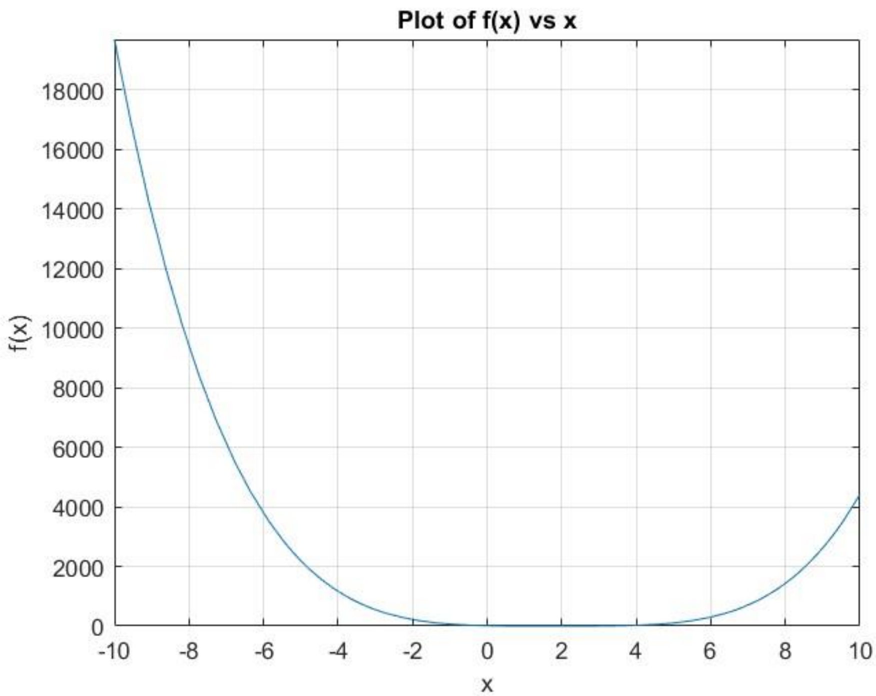
```
Command Window
>> Main
Choose method to find the root. Type:
 1 for Muller Method
 2 for Bairstrow Method
 So which method do you like to use: 2
Enter a polynomial in x: x.^4  - 7.4*x.^3 + 20.44*x.^2 - 24.184*x + 9.6448
Enter starting value1: -5
Enter starting value2: 4
Enter maximum no. of iterations: 50
Enter maximum error: 0.01
fx The roots are 2.200000 and 0.800000>>
```



(2) Starting Values : (-2,2)

Output: The roots are 2.200000 and 0.800000

```
Command Window
>> Main
Choose method to find the root. Type:
 1 for Muller Method
 2 for Bairstrow Method
 So which method do you like to use: 2
Enter a polynomial in x: x.^4  - 7.4*x.^3 + 20.44*x.^2 - 24.184*x + 9.6448
Enter starting value1: -2
Enter starting value2: 2
Enter maximum no. of iterations: 50
Enter maximum error: 0.01
fx The roots are 2.200000 and 0.800000>>
```



Convergence of Bairstow Method:

Bairstow's algorithm inherits the local quadratic convergence of Newton's method, except in the case of quadratic factors of multiplicity higher than 1, when convergence to that factor is linear.

Stability of Bairstow Method:

Bairstow Method is usually stable. A particular kind of instability is observed in Bairstow method when the polynomial has odd degree and only one real root.