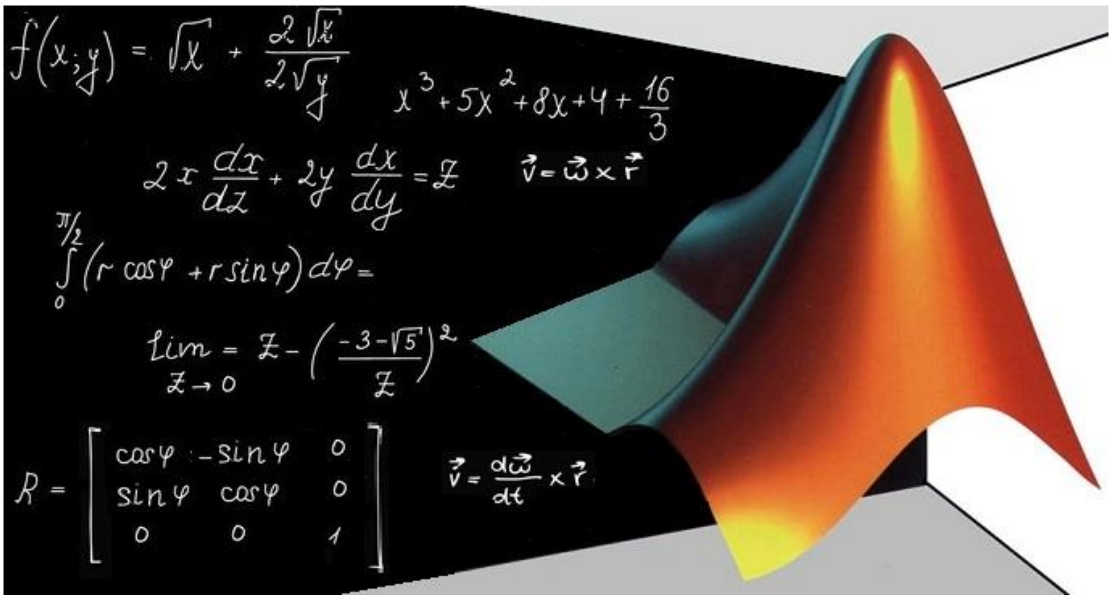
NUMERICAL METHODS

MATH2059

MATLAB PROJECT REPORT



$f(x;y) = \sqrt{x} + \frac{2\sqrt{x}}{2\sqrt{y}}$
$x^3+5x^2+8x+4+\frac{16}{3}$
$2x\frac{dx}{dz}+2y\frac{dx}{dy}=z$
$\vec{v}=\vec{\omega}\times\vec{r}$
$\int_0^{\pi/2}(r\cos\varphi+r\sin\varphi)d\varphi=$
$\lim_{z\to0}= z-\left(\frac{-3-\sqrt{5}}{z}\right)^2$
$R=\begin{bmatrix}\cos\varphi & -\sin\varphi & 0\\ \sin\varphi & \cos\varphi & 0\\ 0 & 0 & 1\end{bmatrix}$
$\vec{v}=\frac{d\vec{\omega}}{dt}\times\vec{r}$

UFUK GÜRBÜZ – 150113058

PROCEDURES

STEP 1

I created a ‘plotpder’ function that takes four inputs (coefficent, x1, x2, number of points) . It describes polinomial equation and plots graphs equation and derivative of equation. It shows them in the same plot. I added title, labels for two axes, a legion, polynomial equation and derivative of equation into graph.

Firstly, the user should define a coefficent vector for equation.

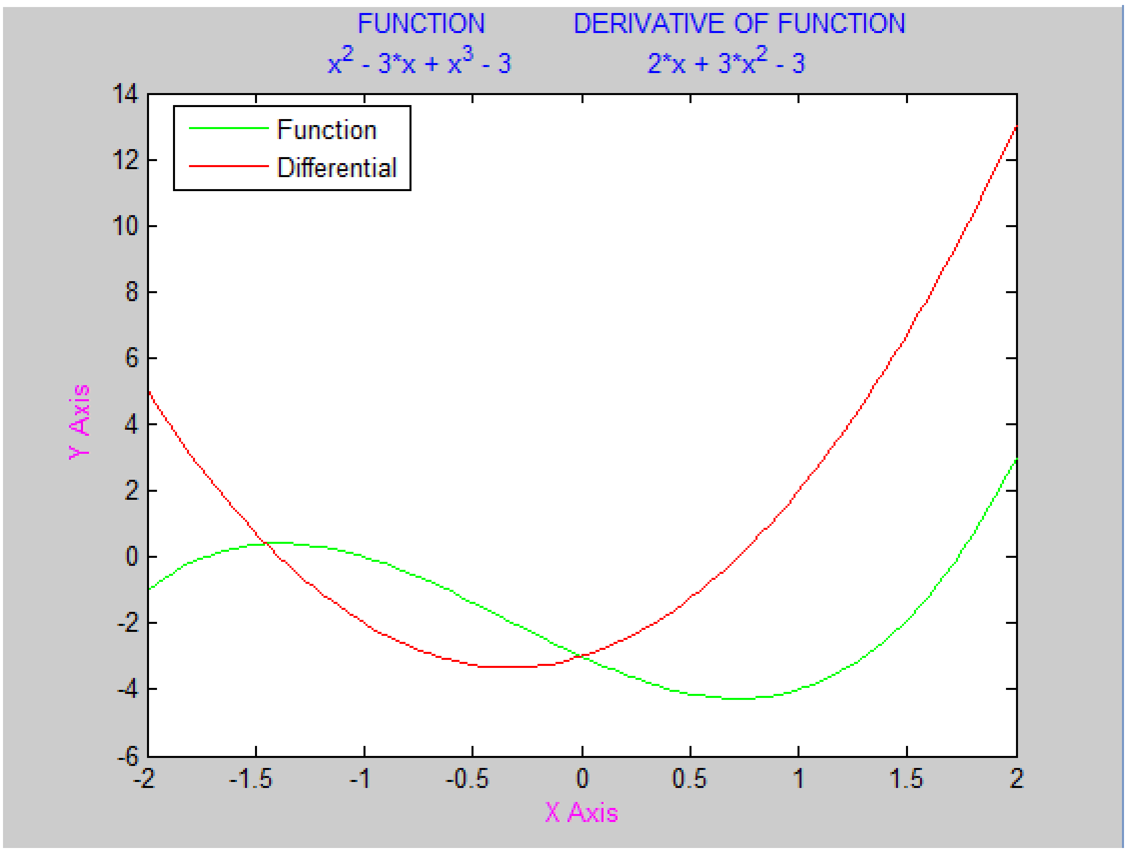
```
>> c=[1 1 -3 -3];
```

Then, the user should call function with necessary parameters.

```
>> plotpder(c,2,-2,100)
```

“c” is coefficent of equation. “2” and “-2” are interval of ‘x1’ and ‘x2’. “100” is number of points between ‘x1’ and ‘x2’. And the function writes equation and plots graph with these values.

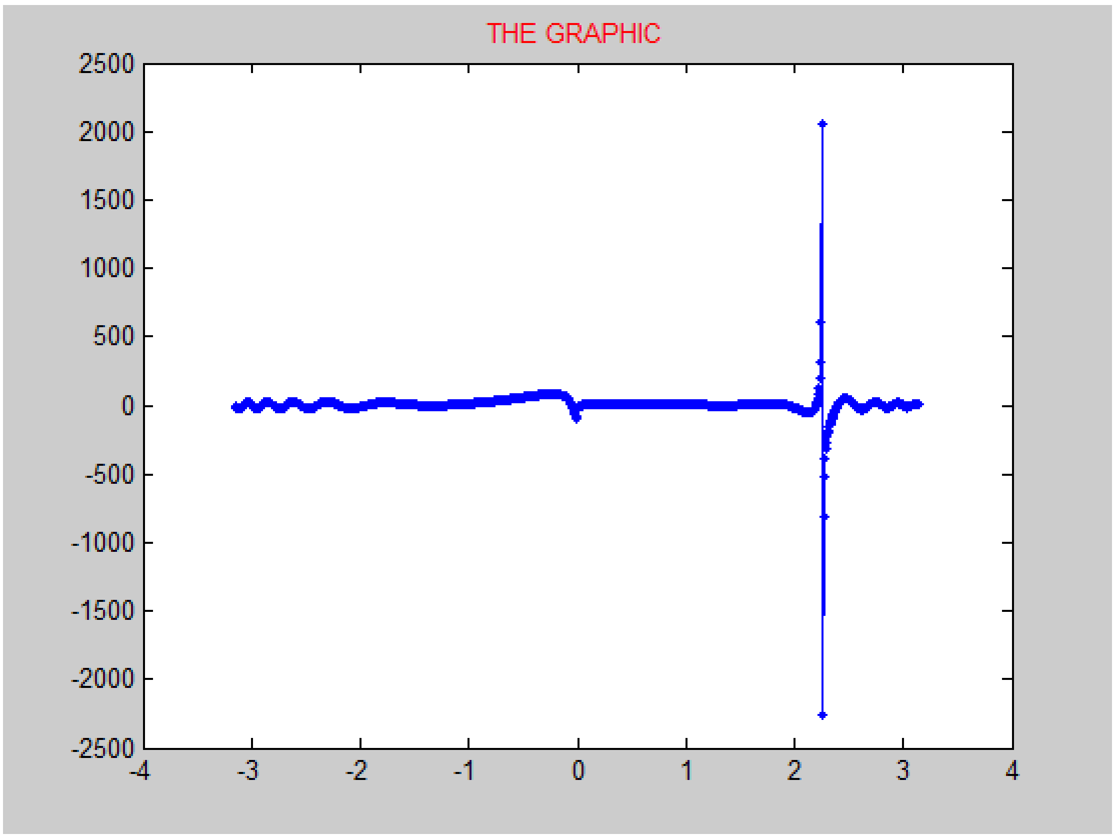
```
>> c=[1 1 -3 -3];
>> plotpder(c,2,-2,100)
    The Equation
--------------------
x^3 + x^2 - 3*x - 3
```



STEP 2

I created a 'funcplot' script that takes no input and returns no output. It shows results of " f(x) = 5cos (x4/3)tan(e0.2x)cos(ln(4x)) " on interval [-pi, pi] using 1000 linearly spaced data points. I defined 1000 times 'x' points on interval [-pi, pi] and found 'y' points using f(x) equation. Then, the script plots 'x' – 'y' values.

The function should call as `>> funcplot()` . The graph of f(x) is below.



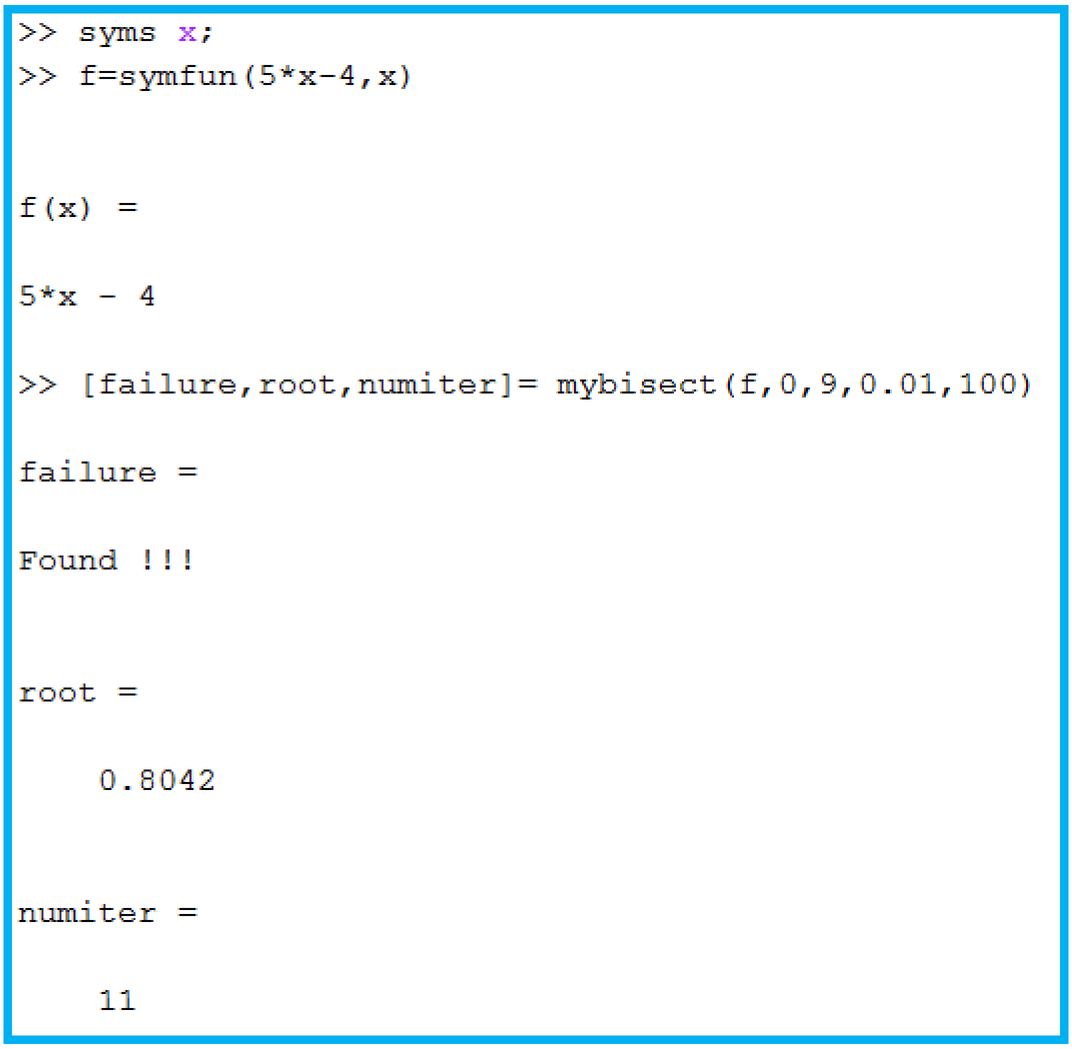
STEP 3

a)

I created a 'mybisect' function that takes five inputs and returns three outputs. The input parameters are polynomial equation, first interval value, last interval value, tolerance value and maximum number of iterations. The output parameters are failure condition, root of equation and total number of iterations. It finds root of equation using Bisection Algorithm.

The user should define a symbolic value as `>> syms x;` . Then, user should creates a symbolic 'f(x)' function as `>> f=symfun(5*x-4,x)` . Finally, user call 'mybisect' function as `>> [failure,root,numiter]= mybisect(f,0,9,0.01,100)` .

The results of ‘mybisect’ function are below :



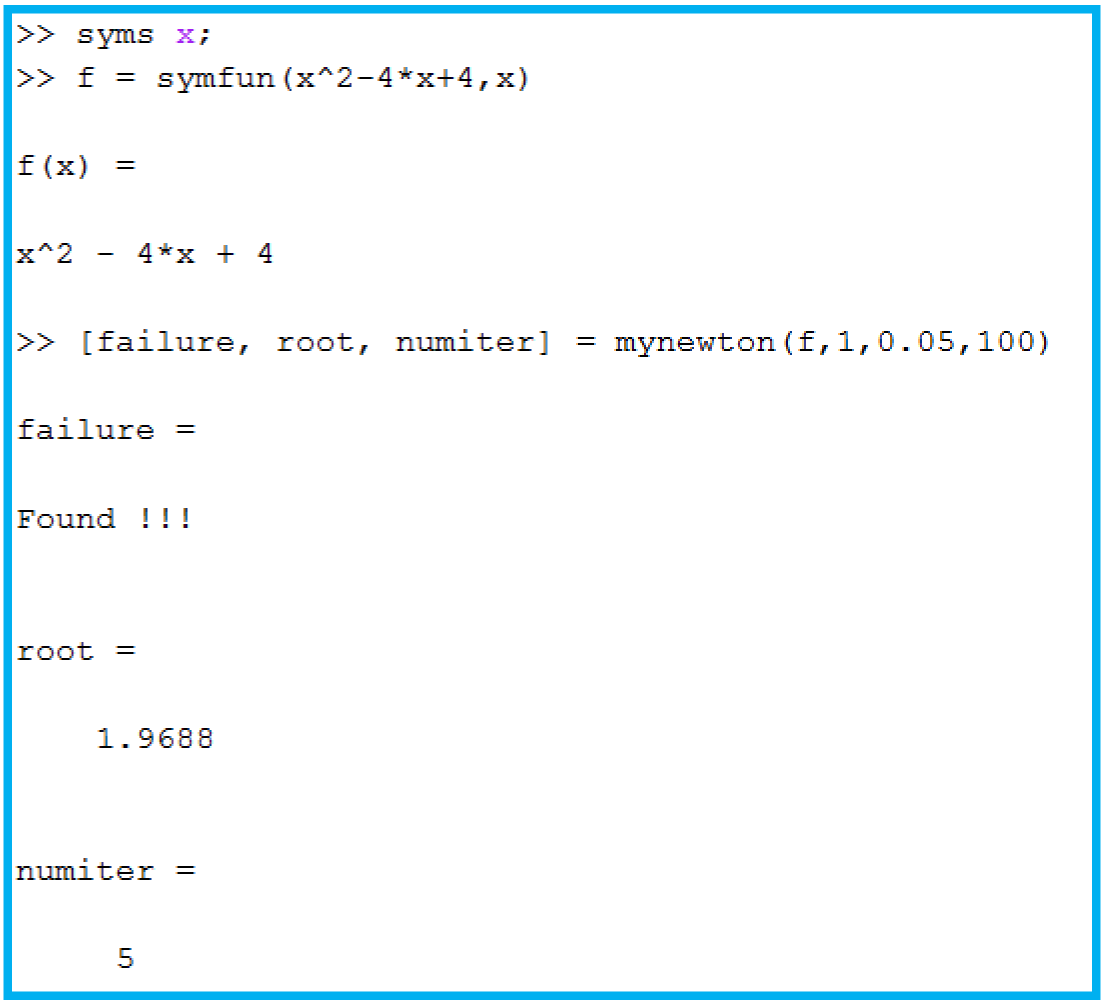
```
>> syms x;
>> f=symfun(5*x-4,x)


f(x) =

5*x - 4

>> [failure,root,numiter]= mybisect(f,0,9,0.01,100)

failure =

Found !!!


root =

    0.8042


numiter =

    11
```

b)

I created a ‘mynewton’ function that takes four inputs and returns three outputs. The input parameters are polynomial equation, beginning root value, tolerance value and maximum number of iterations. The output parameters are failure condition, root of equation and total number of iterations.

The user should define a symbolic value as `>> syms x;` . Then, user should creates a symbolic ‘f(x)’ function as `>> f=symfun(x^2-4*x+4,x)` . Finally, user call ‘mynewton’ function as `>> [failure,root,numiter]=mynewton(f,1,0.05,100)` .

The results of ‘mynewton’ function are below :



```
>> syms x;
>> f = symfun(x^2-4*x+4,x)

f(x) =

x^2 - 4*x + 4

>> [failure, root, numiter] = mynewton(f,1,0.05,100)

failure =

Found !!!


root =

    1.9688


numiter =

     5
```

c)

	BISECTION			NEWTON		
Equation	**Failure**	**Root**	**Numiter**	**Failure**	**Root**	**Numiter**
1	F	1.7320	16	F	1.7321	4
2	F	1.7321	29	F	1.7321	5
3	F	5.1253	17	F	- 1.1653	6
4	F	5.1253	30	F	- 1.1653	7
5	F	- 0.0068	15	F	- 0.0068	2
6	F	- 0.0068	28	F	- 0.0068	3

I saw that “ the newton method ” is comparing less than “ the section method ”. Namely, “ the newton method ” is faster than “ the bisection method ”. So, it can find roots of equations fastly.