PROJECT #1

Q1 Answers:

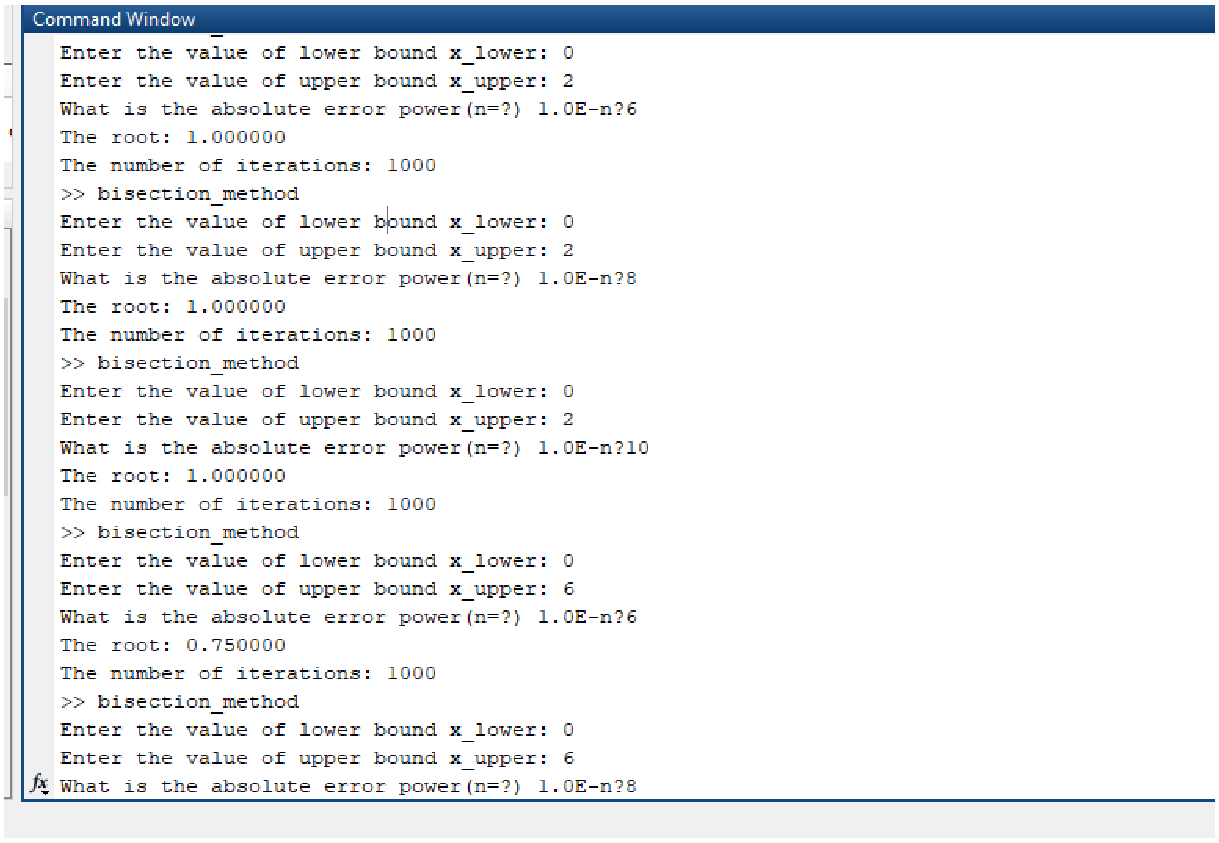
a) Bisection Method:

This method is based on the intermediate value theorem for continuous functions and my code is the following:

```
 myfunction = @(x) x.^3+2*x.^2+10*x-20;   %FUNCTION HANDLING

x_lower = input('Enter the value of lower bound x_lower: ');
x_upper = input('Enter the value of upper bound x_upper: ');
n = input('What is the absolute error power(n=?) 1.0E-n?');

% In any case I checked that no root exist or end_points are roots of the
% myfunction, if so I do not have do any loop iterations
if myfunction(x_lower)*myfunction(x_upper) > 0
    fprintf('No roots exist within the given interval\n');
    return
end

if myfunction(x_lower) == 0
    fprintf('x_lower is a root of equation\n');
    return
elseif myfunction(x_upper) == 0
    fprintf('x_upper is a root of equation\n');
    return
end


% I calculated my end points on the myfunction
f_lower = myfunction(x_lower);
f_upper = myfunction(x_upper);

%%%%%%%%  MAIN LOOP %%%%%%%

%% To avoid too many iterations we can set a max. num of iteration %%%
for i = 1:1000

    x_mid = (x_lower+x_upper)/2; % In each iteration we check the mid point
    f_mid = myfunction(x_mid);

    if f_lower*f_upper < 0
        x_upper = x_mid;  % x_upper is now the mid point
        f_upper = f_mid;
    else
        x_lower = x_mid;  % x_lower is now the mid point
        f_lower = f_mid;
    end
    if abs(myfunction(x_lower)) < 1.0*10^-n  % absolute error criteria
        break
    end
end

%%%%%%%% LOOP ENDS %%%%%%%%%

fprintf('The root: %f\nThe number of iterations: %d\n', x_lower,i);
```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Solutions:


Command Window

```
Enter the value of lower bound x_lower: 0
Enter the value of upper bound x_upper: 2
What is the absolute error power(n=?) 1.0E-n?6
The root: 1.000000
The number of iterations: 1000
>> bisection_method
Enter the value of lower bound x_lower: 0
Enter the value of upper bound x_upper: 2
What is the absolute error power(n=?) 1.0E-n?8
The root: 1.000000
The number of iterations: 1000
>> bisection_method
Enter the value of lower bound x_lower: 0
Enter the value of upper bound x_upper: 2
What is the absolute error power(n=?) 1.0E-n?10
The root: 1.000000
The number of iterations: 1000
>> bisection_method
Enter the value of lower bound x_lower: 0
Enter the value of upper bound x_upper: 6
What is the absolute error power(n=?) 1.0E-n?6
The root: 0.750000
The number of iterations: 1000
>> bisection_method
Enter the value of lower bound x_lower: 0
Enter the value of upper bound x_upper: 6
What is the absolute error power(n=?) 1.0E-n?8
```

Discussion Part on the Bisection Method:

I tried the function with the different absolute criteria and different lower upper bounds which are basically taken as an input from the user. So, I could understand how those value changes effects the success of my solutions, how I am close to finding the real roots of my equation by changing absolute criteria and the bound values.

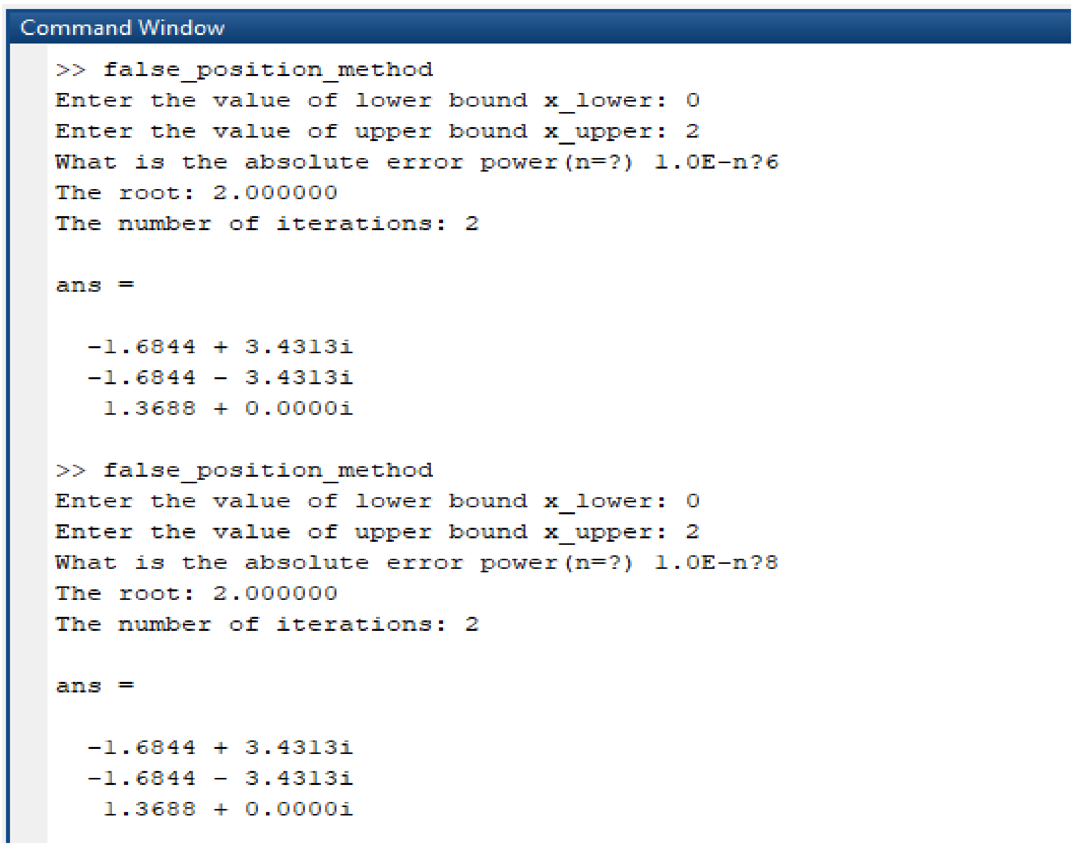
The choice of starting interval , lower upper bounds , is important on the success of the bisection method however, changing of the absolute error power is not certain on those results. Both selecting the interval too wide or narrow causes a problem in terms of prediction of the roots correctly. However, the main conclusion on Bisection Method is that the method is working slower than other methods. The tolerance is the absolute value of the difference between the actual root of the function x and the approximation c : $\varepsilon=|c-x|$. And I used it to break the loop in Matlab code above.

b) False Position Method:

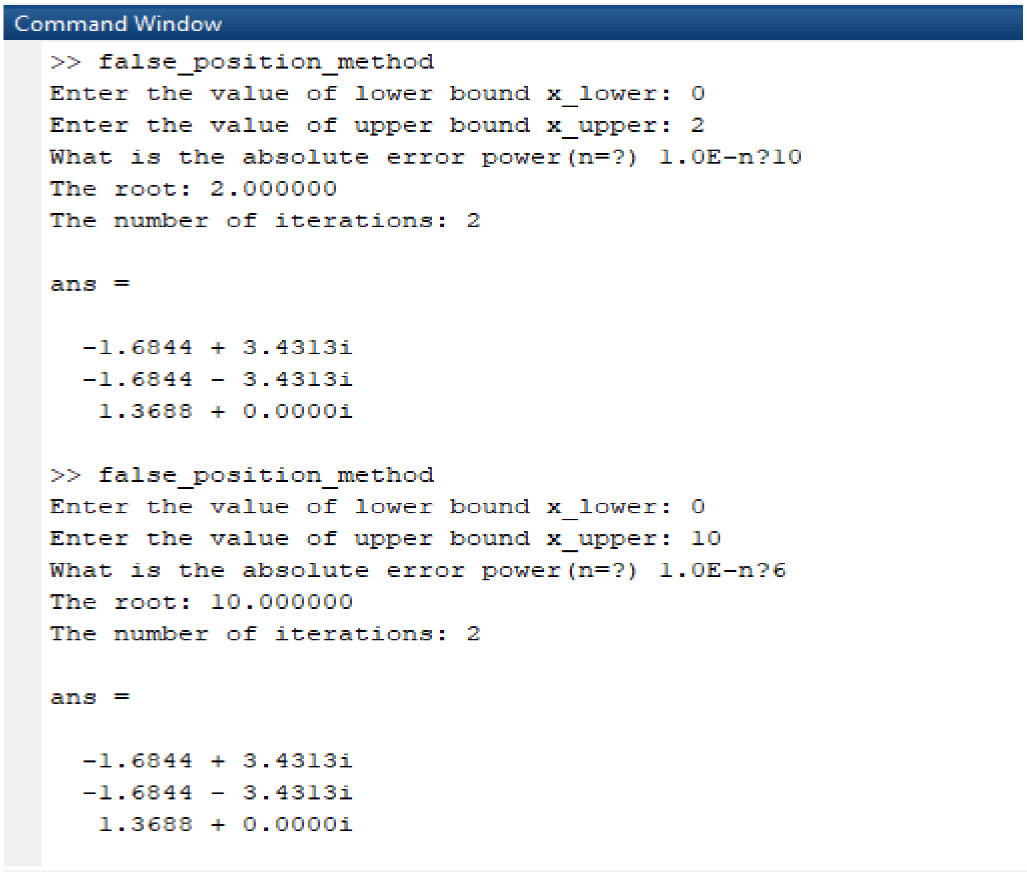
My function is following:

```
myfunction = @(x) x.^3+2*x.^2+10*x-20;  %FUNCTION HANDLING
```

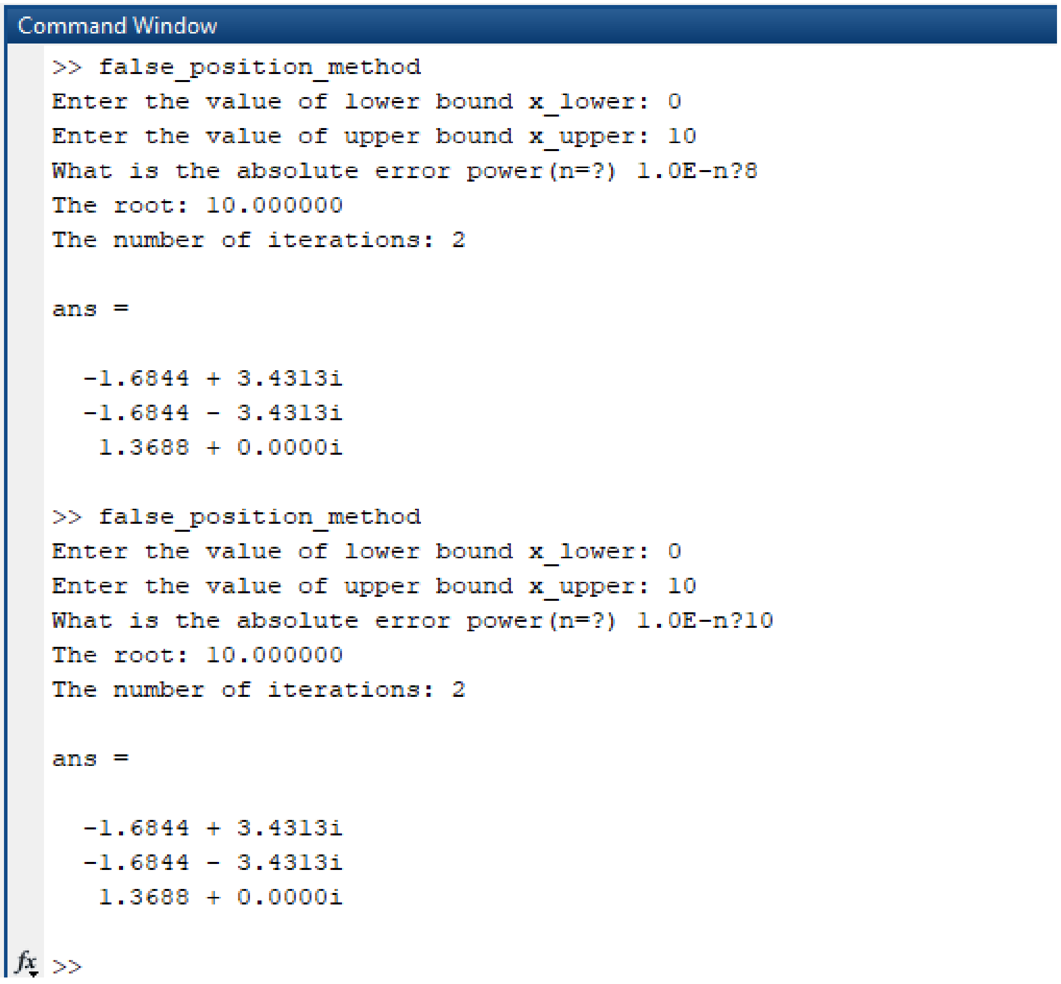
```
%I took interval boundaries and the absolute error as an input so that I
%can try for different values easily

x_lower = input('Enter the value of lower bound x_lower: ');
x_upper = input('Enter the value of upper bound x_upper: ');
n = input('What is the absolute error power(n=?) 1.0E-n?');

% I calculated my end_points on my function
f_lower = myfunction(x_lower);
f_upper = myfunction(x_upper);
x0 = x_lower;
iter = 0; % I count the num of iterations

%I would like to see myfunction visual on that interval so that I can come
%up with an prediction on my root
s = linspace(x_lower,x_upper,100);
f = myfunction(s);
fig = figure();
plot(s,f)

%Loop starts here and it continues until the absolute error criteria
%becomes less than the tolerance

while true
   xr = x_upper-f_upper*(x_upper - x_upper)/(f_upper-f_lower);
   fr = myfunction(xr);
   iter = iter + 1;

   if sign(fr) == sign(f_lower)  %I checked signs so that I can update my
boundaries accordingly
       x_lower = xr;
       f_lower = fr;
   else
       x_upper = xr;
       f_upper = fr;
   end
if (abs((xr-x0)/xr) < 1.0*10^-n)  %I checked the absolute error here
    break;
else
    x0 = xr;
end
end

fprintf('The root: %f\nThe number of iterations: %d\n', xr, iter);

roots([1 2 10 -20])  %I just wanted to check whether my method is close
enough for prediction of the root

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Solutions:
```



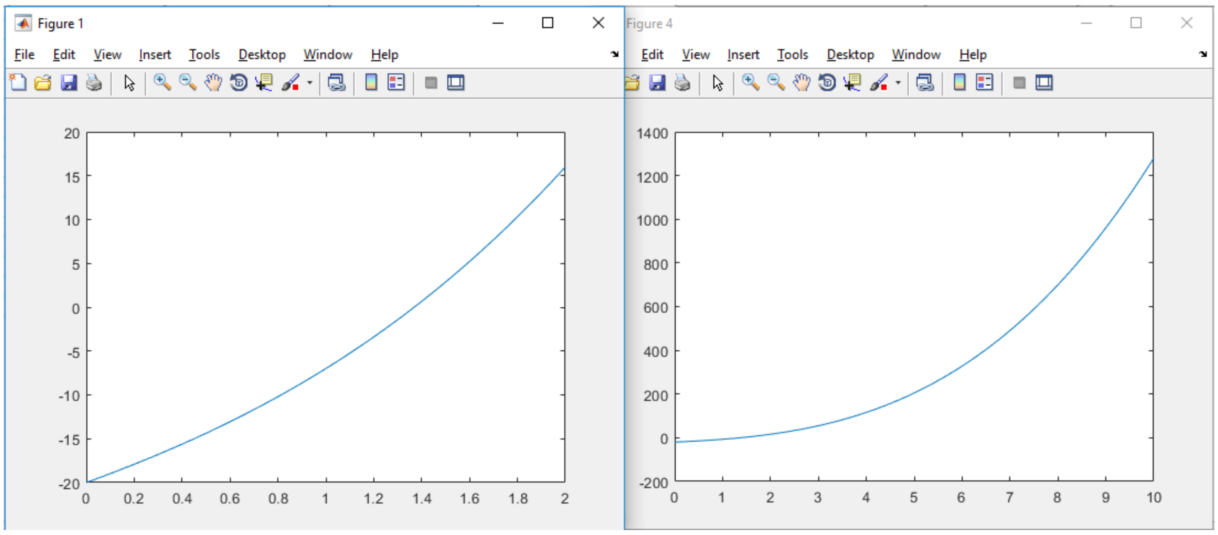
```
>> false_position_method
Enter the value of lower bound x_lower: 0
Enter the value of upper bound x_upper: 2
What is the absolute error power(n=?) 1.0E-n?6
The root: 2.000000
The number of iterations: 2

ans =

  -1.6844 + 3.4313i
  -1.6844 - 3.4313i
   1.3688 + 0.0000i

>> false_position_method
Enter the value of lower bound x_lower: 0
Enter the value of upper bound x_upper: 2
What is the absolute error power(n=?) 1.0E-n?8
The root: 2.000000
The number of iterations: 2

ans =

  -1.6844 + 3.4313i
  -1.6844 - 3.4313i
   1.3688 + 0.0000i
```



```
>> false_position_method
Enter the value of lower bound x_lower: 0
Enter the value of upper bound x_upper: 2
What is the absolute error power(n=?) 1.0E-n?10
The root: 2.000000
The number of iterations: 2

ans =

  -1.6844 + 3.4313i
  -1.6844 - 3.4313i
   1.3688 + 0.0000i

>> false_position_method
Enter the value of lower bound x_lower: 0
Enter the value of upper bound x_upper: 10
What is the absolute error power(n=?) 1.0E-n?6
The root: 10.000000
The number of iterations: 2

ans =

  -1.6844 + 3.4313i
  -1.6844 - 3.4313i
   1.3688 + 0.0000i
```



```
>> false_position_method
Enter the value of lower bound x_lower: 0
Enter the value of upper bound x_upper: 10
What is the absolute error power(n=?) 1.0E-n?8
The root: 10.000000
The number of iterations: 2

ans =

  -1.6844 + 3.4313i
  -1.6844 - 3.4313i
   1.3688 + 0.0000i

>> false_position_method
Enter the value of lower bound x_lower: 0
Enter the value of upper bound x_upper: 10
What is the absolute error power(n=?) 1.0E-n?10
The root: 10.000000
The number of iterations: 2

ans =

  -1.6844 + 3.4313i
  -1.6844 - 3.4313i
   1.3688 + 0.0000i

>>
```



Discussion Part on the False Position Method:

I evaluated the roots of the function again with the different absolute criteria and different interval inputs so that I can easily see how I am close to exact predictions of the roots.

False position method is the another way to obtain roots of the equation but this method retains the main feature of the bisection method, and it is that a root is trapped in a sequence of intervals of decreasing size. Bisection method selects the mid point in each interval and updates the lower and upper bounds. However, this method uses the point where the secant lines intersect with the x axis.

At the above, left figure shows the graph of the function between the interval of [0,2] and the right figure shows the same function in another interval [0,10]. Then, we can visually see that those functions might have different root prediction. Because each time we calculate the point which crosses the x-axis on the line of x_lower and x_upper and update those boundaries according to signs of them.

Even this method is faster and detail than Bisection method, some cases might encourage to use Bisection method as well so all methods must be considered and you should determine the best one for the specific equation.

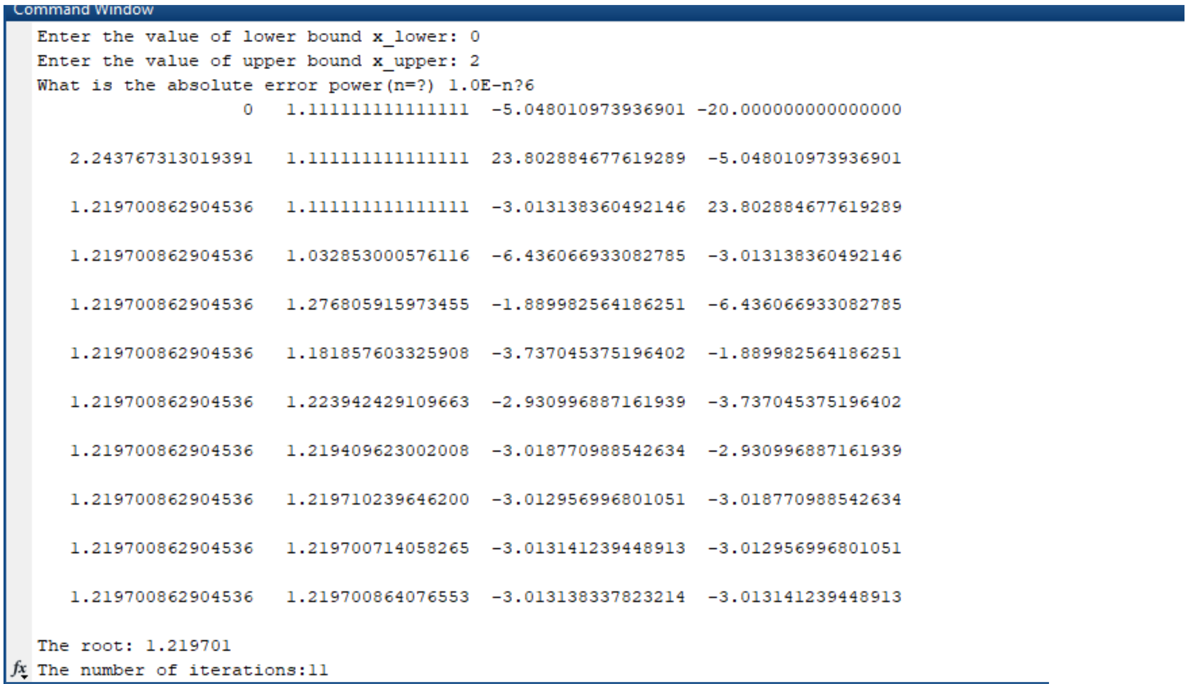
c) Modified False Position Method:

This method is based on the false position method above but the difference is the modification on the bounds not to cause any bound stuck to a value. So code is checking the bounds and if any bound remains the same with the previous calculation then I divided the bound into half.

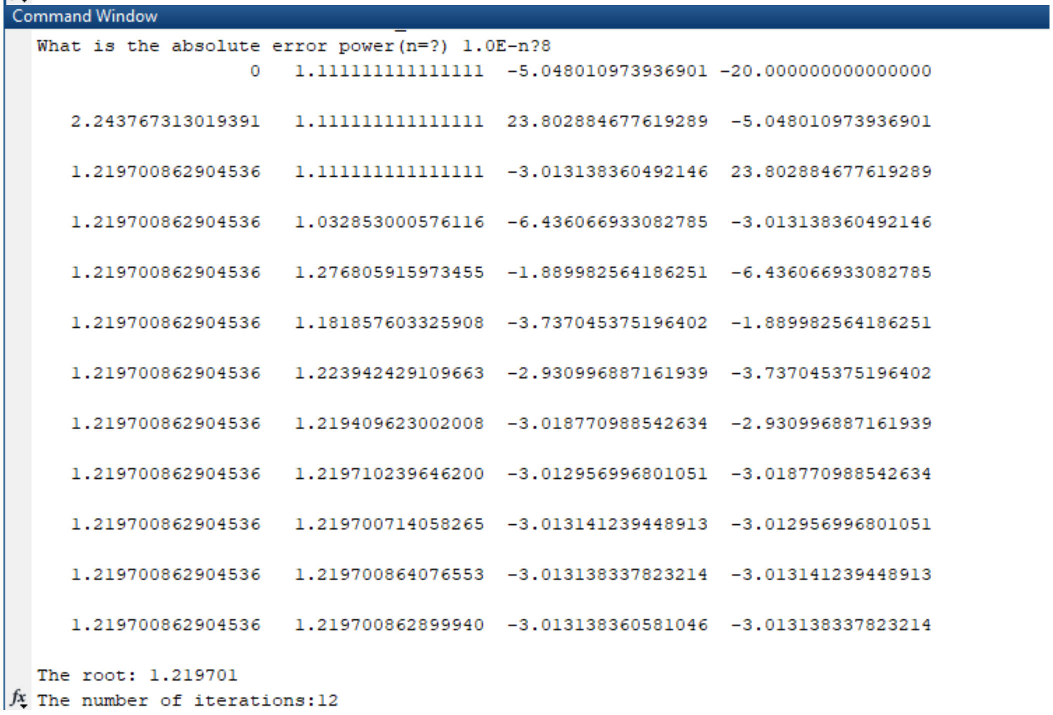
My Matlab code is following:

```
myfunction = @(x) x.^3+2*x.^2+10*x-20;   %FUNCTION HANDLING
```

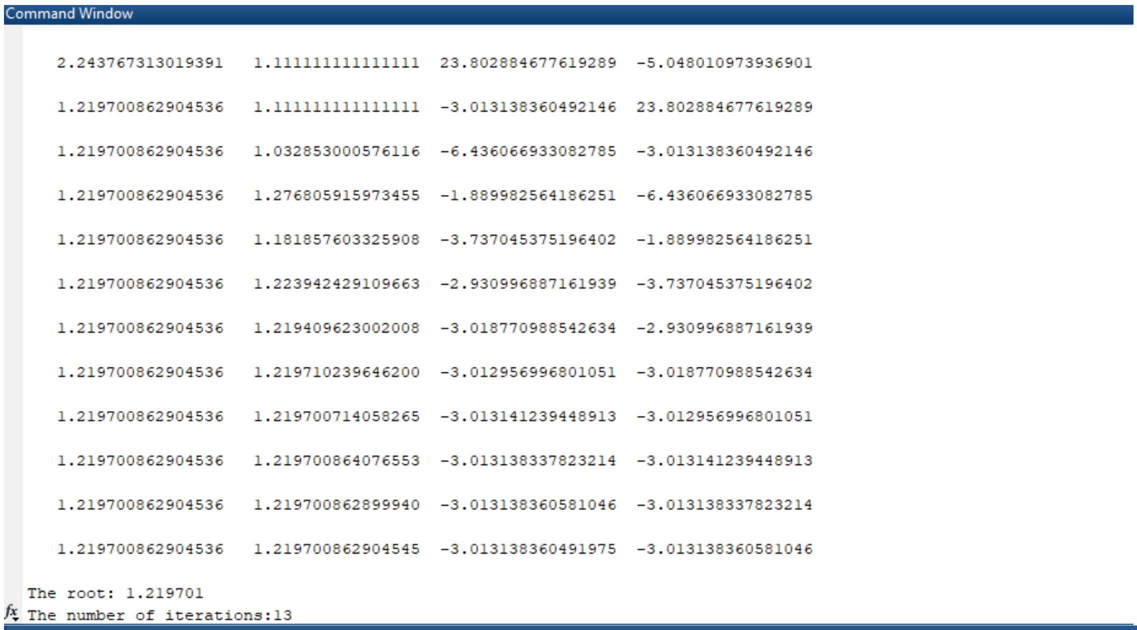
```
 %To make the changes easier I will took interval and the power of the
absolute error as an input:

x_lower = input('Enter the value of lower bound x_lower: ');
x_upper = input('Enter the value of upper bound x_upper: ');
n = input('What is the absolute error power(n=?) 1.0E-n?');

%I calculated the end points on the function.
f_lower = myfunction(x_lower);
f_upper = myfunction(x_upper);
x0= x_lower;
iter = 0;  % I counted the num of iterations
Nmax = 30;
%Loop starts
while iter < Nmax
   xr = (f_upper*x_lower - f_lower*x_upper)/(f_upper-f_lower);
   fr = myfunction(xr);
   f0 = myfunction(x0);
   iter = iter + 1;

   if sign(fr) == sign(f0) %if both negative or positive
       x_upper = xr;
       f_upper = fr;
       f_lower = f_lower /2;
   else
       x_lower = xr;
       f_lower = fr;
       f_upper = f_upper / 2;
   end
disp([x_lower x_upper fr f0])  %I just checked the values

%Absolute error criteria is checked to finish the loop
if (abs((xr-x0)/xr) < 1.0*10^-n) || (fr == f0)  %I checked the absolute
error here
    break;
else
    x0 = xr; %to keep the previous iteration
end
end

fprintf('The root: %f\nThe number of iterations:%d\n', xr, iter);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Solutions:
```



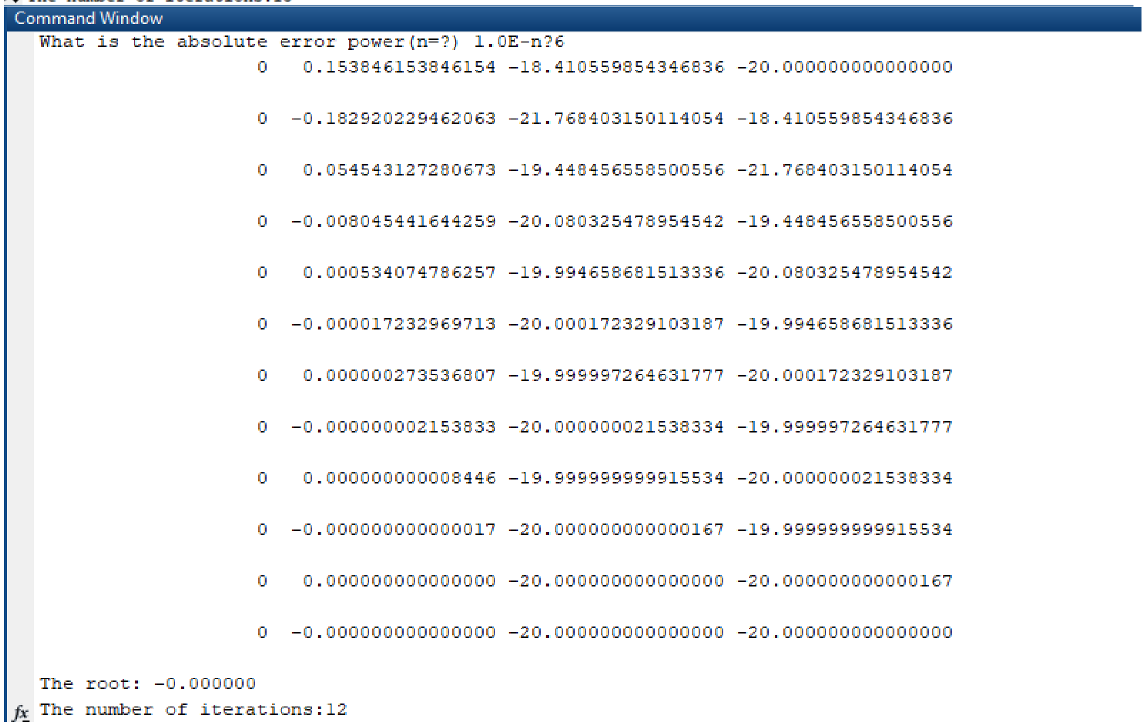
```
Enter the value of lower bound x_lower: 0
Enter the value of upper bound x_upper: 2
What is the absolute error power(n=?) 1.0E-n?6
                  0   1.111111111111111  -5.048010973936901 -20.000000000000000

   2.243767313019391   1.111111111111111  23.802884677619289  -5.048010973936901

   1.219700862904536   1.111111111111111  -3.013138360492146  23.802884677619289

   1.219700862904536   1.032853000576116  -6.436066933082785  -3.013138360492146

   1.219700862904536   1.276805915973455  -1.889982564186251  -6.436066933082785

   1.219700862904536   1.181857603325908  -3.737045375196402  -1.889982564186251

   1.219700862904536   1.223942429109663  -2.930996887161939  -3.737045375196402

   1.219700862904536   1.219409623002008  -3.018770988542634  -2.930996887161939

   1.219700862904536   1.219710239646200  -3.012956996801051  -3.018770988542634

   1.219700862904536   1.219700714058265  -3.013141239448913  -3.012956996801051

   1.219700862904536   1.219700864076553  -3.013138337823214  -3.013141239448913

The root: 1.219701
The number of iterations:11
```



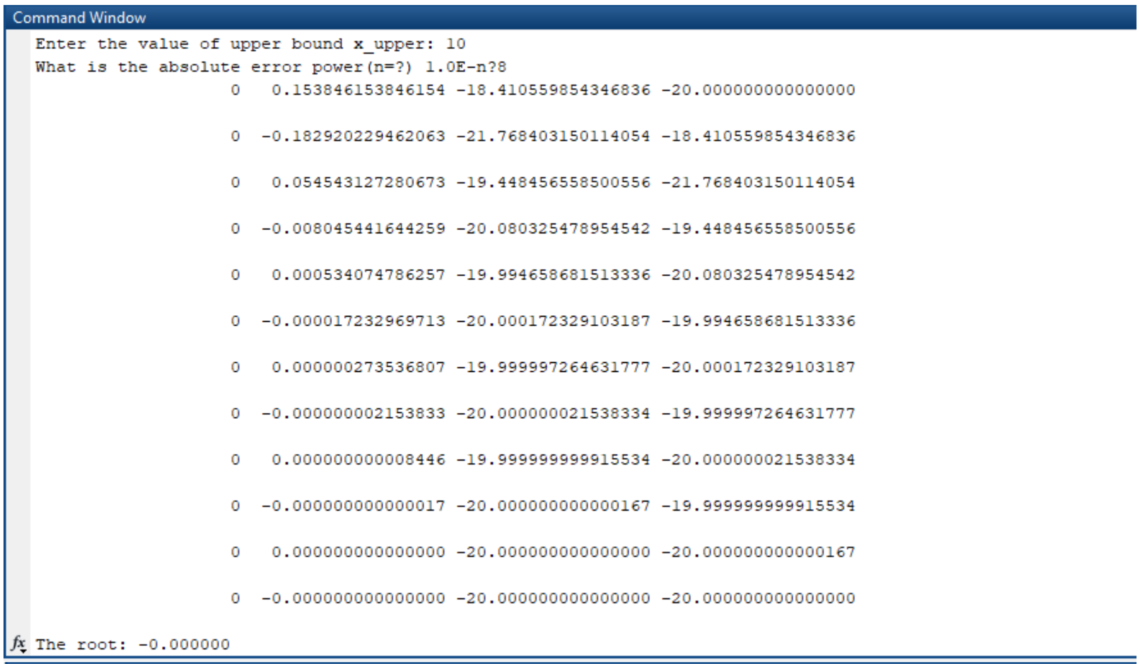
```
What is the absolute error power(n=?) 1.0E-n?8
                  0   1.111111111111111  -5.048010973936901 -20.000000000000000

   2.243767313019391   1.111111111111111  23.802884677619289  -5.048010973936901

   1.219700862904536   1.111111111111111  -3.013138360492146  23.802884677619289

   1.219700862904536   1.032853000576116  -6.436066933082785  -3.013138360492146

   1.219700862904536   1.276805915973455  -1.889982564186251  -6.436066933082785

   1.219700862904536   1.181857603325908  -3.737045375196402  -1.889982564186251

   1.219700862904536   1.223942429109663  -2.930996887161939  -3.737045375196402

   1.219700862904536   1.219409623002008  -3.018770988542634  -2.930996887161939

   1.219700862904536   1.219710239646200  -3.012956996801051  -3.018770988542634

   1.219700862904536   1.219700714058265  -3.013141239448913  -3.012956996801051

   1.219700862904536   1.219700864076553  -3.013138337823214  -3.013141239448913

   1.219700862904536   1.219700862899940  -3.013138360581046  -3.013138337823214

The root: 1.219701
The number of iterations:12
```



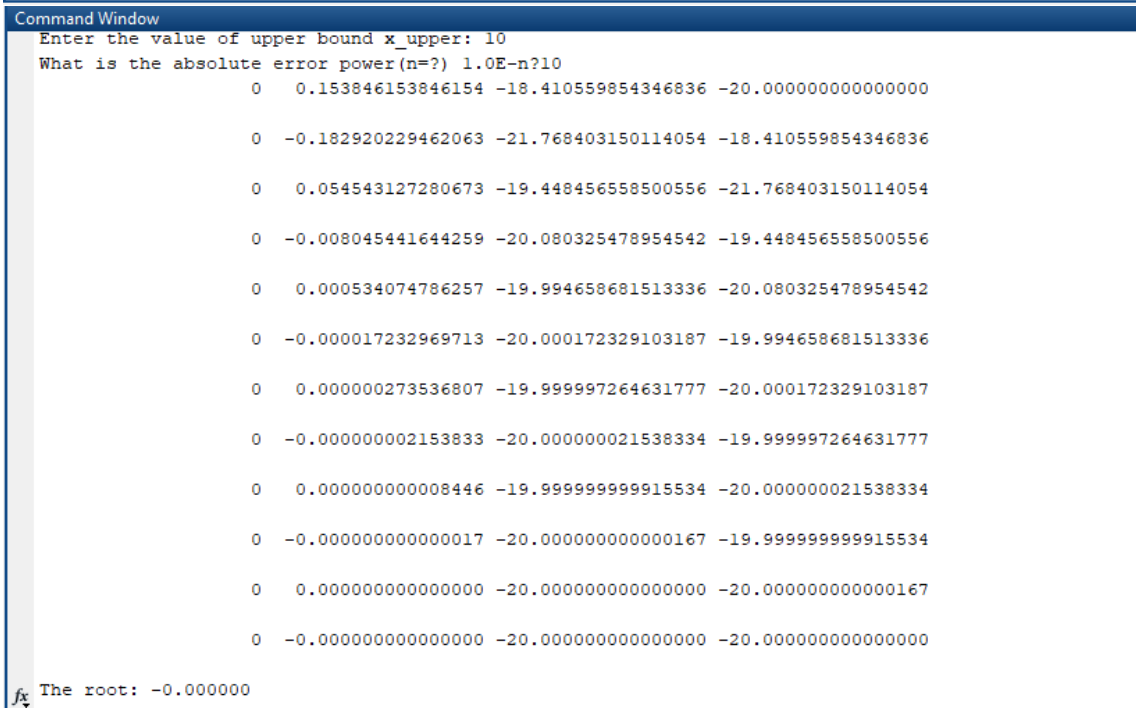
```
   2.243767313019391   1.111111111111111  23.802884677619289  -5.048010973936901

   1.219700862904536   1.111111111111111  -3.013138360492146  23.802884677619289

   1.219700862904536   1.032853000576116  -6.436066933082785  -3.013138360492146

   1.219700862904536   1.276805915973455  -1.889982564186251  -6.436066933082785

   1.219700862904536   1.181857603325908  -3.737045375196402  -1.889982564186251

   1.219700862904536   1.223942429109663  -2.930996887161939  -3.737045375196402

   1.219700862904536   1.219409623002008  -3.018770988542634  -2.930996887161939

   1.219700862904536   1.219710239646200  -3.012956996801051  -3.018770988542634

   1.219700862904536   1.219700714058265  -3.013141239448913  -3.012956996801051

   1.219700862904536   1.219700864076553  -3.013138337823214  -3.013141239448913

   1.219700862904536   1.219700862899940  -3.013138360581046  -3.013138337823214

   1.219700862904536   1.219700862904545  -3.013138360491975  -3.013138360581046

The root: 1.219701
The number of iterations:13
```



```
What is the absolute error power(n=?) 1.0E-n?6
                    0   0.153846153846154 -18.410559854346836 -20.000000000000000

                    0  -0.182920229462063 -21.768403150114054 -18.410559854346836

                    0   0.054543127280673 -19.448456558500556 -21.768403150114054

                    0  -0.008045441644259 -20.080325478954542 -19.448456558500556

                    0   0.000534074786257 -19.994658681513336 -20.080325478954542

                    0  -0.000017232969713 -20.000172329103187 -19.994658681513336

                    0   0.000000273536807 -19.999997264631777 -20.000172329103187

                    0  -0.000000002153833 -20.000000021538334 -19.999997264631777

                    0   0.000000000008446 -19.999999999915534 -20.000000021538334

                    0  -0.000000000000017 -20.000000000000167 -19.999999999915534

                    0   0.000000000000000 -20.000000000000000 -20.000000000000167

                    0  -0.000000000000000 -20.000000000000000 -20.000000000000000

The root: -0.000000
The number of iterations:12
```



```
Enter the value of upper bound x_upper: 10
What is the absolute error power(n=?) 1.0E-n?8
                  0   0.153846153846154 -18.410559854346836 -20.000000000000000

                  0  -0.182920229462063 -21.768403150114054 -18.410559854346836

                  0   0.054543127280673 -19.448456558500556 -21.768403150114054

                  0  -0.008045441644259 -20.080325478954542 -19.448456558500556

                  0   0.000534074786257 -19.994658681513336 -20.080325478954542

                  0  -0.000017232969713 -20.000172329103187 -19.994658681513336

                  0   0.000000273536807 -19.999997264631777 -20.000172329103187

                  0  -0.000000002153833 -20.000000021538334 -19.999997264631777

                  0   0.000000000008446 -19.999999999915534 -20.000000021538334

                  0  -0.000000000000017 -20.000000000000167 -19.999999999915534

                  0   0.000000000000000 -20.000000000000000 -20.000000000000167

                  0  -0.000000000000000 -20.000000000000000 -20.000000000000000

The root: -0.000000
```



```
Enter the value of upper bound x_upper: 10
What is the absolute error power(n=?) 1.0E-n?10
                  0   0.153846153846154 -18.410559854346836 -20.000000000000000

                  0  -0.182920229462063 -21.768403150114054 -18.410559854346836

                  0   0.054543127280673 -19.448456558500556 -21.768403150114054

                  0  -0.008045441644259 -20.080325478954542 -19.448456558500556

                  0   0.000534074786257 -19.994658681513336 -20.080325478954542

                  0  -0.000017232969713 -20.000172329103187 -19.994658681513336

                  0   0.000000273536807 -19.999997264631777 -20.000172329103187

                  0  -0.000000002153833 -20.000000021538334 -19.999997264631777

                  0   0.000000000008446 -19.999999999915534 -20.000000021538334

                  0  -0.000000000000017 -20.000000000000167 -19.999999999915534

                  0   0.000000000000000 -20.000000000000000 -20.000000000000167

                  0  -0.000000000000000 -20.000000000000000 -20.000000000000000

The root: -0.000000
```

Discussion Part on the Modified False Position Method:

This method is useful for much detailed analysis because the modification on the bounds and basically subdividing the interval at each iteration. As we see from the solutions above,

At each iteration the interval changes and at the end prediction and converges occurs.

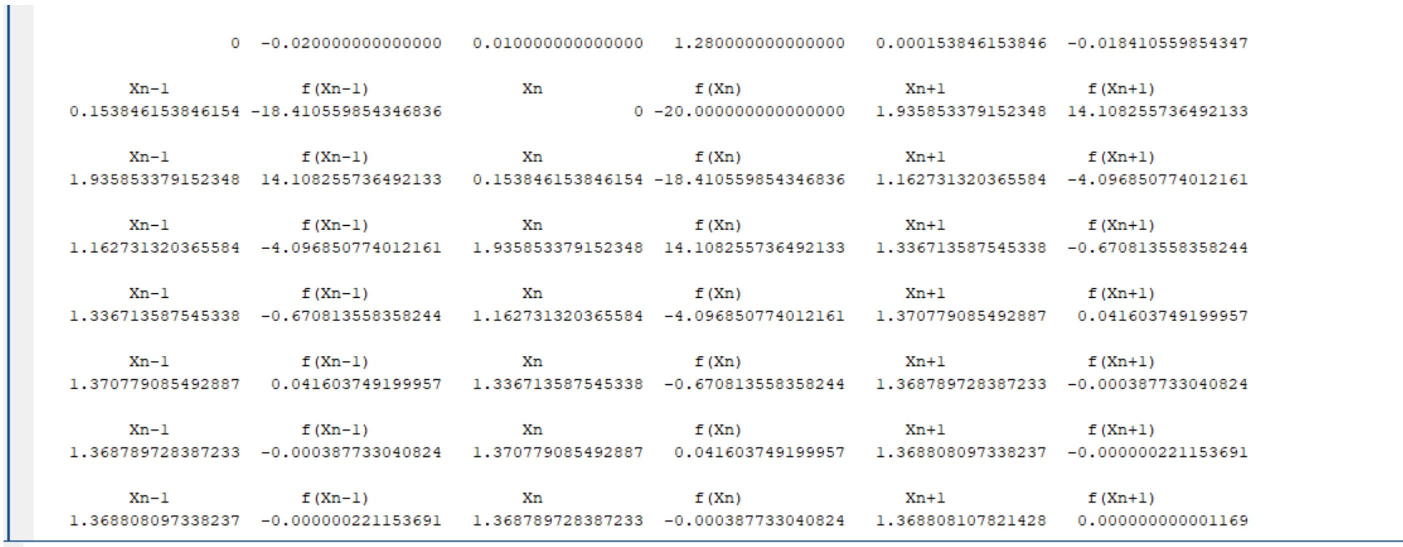
However, the absolute error criteria did not change results too much as we obtained above. Modified False Position is good to detect the boundary which stuck in a value and modify on the interval.

d) Secand Method:

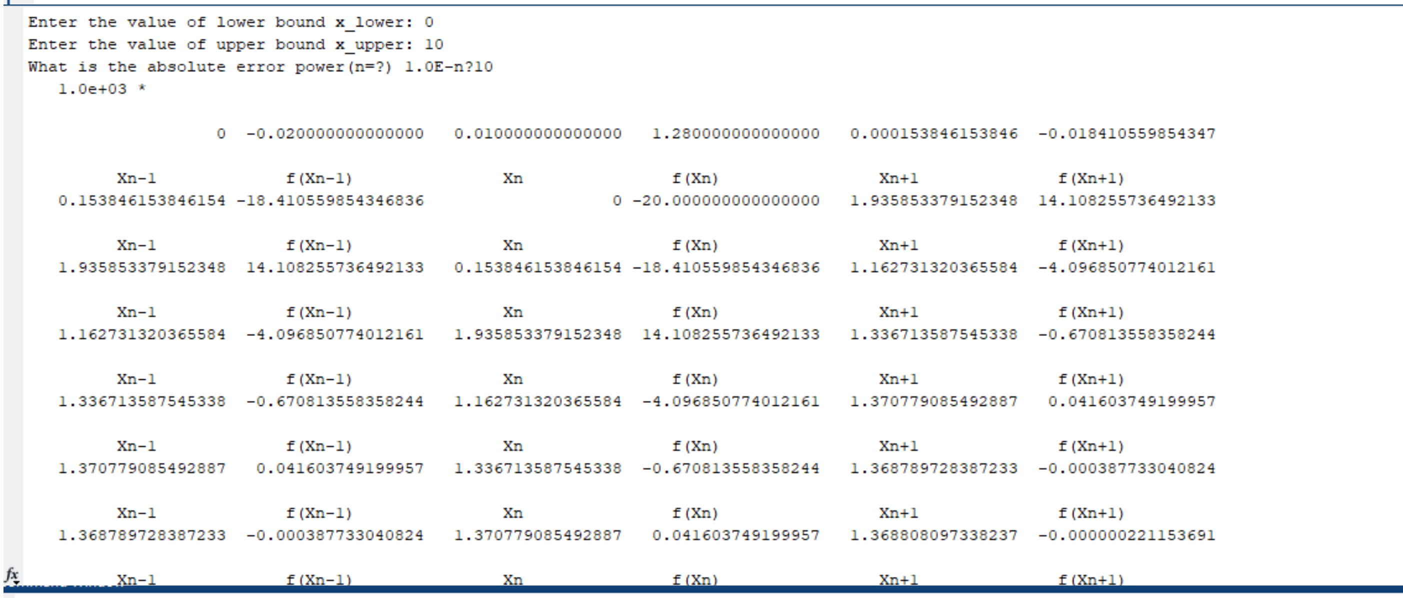
This method is very similar to the false-position method and it also utilizes boundaries to estimate the root of non-linear functions. Bothe Secant and False Position method are the same term-by-term basis. And my code is the following:

```
syms x;
myfunction = input('Enter the function in terms of x:'); %function is taken
by the user

 %To make the changes easier I will took interval and the power of the
absolute error as an
x_lower = input('Enter the value of lower bound x_lower: ');
x_upper = input('Enter the value of upper bound x_upper: ');
n = input('What is the absolute error power(n=?) 1.0E-n?');

%%Inline is used to be able to find roots of the any entered equation in
terms of x
f = inline(myfunction);  % basically myfunction is any function to be
analysed in Secand Method

iter = 0; % I initialized the iteration as 0, to be able to count the num
of iterations
xr = (x_lower*f(x_upper)-x_upper*f(x_lower))/(f(x_upper)-f(x_lower));

disp([x_lower f(x_lower) x_upper f(x_upper) xr f(xr)]);

%%%%% Loop starts

while abs(f(xr)) > 1.0*10.^-n  %Absolute error criteria to stop the loop

    iter = iter + 1;
    x_upper = x_lower;
    x_lower = xr;
    xr = (x_lower*f(x_upper)-x_upper*f(x_lower))/(f(x_upper)-f(x_lower));

    % In each iteration I display to see how values changes
    disp('Xn-1 f(Xn-1) Xn f(Xn) Xn+1 f(Xn+1)');
    disp([x_lower f(x_lower) x_upper f(x_upper) xr f(xr)]);
end

display(['Root is x=' num2str(xr)]);

%%%%%%%%%%%%%%%%%%%%%%%%
```

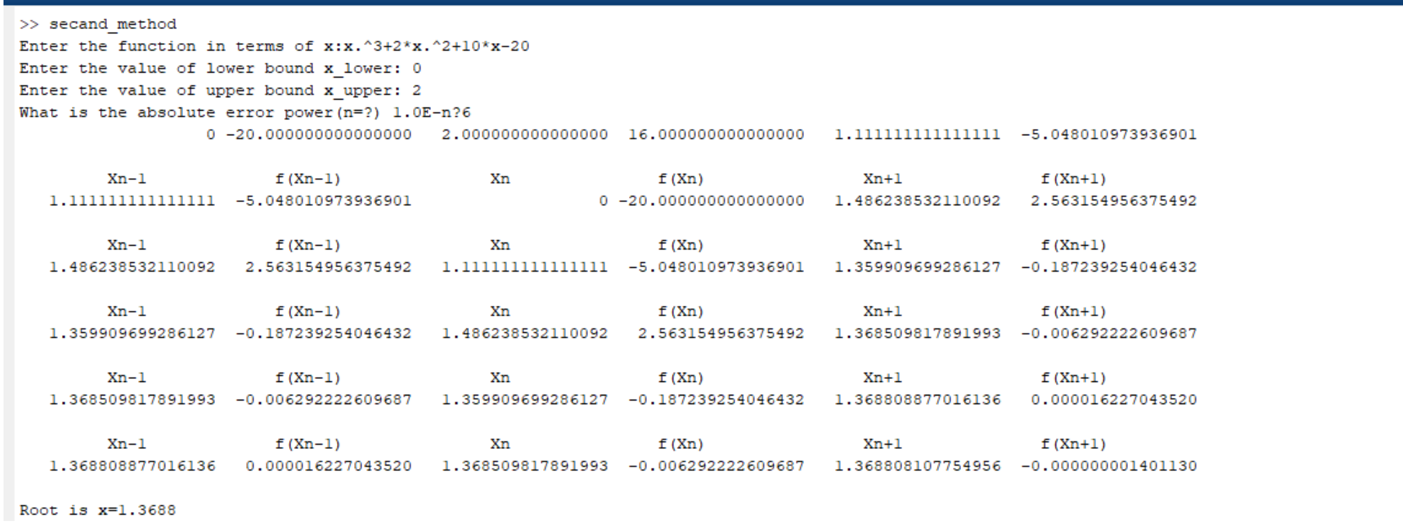
```
Solutions:
```



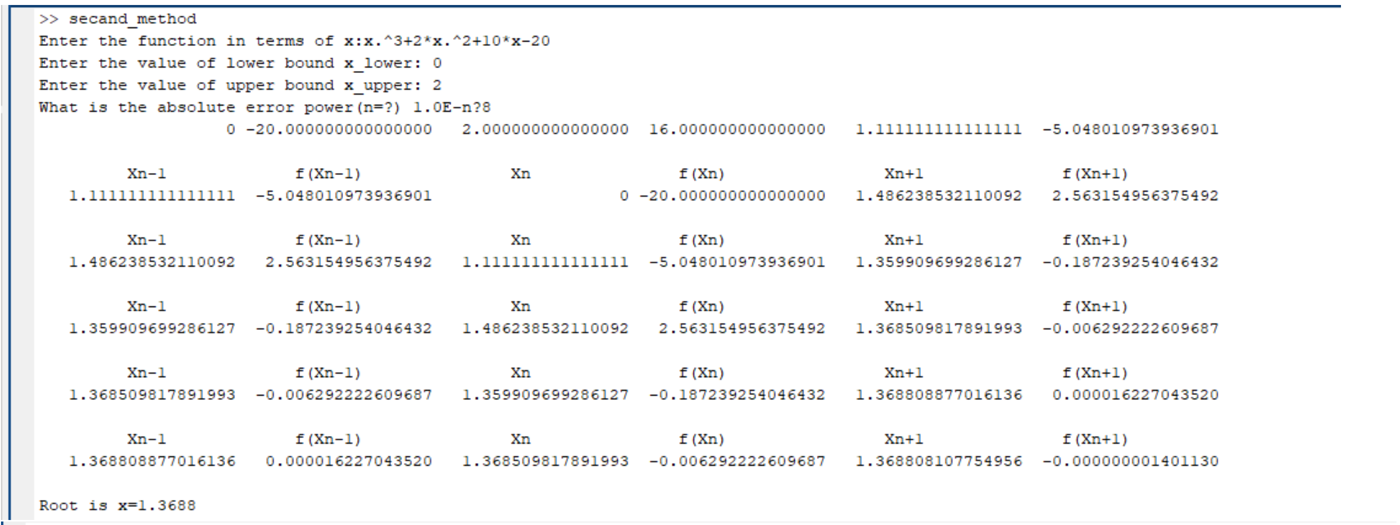
```
                  0  -0.020000000000000   0.010000000000000   1.280000000000000   0.000153846153846  -0.018410559854347

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  0.153846153846154 -18.410559854346836                   0 -20.000000000000000   1.935853379152348  14.108255736492133

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  1.935853379152348  14.108255736492133   0.153846153846154 -18.410559854346836   1.162731320365584  -4.096850774012161

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  1.162731320365584  -4.096850774012161   1.935853379152348  14.108255736492133   1.336713587545338  -0.670813558358244

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  1.336713587545338  -0.670813558358244   1.162731320365584  -4.096850774012161   1.370779085492887   0.041603749199957

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  1.370779085492887   0.041603749199957   1.336713587545338  -0.670813558358244   1.368789728387233  -0.000387733040824

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  1.368789728387233  -0.000387733040824   1.370779085492887   0.041603749199957   1.368808097338237  -0.000000221153691

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  1.368808097338237  -0.000000221153691   1.368789728387233  -0.000387733040824   1.368808107821428   0.000000000001169
```



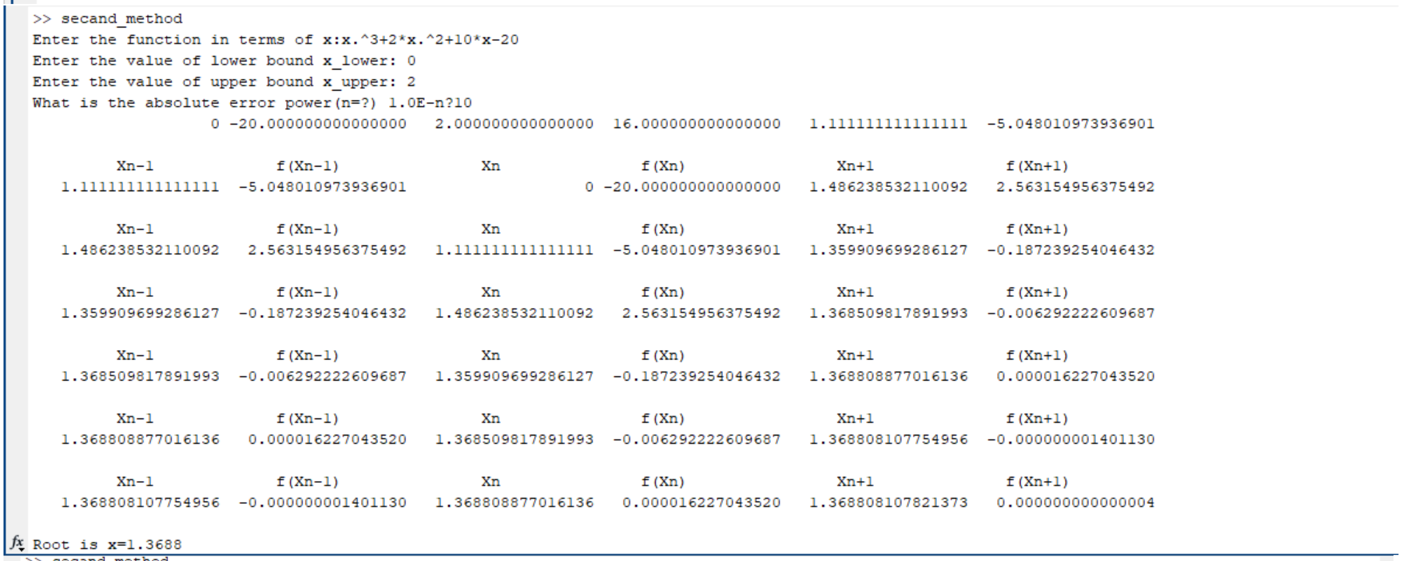
```
 Enter the value of lower bound x_lower: 0
 Enter the value of upper bound x_upper: 10
 What is the absolute error power(n=?) 1.0E-n?10
    1.0e+03 *

                  0  -0.020000000000000   0.010000000000000   1.280000000000000   0.000153846153846  -0.018410559854347

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  0.153846153846154 -18.410559854346836                   0 -20.000000000000000   1.935853379152348  14.108255736492133

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  1.935853379152348  14.108255736492133   0.153846153846154 -18.410559854346836   1.162731320365584  -4.096850774012161

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  1.162731320365584  -4.096850774012161   1.935853379152348  14.108255736492133   1.336713587545338  -0.670813558358244

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  1.336713587545338  -0.670813558358244   1.162731320365584  -4.096850774012161   1.370779085492887   0.041603749199957

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  1.370779085492887   0.041603749199957   1.336713587545338  -0.670813558358244   1.368789728387233  -0.000387733040824

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  1.368789728387233  -0.000387733040824   1.370779085492887   0.041603749199957   1.368808097338237  -0.000000221153691

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
```



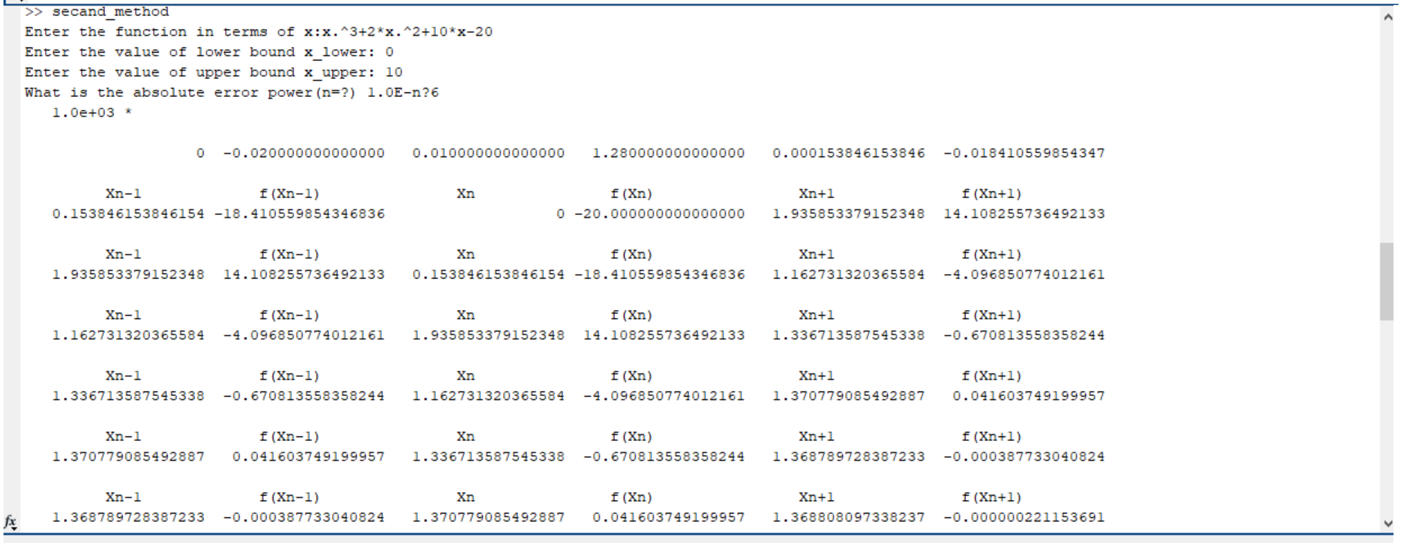
```
>> secand_method
Enter the function in terms of x:x.^3+2*x.^2+10*x-20
Enter the value of lower bound x_lower: 0
Enter the value of upper bound x_upper: 2
What is the absolute error power(n=?) 1.0E-n?6
                  0 -20.000000000000000   2.000000000000000  16.000000000000000   1.111111111111111  -5.048010973936901

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  1.111111111111111  -5.048010973936901                   0 -20.000000000000000   1.486238532110092   2.563154956375492

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  1.486238532110092   2.563154956375492   1.111111111111111  -5.048010973936901   1.359909699286127  -0.187239254046432

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  1.359909699286127  -0.187239254046432   1.486238532110092   2.563154956375492   1.368509817891993  -0.006292222609687

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  1.368509817891993  -0.006292222609687   1.359909699286127  -0.187239254046432   1.368808877016136   0.000016227043520

        Xn-1              f(Xn-1)              Xn               f(Xn)               Xn+1              f(Xn+1)
  1.368808877016136   0.000016227043520   1.368509817891993  -0.006292222609687   1.368808107754956  -0.000000001401130

Root is x=1.3688
```



```
>> secand_method
Enter the function in terms of x:x.^3+2*x.^2+10*x-20
Enter the value of lower bound x_lower: 0
Enter the value of upper bound x_upper: 2
What is the absolute error power(n=?) 1.0E-n?8
                 0 -20.000000000000000   2.000000000000000  16.000000000000000   1.111111111111111  -5.048010973936901

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   1.111111111111111  -5.048010973936901                   0 -20.000000000000000   1.486238532110092   2.563154956375492

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   1.486238532110092   2.563154956375492   1.111111111111111  -5.048010973936901   1.359909699286127  -0.187239254046432

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   1.359909699286127  -0.187239254046432   1.486238532110092   2.563154956375492   1.368509817891993  -0.006292222609687

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   1.368509817891993  -0.006292222609687   1.359909699286127  -0.187239254046432   1.368808877016136   0.000016227043520

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   1.368808877016136   0.000016227043520   1.368509817891993  -0.006292222609687   1.368808107754956  -0.000000001401130

Root is x=1.3688
```



```
>> secand_method
Enter the function in terms of x:x.^3+2*x.^2+10*x-20
Enter the value of lower bound x_lower: 0
Enter the value of upper bound x_upper: 2
What is the absolute error power(n=?) 1.0E-n?10
                 0 -20.000000000000000   2.000000000000000  16.000000000000000   1.111111111111111  -5.048010973936901

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   1.111111111111111  -5.048010973936901                   0 -20.000000000000000   1.486238532110092   2.563154956375492

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   1.486238532110092   2.563154956375492   1.111111111111111  -5.048010973936901   1.359909699286127  -0.187239254046432

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   1.359909699286127  -0.187239254046432   1.486238532110092   2.563154956375492   1.368509817891993  -0.006292222609687

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   1.368509817891993  -0.006292222609687   1.359909699286127  -0.187239254046432   1.368808877016136   0.000016227043520

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   1.368808877016136   0.000016227043520   1.368509817891993  -0.006292222609687   1.368808107754956  -0.000000001401130

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   1.368808107754956  -0.000000001401130   1.368808877016136   0.000016227043520   1.368808107821373   0.000000000000004

Root is x=1.3688
```



```
>> secand_method
Enter the function in terms of x:x.^3+2*x.^2+10*x-20
Enter the value of lower bound x_lower: 0
Enter the value of upper bound x_upper: 10
What is the absolute error power(n=?) 1.0E-n?6
   1.0e+03 *

                 0  -0.020000000000000   0.010000000000000   1.280000000000000   0.000153846153846  -0.018410559854347

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   0.153846153846154 -18.410559854346836                   0 -20.000000000000000   1.935853379152348  14.108255736492133

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   1.935853379152348  14.108255736492133   0.153846153846154 -18.410559854346836   1.162731320365584  -4.096850774012161

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   1.162731320365584  -4.096850774012161   1.935853379152348  14.108255736492133   1.336713587545338  -0.670813558358244

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   1.336713587545338  -0.670813558358244   1.162731320365584  -4.096850774012161   1.370779085492887   0.041603749199957

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   1.370779085492887   0.041603749199957   1.336713587545338  -0.670813558358244   1.368789728387233  -0.000387733040824

         Xn-1              f(Xn-1)              Xn               f(Xn)              Xn+1              f(Xn+1)
   1.368789728387233  -0.000387733040824   1.370779085492887   0.041603749199957   1.368808097338237  -0.000000221153691
```

Discussion Part on the Secand Method:

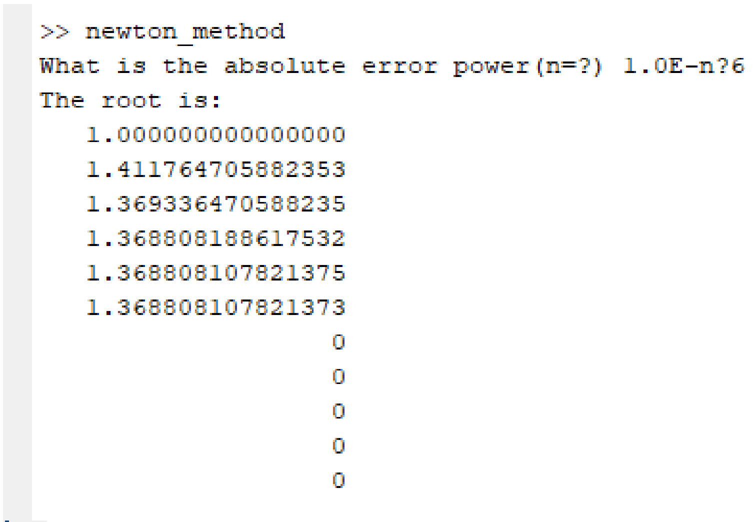
Secand method converges than Newton's method and as we see from above secand method takes more iteration to approximate. However, it is much more close to the exact root than Bisection method and moreover, Secand method claims that the root is around 1.3688.

e) Newton’s Method:

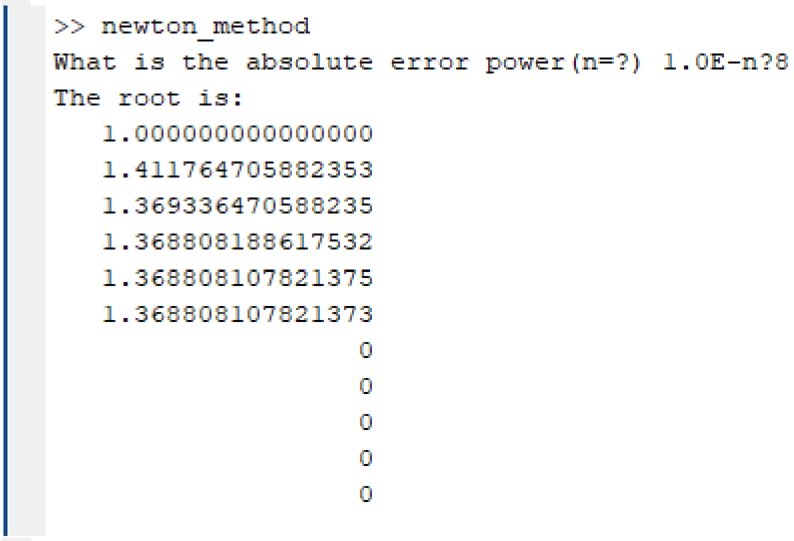
Newton Method has some requirements to be applied.For example, it needs to use derivative of the function so it is important that the function satisfies this availability. However, one of the avantages of the Newton Method is the fast convergence. And my code in Matlab is the following:

```
f = @(x) x.^3+2*x.^2+10*x-20;  %equation definition
f_derivative = @(x)3*x.^2+4*x+10;  %first order derivative of f

% We use Newton's iteration with a starting value in that range to
approximate the root.
x0 = 1;
Nmax = 10; %maximum number of iterations
n = input('What is the absolute error power(n=?) 1.0E-n?');
x = zeros(Nmax +1,1); %each iteration solution will be shown
x(1) = x0;

iter=2;
final_iteration = 1 + Nmax;

%%%%% Loop starts

while (iter <= Nmax + 1)
    fold = f(x(iter -1));
    fnext = f_derivative(x(iter-1));
    x(iter) = x(iter-1) - fold/fnext; %new point is founded

    %Absolute error criteria to stop the loop
    if (abs(fe) <= 1.0*10.^-n)
        final_iteration = iter;
        disp('The root is: ')
        disp(x)
        break;
    end
    iter = iter + 1;
end

%%%%%%%%%%%%%%%%%%%%%%%%
```

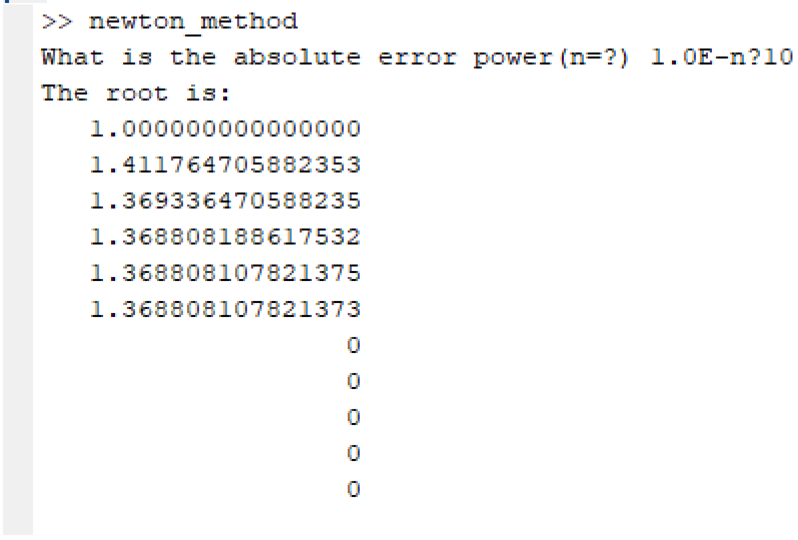
Solutions:



```
>> newton_method
What is the absolute error power(n=?) 1.0E-n?6
The root is:
   1.000000000000000
   1.411764705882353
   1.369336470588235
   1.368808188617532
   1.368808107821375
   1.368808107821373
                   0
                   0
                   0
                   0
                   0
```



```
>> newton_method
What is the absolute error power(n=?) 1.0E-n?8
The root is:
   1.000000000000000
   1.411764705882353
   1.369336470588235
   1.368808188617532
   1.368808107821375
   1.368808107821373
                   0
                   0
                   0
                   0
                   0
```



```
>> newton_method
What is the absolute error power(n=?) 1.0E-n?10
The root is:
   1.000000000000000
   1.411764705882353
   1.369336470588235
   1.368808188617532
   1.368808107821375
   1.368808107821373
                   0
                   0
                   0
                   0
                   0
```

Discussion Part on the Newton’s Method:

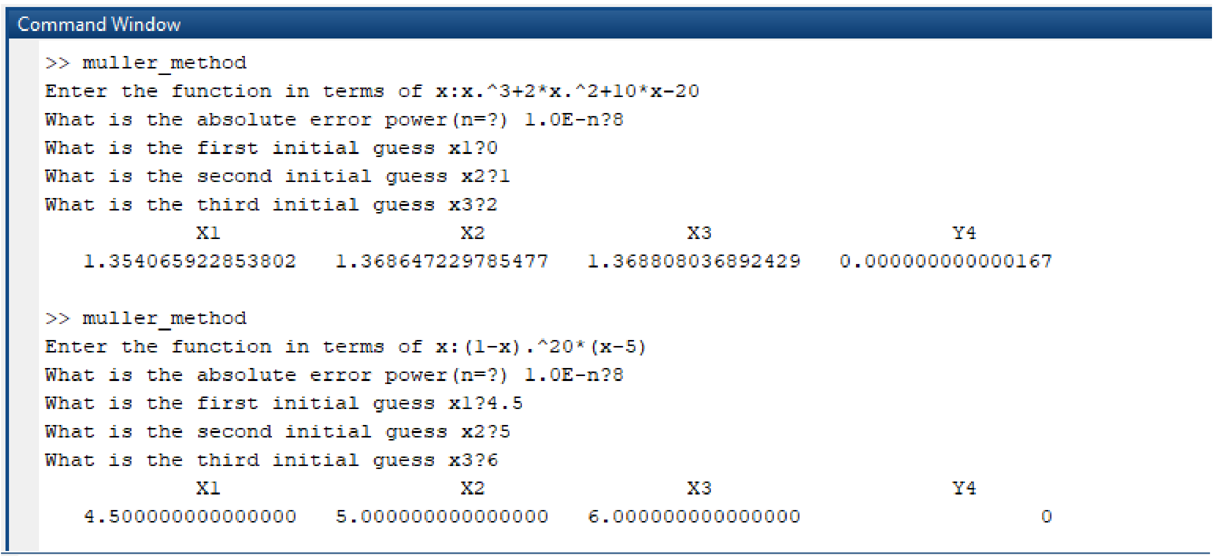
In this method, it uses the slope at the point and finds new point to apply the same way to this point but Newton method is easy to fail. However, as we can see from the iterations above I get the same value with the Secan Method, the root seems like close to 1.3688.

Q2 Answers:

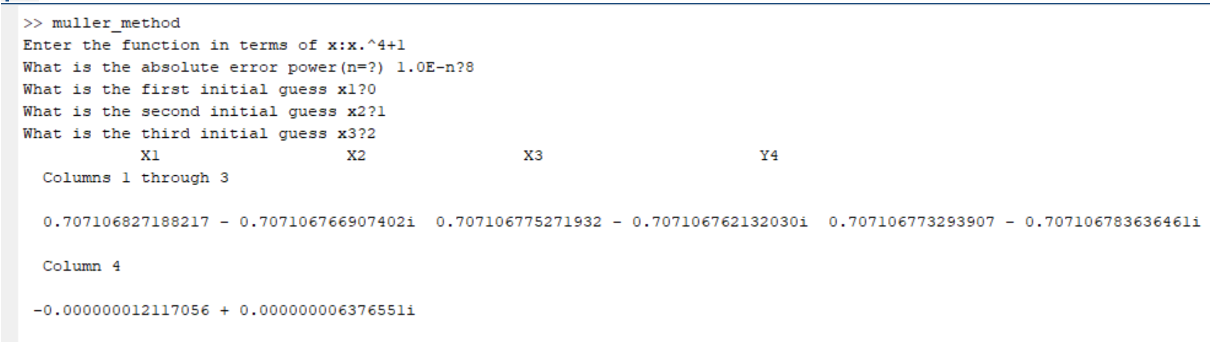
Muller Method:

This is a root finding method that uses three points of initial guesses to construct a parabola and it decides the new point by looking for the intersection of parabola with the x-axis then do the same work again until the tolerance is reached. And my Matlab code is the following for this method:

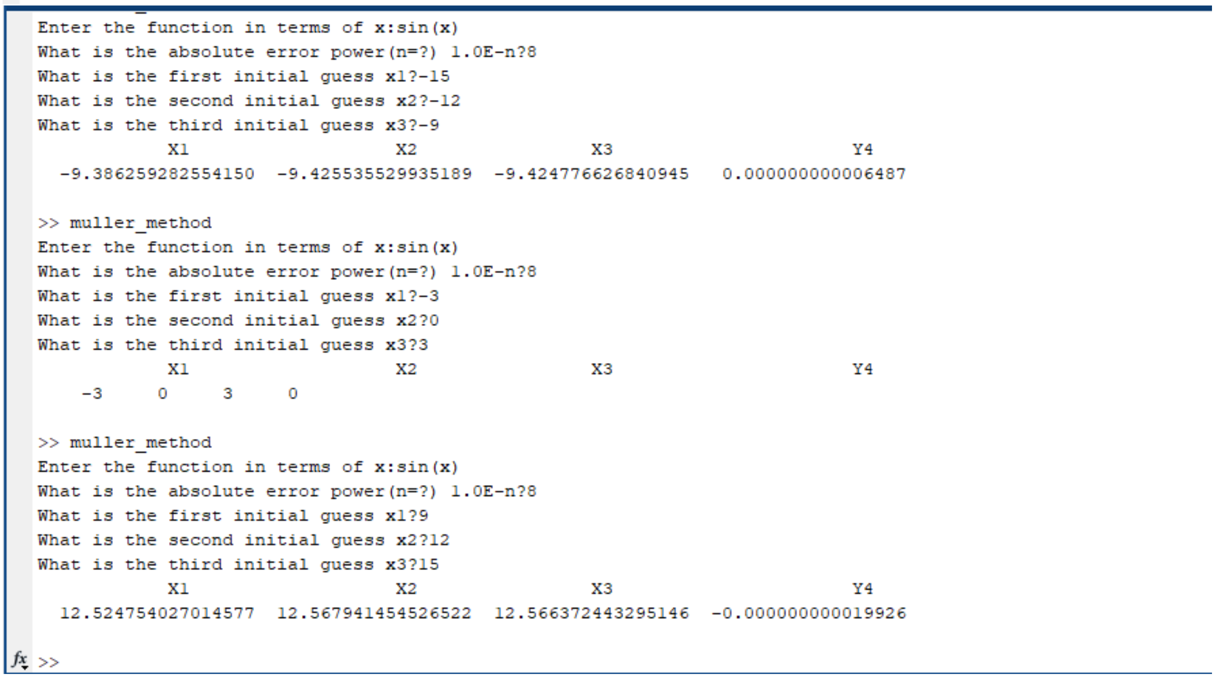
```
syms x;
myfunction = input('Enter the function in terms of x:'); %function is taken
by the user

n = input('What is the absolute error power(n=?) 1.0E-n?');

%%Inline is used to be able to find roots of the any entered equation in
terms of x
f = inline(myfunction);  % basically myfunction is any function to be
analysed in Secand Method

%three approximation for muller method
x1 = input('What is the first initial guess x1?');
x2 = input('What is the second initial guess x2?');
x3 = input('What is the third initial guess x3?');

%evaluates each point on the function: f(x1), f(x2), f(x3)
y1 = f(x1);
y2 = f(x2);
y3 = f(x3);
delta0 = (y2-y1)/(x2-x1);

%Loop starts

while ( true )
   delta1=(y3-y2)/(x3-x2);
    a= (delta1-delta0)/(x3-x1);
    b=delta1 + (x3-x2)*a;
    c=f(x3);
    x4=x3-2*c/(b+sign(b)*sqrt(b*b-4.0*a*c));   %-2c/b(+or-)root of(b^2-4ac)
formula
    y4=f(x4);

     if ( abs( x4 - x3 ) < 1.0*10.^-n  || abs( y4 ) < 1.0*10.^-n  )
%absolute error criteria to break loop
         break;
     end
     x1=x2;
     x2=x3;
     x3=x4;
     y1=y2;
     y2=y3;
     y3=y4;
     delta0=delta1;
end

disp('            X1                    X2                X3
Y4');
disp([x1 x2 x3 y4])

%%%%%%%%%%%%%%%%%%%%%%


Solutions:
```



```
>> muller_method
Enter the function in terms of x:x.^3+2*x.^2+10*x-20
What is the absolute error power(n=?) 1.0E-n?8
What is the first initial guess x1?0
What is the second initial guess x2?1
What is the third initial guess x3?2
            X1                    X2                  X3                     Y4
   1.354065922853802   1.368647229785477   1.368808036892429   0.000000000000167

>> muller_method
Enter the function in terms of x:(1-x).^20*(x-5)
What is the absolute error power(n=?) 1.0E-n?8
What is the first initial guess x1?4.5
What is the second initial guess x2?5
What is the third initial guess x3?6
            X1                    X2                  X3                     Y4
   4.500000000000000   5.000000000000000   6.000000000000000                   0
```



```
>> muller_method
Enter the function in terms of x:x.^4+1
What is the absolute error power(n=?) 1.0E-n?8
What is the first initial guess x1?0
What is the second initial guess x2?1
What is the third initial guess x3?2
            X1                    X2                  X3                     Y4
  Columns 1 through 3

  0.707106827188217 - 0.707106766907402i  0.707106775271932 - 0.707106762132030i  0.707106773293907 - 0.707106783636461i

  Column 4

 -0.000000012117056 + 0.000000006376551i
```



```
Enter the function in terms of x:sin(x)
What is the absolute error power(n=?) 1.0E-n?8
What is the first initial guess x1?-15
What is the second initial guess x2?-12
What is the third initial guess x3?-9
            X1                    X2                  X3                     Y4
  -9.386259282554150  -9.425535529935189  -9.424776626840945   0.000000000006487

>> muller_method
Enter the function in terms of x:sin(x)
What is the absolute error power(n=?) 1.0E-n?8
What is the first initial guess x1?-3
What is the second initial guess x2?0
What is the third initial guess x3?3
            X1                    X2                  X3                     Y4
    -3      0      3      0

>> muller_method
Enter the function in terms of x:sin(x)
What is the absolute error power(n=?) 1.0E-n?8
What is the first initial guess x1?9
What is the second initial guess x2?12
What is the third initial guess x3?15
            X1                    X2                  X3                     Y4
  12.524754027014577  12.567941454526522  12.566372443295146  -0.000000000019926

>>
```

Discussion Part on the Muller Method:

In this part, I wrote a program for all functions and I took the function as an input from the user in terms of x. In such form:

a) x.^3+2*x.^2+10*x-20
b) (1-x).^20*(x-5)

c) x.^4+1
d) sin(x)

And, I also took three points which are going to be used as initial guesses as an input from the user, so that it can works for the different functions. As we considered the solutions above, a) is the same equation I tried in different methods in the first part of the project and I obtained the same value 1.3688.

For other functions, some of them could not find real roots but we obtained complex roots in the given intervals.

As conclusion, this code works for different type of functions both linear, non-linear.

Muller Method is different from the some of methods which I used in the first part. For instance, Bisection method uses two point for estimations. Even Muller Method based on Secand Method, it is different again in terms of the number of points to be used in the estimation of roots. So that in general, all methods have different properties and the behavior of those methods changes on the functions. Therefore, the selection of interval, function and method types are all related with each other and before come up with a solution on the root we should try different methods on our function.