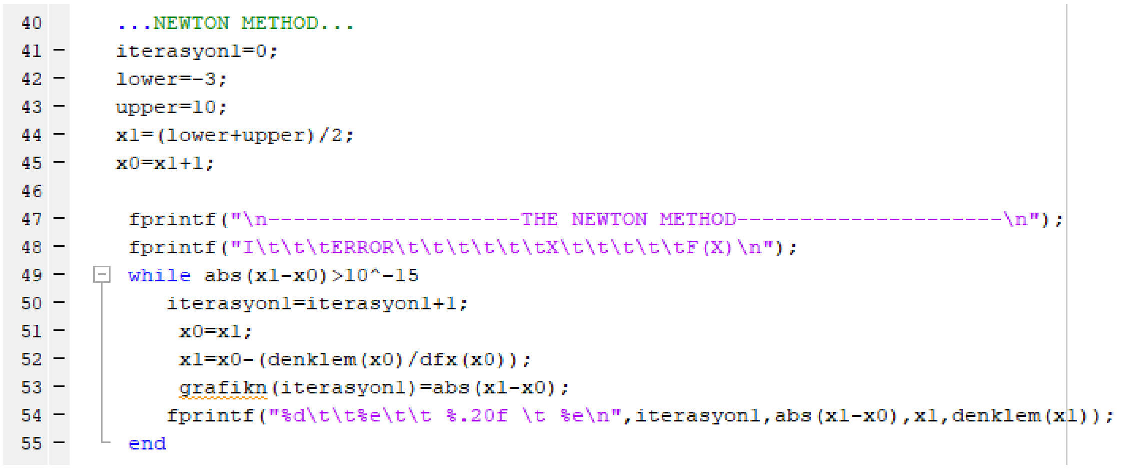
Prepared by
Muhammet Asim Uyanik

We tried to find the roots of an equation given in this project by Newton method, Bisection method and Secant method, and also plotted the roots, errors and results we found with these methods using various functions.

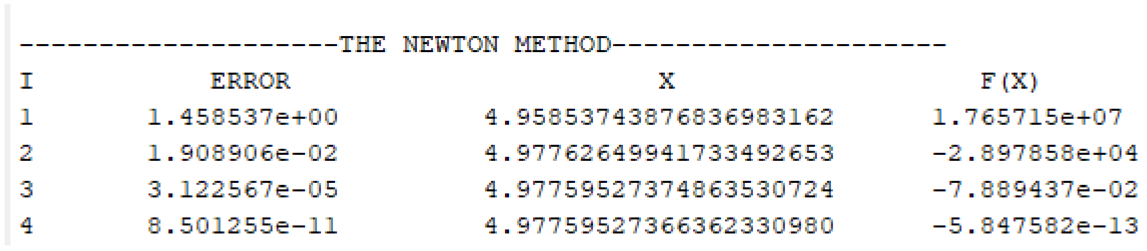
The first of these methods is the Newton Raphson method. Suppose we choose any arbitrary value of x_i. What we have to do is calculate the value f (x_i) corresponding to this point. A tangent to the function is drawn from the point (x_i, f (x_i)) after this point. The point where the drawn tangent intersects the x axis is taken as the new x_{i+1} value and the same operation is applied for x_{i+1}. To explain this method from the codes on matlab in figure 1;

First, an upper and a lower value are determined, and we arbitrarily choose the midpoint of these two values. Then, in line 49, we put it in a loop that will stop if the error value (x_1-x_0) is less than the given tolerance value (10^{-10}). As can be seen in line 52, we defined the formula of Newton's method and set the value of x_1 to the value of x_2. At the end of the loop, we approached zero 10^{-13}as shown in Figure 1.2.



```
40        ...NEWTON METHOD...
41 -      iterasyonl=0;
42 -      lower=-3;
43 -      upper=10;
44 -      xl=(lower+upper)/2;
45 -      x0=xl+1;
46
47 -       fprintf("\n--------------------THE NEWTON METHOD---------------------\n");
48 -       fprintf("I\t\t\tERROR\t\t\t\t\t\tX\t\t\t\t\tF(X)\n");
49 -       while abs(xl-x0)>10^-15
50 -          iterasyonl=iterasyonl+1;
51 -           x0=xl;
52 -           xl=x0-(denklem(x0)/dfx(x0));
53 -           grafikn(iterasyonl)=abs(xl-x0);
54 -          fprintf("%d\t\t%e\t\t %.20f \t %e\n",iterasyonl,abs(xl-x0),xl,denklem(xl));
55 -       end
```

(Figure 1)



```
--------------------THE NEWTON METHOD---------------------
I           ERROR                       X                      F(X)
1        1.458537e+00        4.95853743876836983162       1.765715e+07
2        1.908906e-02        4.97762649941733492653      -2.897858e+04
3        3.122567e-05        4.97759527374863530724      -7.889437e-02
4        8.501255e-11        4.97759527366362330980      -5.847582e-13
```

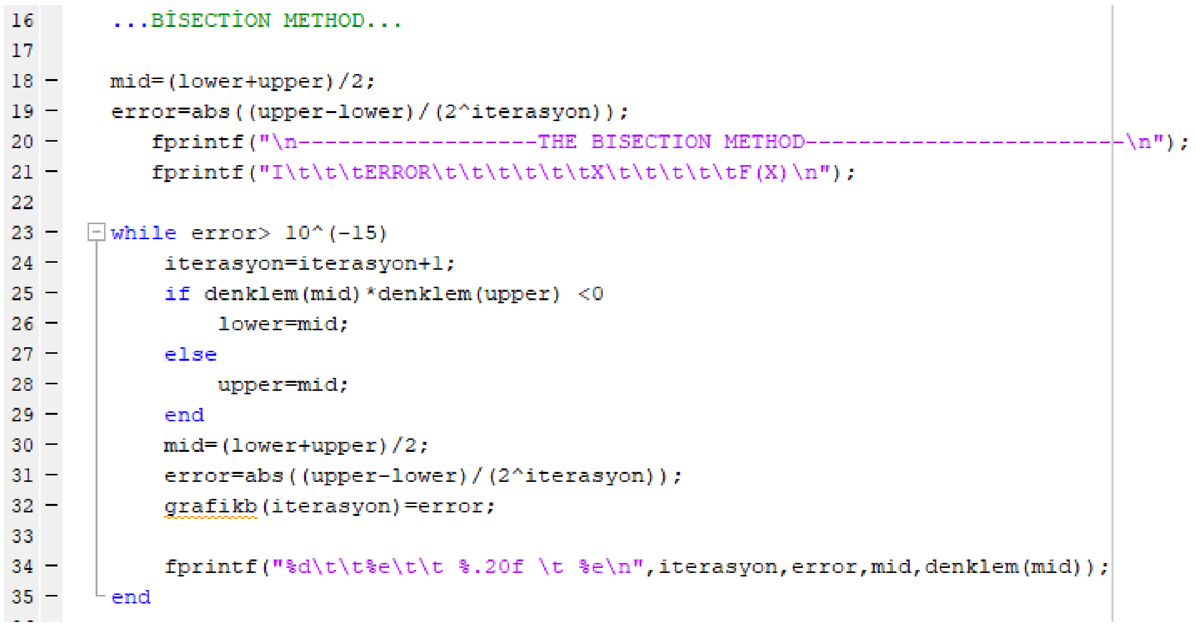
(Figure 1.2)

The other method is the Bisection method.

Let us take a function f (x). This function is continuous in the closed interval [a, b] and the numbers a and b are elements of the set of real numbers. If f (a) .f (b) <0 in this interval, this interval has at least one root of this function. After ensuring that the root exists, all that needs to be done is to satisfy the condition f (a) .f (b) <0 by continuing to divide the gap in half.

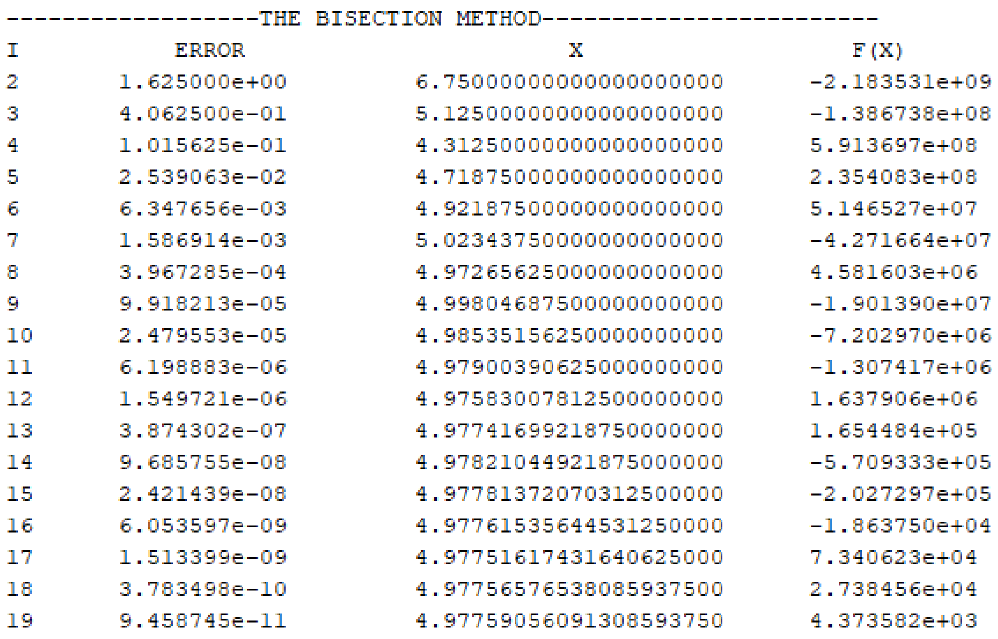
If we explain it on the matlab in Figure 2;

As seen in line 18, we assign the mid value to the middle of the lower and upper limit. Then we define the error value in line 19 and put it in a loop that will stop if the error value is less than the given tolerance value. Since our aim is to catch this situation constantly f (a) .f (b) <0, we made the checks in the 25. and 27. lines and determined our new lower and upper limits.At the end of the loop, we were able to approach zero by 10^3 as shown in Figure 2.2.



```
...BİSECTİON METHOD...

mid=(lower+upper)/2;
error=abs((upper-lower)/(2^iterasyon));
   fprintf("\n------------------THE BISECTION METHOD------------------------\n");
   fprintf("I\t\t\tERROR\t\t\t\t\t\tX\t\t\t\t\tF(X)\n");

while error> 10^(-15)
    iterasyon=iterasyon+1;
    if denklem(mid)*denklem(upper) <0
        lower=mid;
    else
        upper=mid;
    end
    mid=(lower+upper)/2;
    error=abs((upper-lower)/(2^iterasyon));
    grafikb(iterasyon)=error;

    fprintf("%d\t\t%e\t\t %.20f \t %e\n",iterasyon,error,mid,denklem(mid));
end
```

(Figure 2)



```
------------------THE BISECTION METHOD------------------------
I            ERROR                      X                          F(X)
2        1.625000e+00          6.75000000000000000000          -2.183531e+09
3        4.062500e-01          5.12500000000000000000          -1.386738e+08
4        1.015625e-01          4.31250000000000000000          5.913697e+08
5        2.539063e-02          4.71875000000000000000          2.354083e+08
6        6.347656e-03          4.92187500000000000000          5.146527e+07
7        1.586914e-03          5.02343750000000000000          -4.271664e+07
8        3.967285e-04          4.97265625000000000000          4.581603e+06
9        9.918213e-05          4.99804687500000000000          -1.901390e+07
10       2.479553e-05          4.98535156250000000000          -7.202970e+06
11       6.198883e-06          4.97900390625000000000          -1.307417e+06
12       1.549721e-06          4.97583007812500000000          1.637906e+06
13       3.874302e-07          4.97741699218750000000          1.654484e+05
14       9.685755e-08          4.97821044921875000000          -5.709333e+05
15       2.421439e-08          4.97781372070312500000          -2.027297e+05
16       6.053597e-09          4.97761535644531250000          -1.863750e+04
17       1.513399e-09          4.97751617431640625000          7.340623e+04
18       3.783498e-10          4.97756576538085937500          2.738456e+04
19       9.458745e-11          4.97759056091308593750          4.373582e+03
```

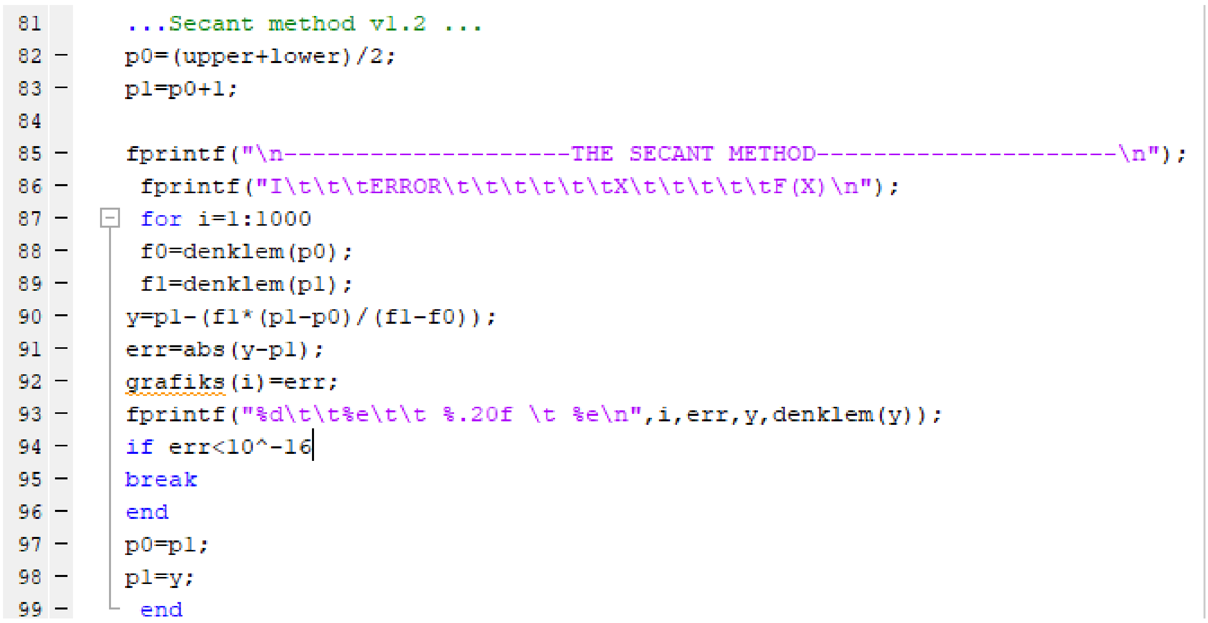
(Figure 2.2)

Our last method is the Secant method.
Although Newton's method is a very powerful root finding technique, the requirement to control the derivative value of the function f in each iteration is a emerges as a difficulty. By

eliminating this problem, Newton It is possible to obtain a weaker method than method.

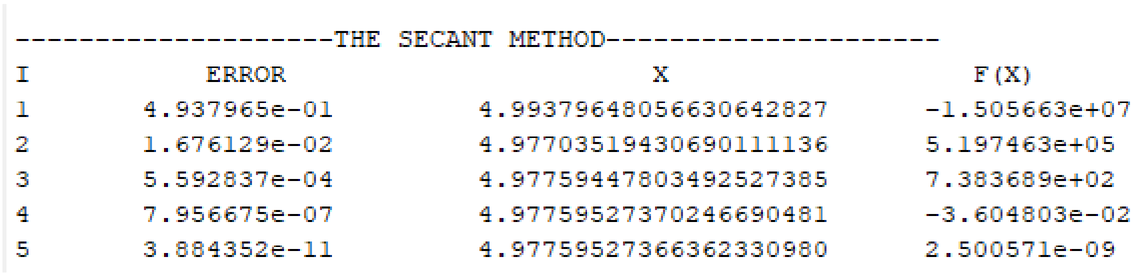
If we explain over the matlab codes shown in Figure 3

Unlike Newton's method, we specified 2 initial values in lines 82 and 83. Then we put it in a loop that will be repeated 1000 times and will end if the error value is less than the tolerance value. We wrote the general formula of the Secant method in line 90 and defined the error formula in line 91. At the end of the loop we were able to get close to zero 10^{-9} as shown in Figure 3.1.



```
81        ...Secant method v1.2 ...
82 -      p0=(upper+lower)/2;
83 -      p1=p0+1;
84
85 -      fprintf("\n--------------------THE SECANT METHOD---------------------\n");
86 -       fprintf("I\t\t\tERROR\t\t\t\t\t\tX\t\t\t\t\tF(X)\n");
87 -       for i=1:1000
88 -       f0=denklem(p0);
89 -       f1=denklem(p1);
90 -      y=p1-(f1*(p1-p0)/(f1-f0));
91 -      err=abs(y-p1);
92 -      grafiks(i)=err;
93 -      fprintf("%d\t\t%e\t\t %.20f \t %e\n",i,err,y,denklem(y));
94 -      if err<10^-16
95 -      break
96 -      end
97 -      p0=p1;
98 -      p1=y;
99 -       end
```

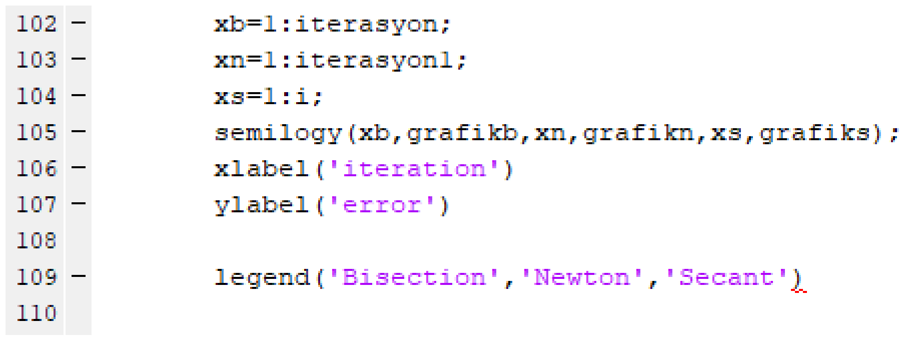
(Figure 3)



```
--------------------THE SECANT METHOD---------------------
I            ERROR                    X                         F(X)
1        4.937965e-01        4.99379648056630642827       -1.505663e+07
2        1.676129e-02        4.97703519430690111136       5.197463e+05
3        5.592837e-04        4.97759447803492527385       7.383689e+02
4        7.956675e-07        4.97759527370246690481       -3.604803e-02
5        3.884352e-11        4.97759527366362330980       2.500571e-09
```

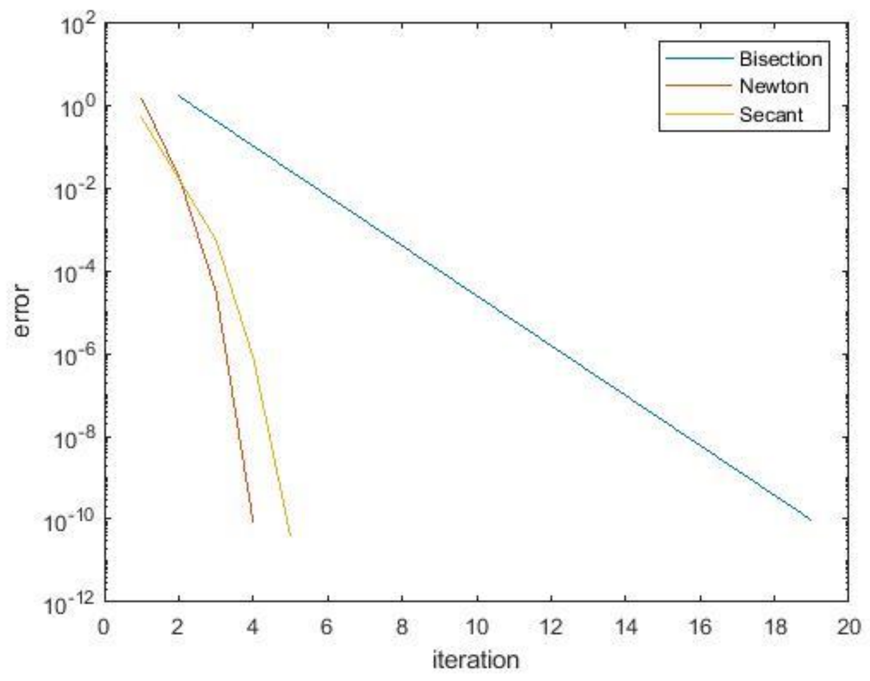
(Figure 3.2)

In Figure 4, we plotted the errors and iterations we made using the semiology function. As shown in the graph in Figure 4.1. the Newton Raphson method was the fastest and most accurate of these methods we used. Because we got close to zero 10^{-13} in just 5 iterations. The second most accurate was the secant method. In this method, we got close to zero 10^{-9}. And the slowest and most incorrect method was the Bisection method.



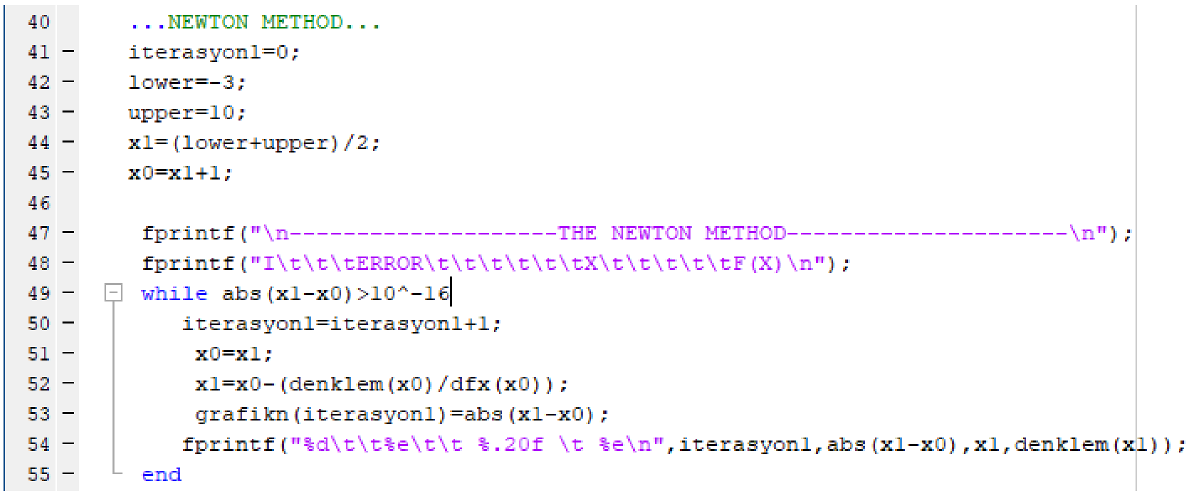
```
        xb=1:iterasyon;
        xn=1:iterasyon1;
        xs=1:i;
        semilogy(xb,grafikb,xn,grafikn,xs,grafiks);
        xlabel('iteration')
        ylabel('error')

        legend('Bisection','Newton','Secant')
```

(Figure 4)



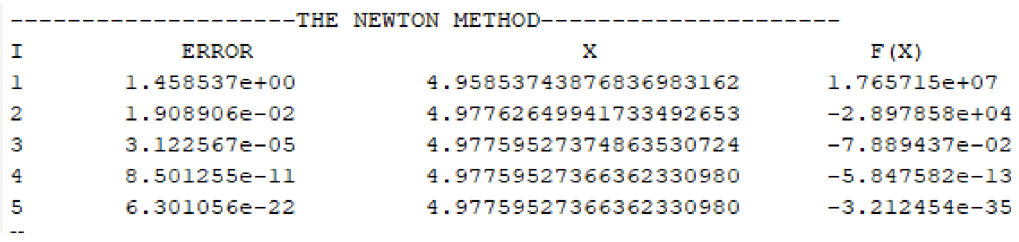
(Figure 4.2)

As seen in Figure 5, we made the tolerance value of 10^{-16} in the newton method. Since we reduced the tolerance value, our error value also decreased and we got closer to zero as shown in figure 5.1.



```
...NEWTON METHOD...
iterasyon1=0;
lower=-3;
upper=10;
x1=(lower+upper)/2;
x0=x1+1;

 fprintf("\n---------------------THE NEWTON METHOD---------------------\n");
 fprintf("I\t\t\tERROR\t\t\t\t\t\tX\t\t\t\t\tF(X)\n");
 while abs(x1-x0)>10^-16
    iterasyon1=iterasyon1+1;
     x0=x1;
     x1=x0-(denklem(x0)/dfx(x0));
     grafikn(iterasyon1)=abs(x1-x0);
    fprintf("%d\t\t%e\t\t %.20f \t %e\n",iterasyon1,abs(x1-x0),x1,denklem(x1));
 end
```

(Figure 5)



```
---------------------THE NEWTON METHOD---------------------
I           ERROR                  X                       F(X)
1        1.458537e+00          4.95853743876836983162      1.765715e+07
2        1.908906e-02          4.97762649941733492653      -2.897858e+04
3        3.122567e-05          4.97759527374863530724      -7.889437e-02
4        8.501255e-11          4.97759527366362330980      -5.847582e-13
5        6.301056e-22          4.97759527366362330980      -3.212454e-35
--
```

(Figure 5.1)

CODES

```
clear;
clear all;

e0=((1/(36*pi))*10^-9);...Define epsilon
iterasyon=1;
lower=-3;
upper=10;
...define equation
syms 'x'
denklem(x)=(1/(4*pi*e0))*((13*(x+7)/(abs(x+7))^3)+(9*(x+4)/(abs(x+4))^3)+(6
*(x-11)/(abs(x-11))^3)+(3*(x-14)/(abs(x-14))^3));
```

```
...define derivative
dfx(x)=81000000000/abs(x + 4)^3 + 117000000000/abs(x + 7)^3 +
54000000000/abs(x - 11)^3 + 27000000000/abs(x - 14)^3 - (27000000000*sign(x
+ 4)*(9*x + 36))/abs(x + 4)^4 - (27000000000*sign(x - 14)*(3*x - 42))/abs(x
- 14)^4 - (27000000000*sign(x - 11)*(6*x - 66))/abs(x - 11)^4 -
(27000000000*sign(x + 7)*(13*x + 91))/abs(x + 7)^4;

......................................................................

...BİSECTİON METHOD...

mid=(lower+upper)/2;
error=abs((upper-lower)/(2^iterasyon));
   fprintf("\n------------------THE BISECTION METHOD-----------------------
-\n");
   fprintf("I\t\t\tERROR\t\t\t\t\t\tX\t\t\t\t\tF(X)\n");

while error> 10^(-10)
    iterasyon=iterasyon+1;
    if denklem(mid)*denklem(upper) <0
        lower=mid;
    else
        upper=mid;
    end
    mid=(lower+upper)/2;
    error=abs((upper-lower)/(2^iterasyon));
    grafikb(iterasyon)=error;

    fprintf("%d\t\t%e\t\t %.20f \t %e\n",iterasyon,error,mid,denklem(mid));
end


......................................................................

...NEWTON METHOD...
iterasyon1=0;
lower=-3;
upper=10;
x1=(lower+upper)/2;
x0=x1+1;

 fprintf("\n--------------------THE NEWTON METHOD---------------------\n");
 fprintf("I\t\t\tERROR\t\t\t\t\t\tX\t\t\t\t\tF(X)\n");
 while abs(x1-x0)>10^-10
    iterasyon1=iterasyon1+1;
     x0=x1;
     x1=x0-(denklem(x0)/dfx(x0));
     grafikn(iterasyon1)=abs(x1-x0);
    fprintf("%d\t\t%e\t\t %.20f \t %e\n",iterasyon1,abs(x1-
x0),x1,denklem(x1));
 end

.....................................................................

% %  ...SECANT METHOD
%
% lower=-3;
% upper=10;
% p0=(upper+lower)/2;
% p1=p0+1;
% p=0;
% while abs(p-p1)< 10^-10
%   f0=denklem(p0);
%   f1=denklem(p1);
%     p=p1-((f1*(p1-p0))/(f1-f0));
%     p0=p1;
%     p1=p;
% end
%     fprintf("The root is with The secant methot");
%     format long
%     double(p)
%     double(denklem(p))

.......................................................................


...Secant method v1.2 ...
p0=(upper+lower)/2;
p1=p0+1;

fprintf("\n--------------------THE SECANT METHOD---------------------\n");
 fprintf("I\t\t\tERROR\t\t\t\t\t\tX\t\t\t\t\tF(X)\n");
 for i=1:1000
 f0=denklem(p0);
 f1=denklem(p1);
y=p1-(f1*(p1-p0)/(f1-f0));
err=abs(y-p1);
grafiks(i)=err;
fprintf("%d\t\t%e\t\t %.20f \t %e\n",i,err,y,denklem(y));
if err<10^-10
break
end
p0=p1;
p1=y;
 end


    xb=1:iterasyon;
    xn=1:iterasyon1;
    xs=1:i;
    semilogy(xb,grafikb,xn,grafikn,xs,grafiks);
    xlabel('iteration')
    ylabel('error')

    legend('Bisection','Newton','Secant')
```