Scientific Computing Using Matlab

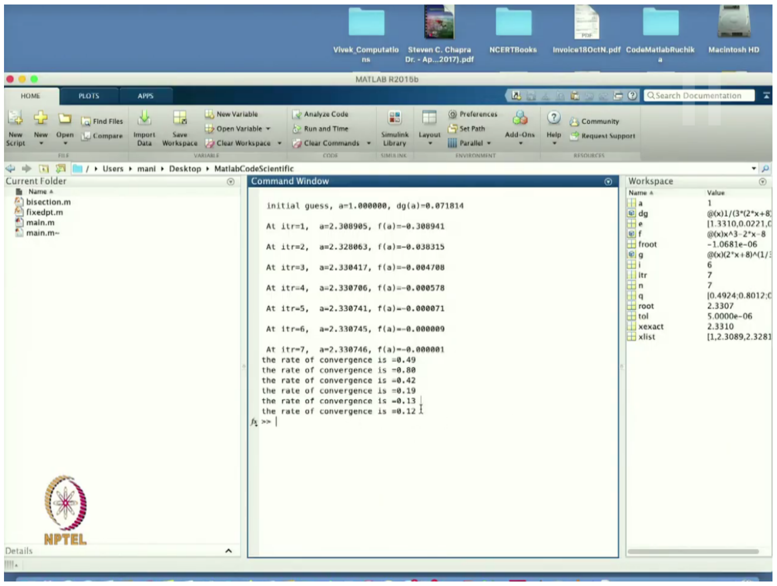
Professor Vivek Aggarwal & Mani Mehra
Department of Mathematics
Indian Institute of Technology, Delhi/DTU

Lecture 16

Matlab Code For Newton-Raphson and Regula-Falsi Method

Hello viewers, welcome back to the course on scientific computing using MATLAB. So, today we are going to start lecture 16. So in the previous lecture we have written the MATLAB code for a fixed point method. So today we will start with the MATLAB code of the Newton-Raphson method. So let us write this one.

(Refer Slide Time: 00:41)

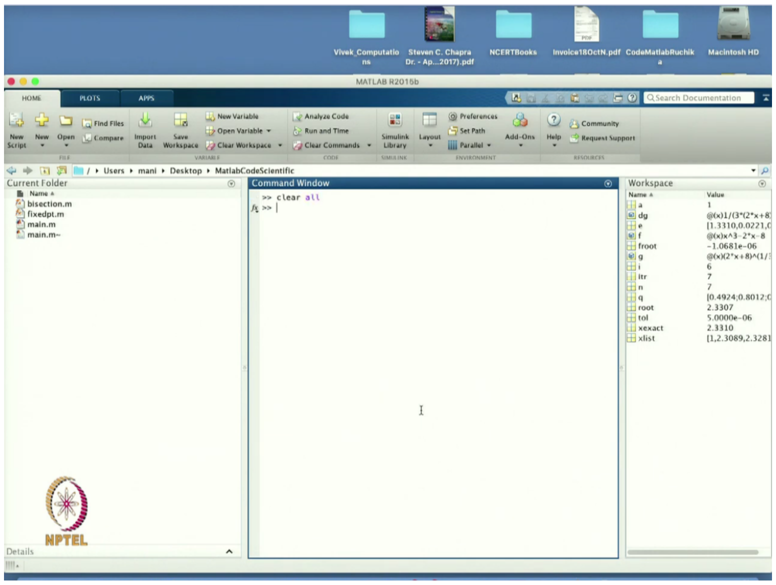


```
 initial guess, a=1.000000, dg(a)=0.071814

 At itr=1,  a=2.308905, f(a)=-0.308941

 At itr=2,  a=2.328063, f(a)=-0.038315

 At itr=3,  a=2.330417, f(a)=-0.004708

 At itr=4,  a=2.330706, f(a)=-0.000578

 At itr=5,  a=2.330741, f(a)=-0.000071

 At itr=6,  a=2.330745, f(a)=-0.000009

 At itr=7,  a=2.330746, f(a)=-0.000001
the rate of convergence is =0.49
the rate of convergence is =0.80
the rate of convergence is =0.42
the rate of convergence is =0.19
the rate of convergence is =0.13
the rate of convergence is =0.12
>>
```

Name	Value
a	1
dg	@(x)1/(3*(2*x+8)
e	[1.3310,0.0221,0
f	@(x)x^3-2*x-8
froot	-1.0681e-06
g	@(x)(2*x+8)^(1/
i	6
itr	7
n	7
q	[0.4924;0.8012;0
root	2.3307
tol	5.0000e-06
xexact	2.3310
xlist	[1,2.3089,2.3281

So in the previous class, we have shown that the rate of convergence was 0.2.

(Refer Slide Time: 00:47)

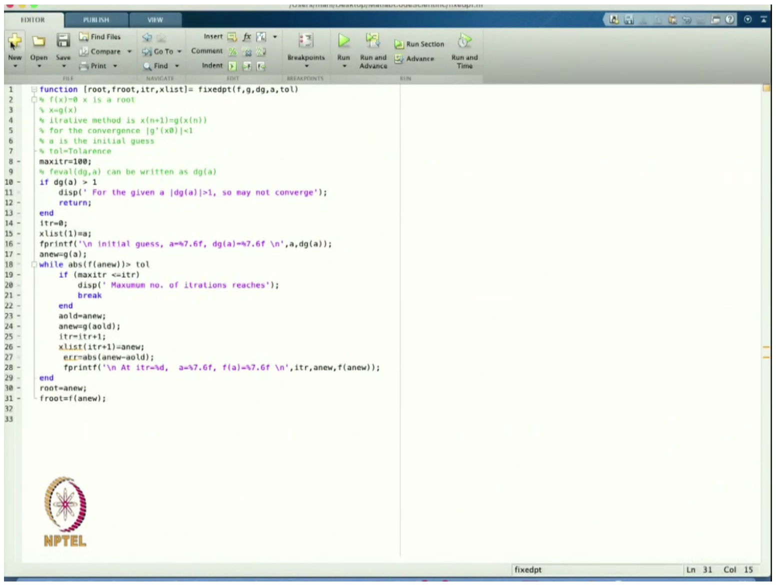


```
>> clear all
>>
```

Name	Value
a	1
dg	@(x)1/(3*(2*x+8)
e	[1.3310,0.0221,0
f	@(x)x^3-2*x-8
froot	-1.0681e-06
g	@(x)(2*x+8)^(1/3
i	6
itr	7
n	7
q	[0.4924;0.8012;0
root	2.3307
tol	5.0000e-06
xexact	2.3310
xlist	[1,2.3089,2.3281

So I will just write the CLC and clear all. Now based on this one.

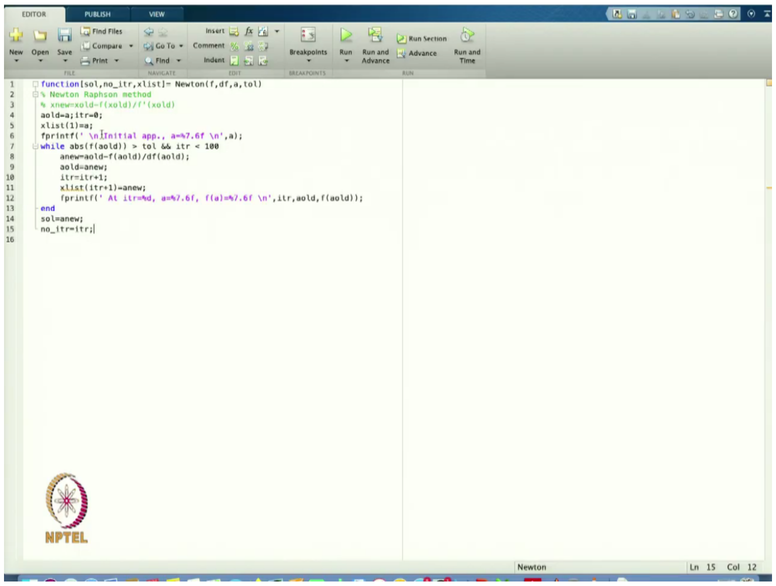
(Refer Slide Time: 00:54)



```matlab
function [root,froot,itr,xlist]= fixedpt(f,g,dg,a,tol)
% f(x)=0 x is a root
% x=g(x)
% itrative method is x(n+1)=g(x(n))
% for the convergence |g'(x0)|<1
% a is the initial guess
% tol=Tolarence
maxitr=100;
% feval(dg,a) can be written as dg(a)
if dg(a) > 1
    disp(' For the given a |dg(a)|>1, so may not converge');
    return;
end
itr=0;
xlist(1)=a;
fprintf('\n initial guess, a=%7.6f, dg(a)=%7.6f \n',a,dg(a));
anew=g(a);
while abs(f(anew))> tol
    if (maxitr <=itr)
        disp(' Maxumum no. of itrations reaches');
        break
    end
    aold=anew;
    anew=g(aold);
    itr=itr+1;
    xlist(itr+1)=anew;
     err=abs(anew-aold);
     fprintf('\n At itr=%d,  a=%7.6f, f(a)=%7.6f \n',itr,anew,f(anew));
end
root=anew;
froot=f(anew);
```

So this is my fixed point iteration.

(Refer Slide Time: 00:57)



```
function[sol,no_itr,xlist]= Newton(f,df,a,tol)
% Newton Raphson method
% xnew=xold-f(xold)/f'(xold)
aold=a;itr=0;
xlist(1)=a;
fprintf(' \nInitial app., a=%7.6f \n',a);
while abs(f(aold)) > tol && itr < 100
    anew=aold-f(aold)/df(aold);
    aold=anew;
    itr=itr+1;
    xlist(itr+1)=anew;
    fprintf(' At itr=%d, a=%7.6f, f(a)=%7.6f \n',itr,aold,f(aold));
end
sol=anew;
no_itr=itr;
```

So I will start writing a new code. So let us define a function. So, the output of this function is I want my solution and the number of iterations and then I want xlist. So, I want only three output from there. So that is equal to I will write Newton method, the argument to the function will be passed, the derivative of the function will be passed the initial data and the tolerance. So, this is the requirement of the Newton-Raphson methods.

Now from the Newton-Raphson method, I will write so from here I can write that the and from here we know that that is equal to x at xnew I can write:

$$x_{new} = x_{old} - \frac{f\left(x_{old}\right)}{f'\left(x_{old}\right)}$$

So, I will this value will be saved in the aold the new variable I am defining, I will start my itr = 0. Then, I will define my xlist so, the value this is equal to the initial a whatever we have and then I can define my f print f. So, it will give you that so I will write at the initial a new line and I will write my initial approximation or initial guess.

function[sol,no_itr,xlist]= Newton(f,df,a,tol)

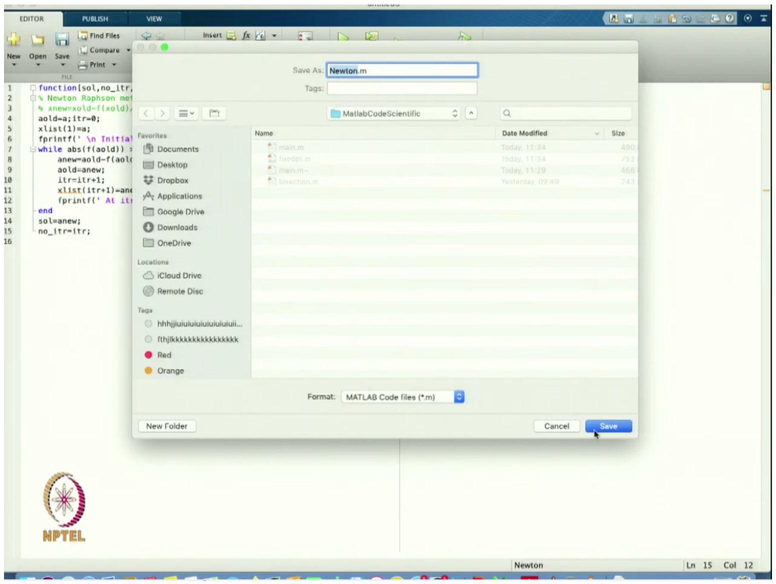
% Newton Raphson method

```
% xnew=xold-f(xold)/f'(xold)

aold=a;itr=0;

xlist(1)=a;

fprintf(' \n Initial app., a=%7.6f \n',a);

while abs(f(aold)) > tol && itr < 100

    anew=aold-f(aold)/df(aold);

    aold=anew;

    itr=itr+1;

    xlist(itr+1)=anew;

    fprintf(' At itr=%d, a=%7.6f, f(a)=%7.6f \n',itr,aold,f(aold));

end

sol=anew;

no_itr=itr;
```

So this while loop will, I will enter the while loop when this value and this value both are true. If either of them is not true, then this will come out from the while loop. So I am finding that what is the value of the f at the root, if it is either greater than tolerance and the iteration is less than 100.

So, in that case I will enter there and then I will find my anew. So, this I can put on my x list is equal to itr + 1. So, that is equal to a. So, what are the anew or aold because here it is value same, I will put the value here and then I will write from here I will write my value of fprintf. So, I can find number itr, so that is equal to itr, so, that is the Newton-Raphson method.

(Refer Slide Time: 08:52)

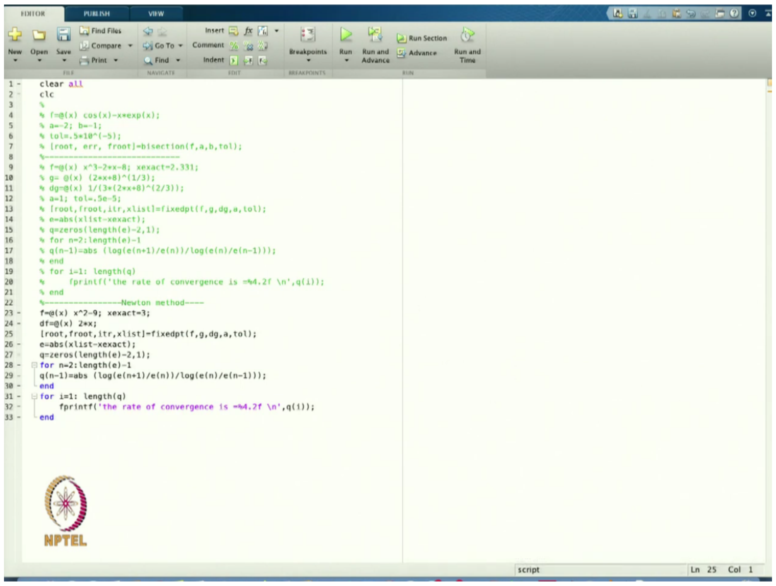


```
function[sol,no_itr,
% Newton Raphson met
% xnew=xold-f(xold)/
aold=a;itr=0;
xlist(1)=a;
fprintf(' \n Initial
while abs(f(aold)) >
    anew=aold-f(aolc
    aold=anew;
    itr=itr+1;
    xlist(itr+1)=ane
    fprintf(' At it
end
sol=anew;
no_itr=itr;
```

Save As: Newton.m
Tags:
Format: MATLAB Code files (*.m)

I can save this one as save as Newton. So, this is the Newton-Raphson method now.

(Refer Slide Time: 08:58)



```
clear all
clc
%
% f=@(x) cos(x)-x*exp(x);
% a=-2; b=-1;
% tol=.5*10^(-5);
% [root, err, froot]=bisection(f,a,b,tol);
%----------------------------
% f=@(x) x^3-2*x-8; xexact=2.331;
% g= @(x) (2*x+8)^(1/3);
% dg=@(x) 1/(3*(2*x+8)^(2/3));
% a=1; tol=.5e-5;
% [root,froot,itr,xlist]=fixedpt(f,g,dg,a,tol);
% e=abs(xlist-xexact);
% q=zeros(length(e)-2,1);
% for n=2:length(e)-1
% q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
% end
% for i=1: length(q)
%     fprintf('the rate of convergence is =%4.2f \n',q(i));
% end
%----------------Newton method----
f=@(x) x^2-9; xexact=3;
df=@(x) 2*x;
[root,froot,itr,xlist]=fixedpt(f,g,dg,a,tol);
e=abs(xlist-xexact);
q=zeros(length(e)-2,1);
for n=2:length(e)-1
q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
end
for i=1: length(q)
    fprintf('the rate of convergence is =%4.2f \n',q(i));
end
```

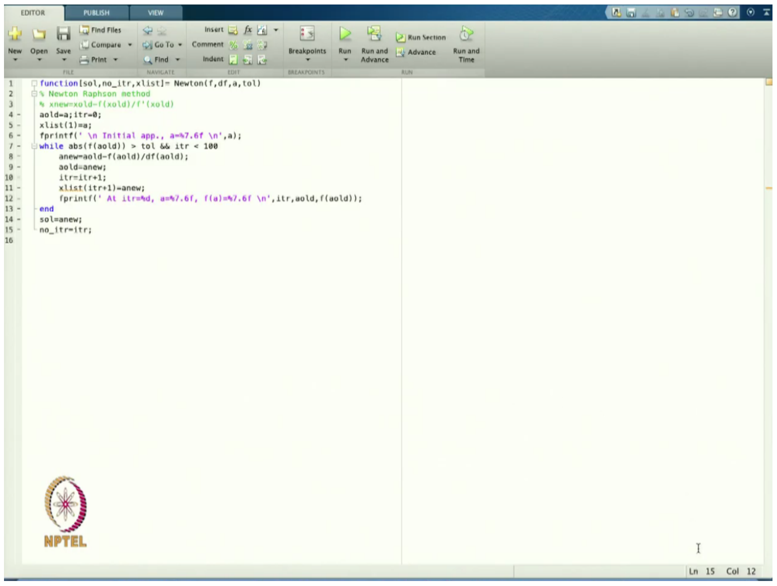
Now I can change in the main loop. So, now everything will be same except all these terms up to here I can now everything will be same here. So maybe I can so the same code I will use for Newton method. So in the Newton methods, we define the new function, so let us write down the function. So, let us say take f is equal to at the rate maybe I will take a simple function.

x^2-9 =0

So this is the function I am defining. then I define df so, df = 2x. So, this is the value of the 2x. Now, I know that from this equation the xexact = 3 or -3. So, in this case we have a roots x square is equal to 9. So, x= ± 3, I am just looking for the positive root. So, I will just take xexact = 3. Now, in this case so, other things I will just print from the above and I will write and I will comment this one.

Now, my fe this one and this is the root I am finding. So here, I am having the four outputs, let us see that how many outputs are there in Newton.

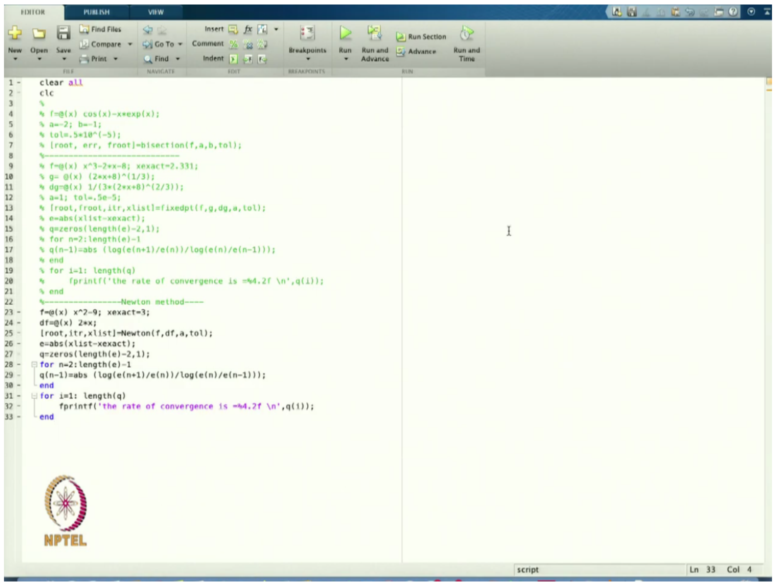
(Refer Slide Time: 11:06)



```
function[sol,no_itr,xlist]= Newton(f,df,a,tol)
% Newton Raphson method
% xnew=xold-f(xold)/f'(xold)
aold=a;itr=0;
xlist(1)=a;
fprintf(' \n Initial app., a=%7.6f \n',a);
while abs(f(aold)) > tol && itr < 100
    anew=aold-f(aold)/df(aold);
    aold=anew;
    itr=itr+1;
    xlist(itr+1)=anew;
    fprintf(' At itr=%d, a=%7.6f, f(a)=%7.6f \n',itr,aold,f(aold));
end
sol=anew;
no_itr=itr;
```

So you can see that solution, number of iteration and x list. So we have only three numbers of outputs, so I have to change this one.

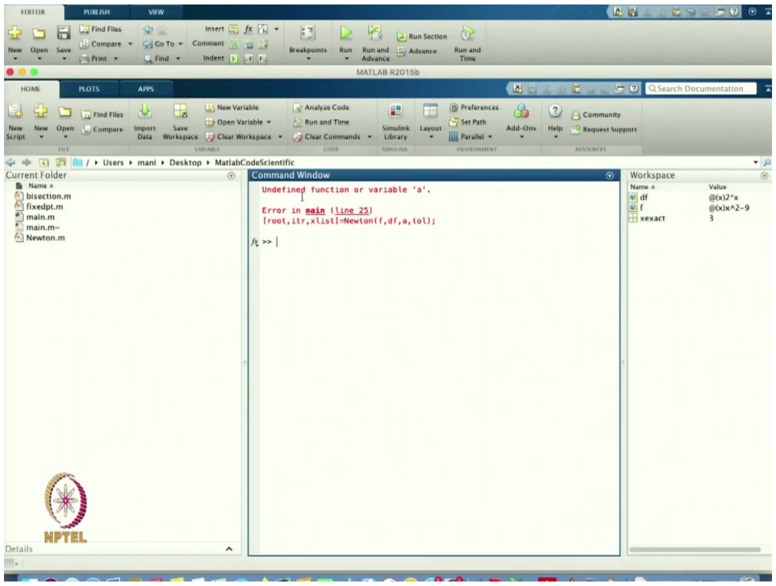
(Refer Slide Time: 11:17)



```
clear all
clc
%
% f=@(x) cos(x)-x*exp(x);
% a=-2; b=-1;
% tol=.5*10^(-5);
% [root, err, froot]=bisection(f,a,b,tol);
%---------------------------
% f=@(x) x^3-2*x-8; xexact=2.331;
% g= @(x) (2*x+8)^(1/3);
% dg=@(x) 1/(3*(2*x+8)^(2/3));
% a=1; tol=.5e-5;
% [root,froot,itr,xlist]=fixedpt(f,g,dg,a,tol);
% e=abs(xlist-xexact);
% q=zeros(length(e)-2,1);
% for n=2:length(e)-1
% q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
% end
% for i=1: length(q)
%     fprintf('the rate of convergence is =%4.2f \n',q(i));
% end
%----------------Newton method----
f=@(x) x^2-9; xexact=3;
df=@(x) 2*x;
[root,itr,xlist]=Newton(f,df,a,tol);
e=abs(xlist-xexact);
q=zeros(length(e)-2,1);
for n=2:length(e)-1
q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
end
for i=1: length(q)
    fprintf('the rate of convergence is =%4.2f \n',q(i));
end
```

So I am getting this. So I am not passing the value of the function at that. So I am passing my root and iteration the list. And this is the value I am passing, so fixed, now I have to change this one to Newton. So in the Newton, I am passing my f, my df the value of a and the tolerance. So from here, I am getting my xlist xexact, this is the value and that is the convergence. So I hope I should this should work. So I will just save this one and run this.

(Refer Slide Time: 12:04)

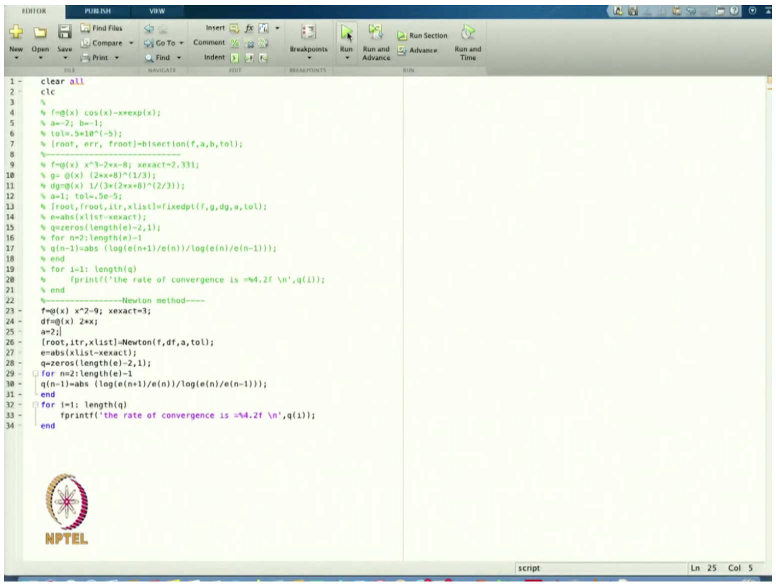


```
Undefined function or variable 'a'.

Error in main (line 25)
[root,itr,xlist]=Newton(f,df,a,tol);

>> 
```

Name	Value
df	@(x)2*x
f	@(x)x^2-9
xexact	3

So let us see what is happening here. Undefined function or variable a, in the main.

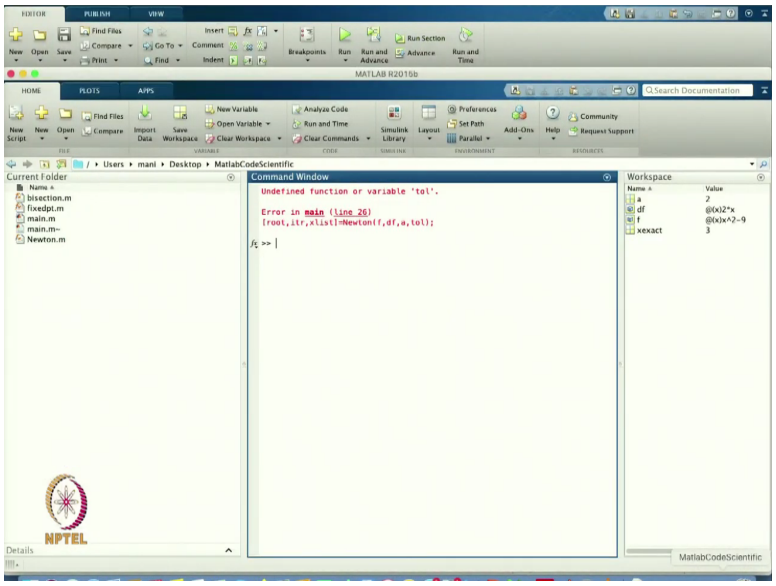
(Refer Slide Time: 12:09)



```
clear all
clc
%
% f=@(x) cos(x)-x*exp(x);
% a=-2; b=-1;
% tol=.5*10^(-5);
% [root, err, froot]=bisection(f,a,b,tol);
%--------------------------
% f=@(x) x^3-2*x-8; xexact=2.331;
% g= @(x) (2*x+8)^(1/3);
% dg=@(x) 1/(3*(2*x+8)^(2/3));
% a=1; tol=.5e-5;
% [root,froot,itr,xlist]=fixedpt(f,g,dg,a,tol);
% e=abs(xlist-xexact);
% q=zeros(length(e)-2,1);
% for n=2:length(e)-1
% q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
% end
% for i=1: length(q)
%     fprintf('the rate of convergence is =%4.2f \n',q(i));
% end
%---------------Newton method----
f=@(x) x^2-9; xexact=3;
df=@(x) 2*x;
a=2;
[root,itr,xlist]=Newton(f,df,a,tol);
e=abs(xlist-xexact);
q=zeros(length(e)-2,1);
for n=2:length(e)-1
q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
end
for i=1: length(q)
    fprintf('the rate of convergence is =%4.2f \n',q(i));
end
```

This is my main. So my a is I am suppose 3 is the root. So let us start with 2, so I am starting with the a = 2, now let us see.

(Refer Slide Time: 12:31)



Current Folder
- bisection.m
- fixedpt.m
- main.m
- main.m~
- Newton.m

Command Window

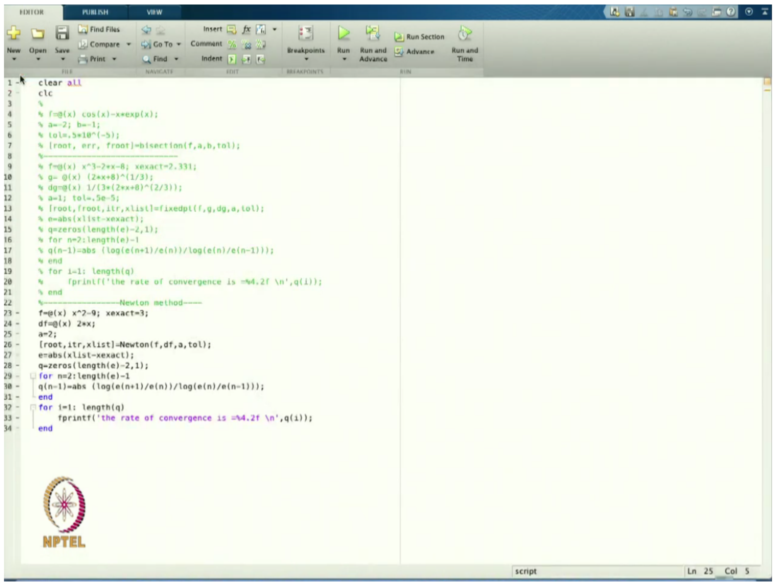
```
Undefined function or variable 'tol'.

Error in main (line 26)
[root,itr,xlist]=Newton(f,df,a,tol);

>>
```

Workspace

Name	Value
a	2
df	@(x)2*x
f	@(x)x^2-9
xexact	3

So undefined the tolerance because I have not defined the tolerance there.

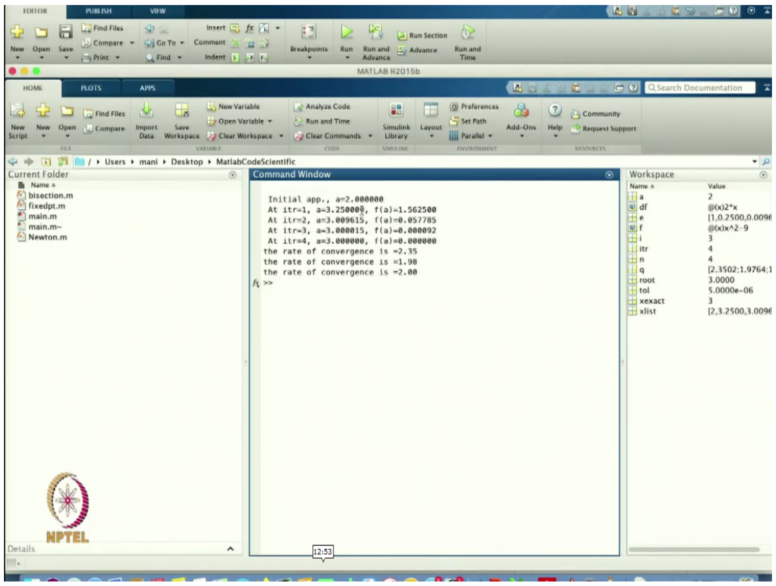
(Refer Slide Time: 12:37)



```
clear all
clc
%
% f=@(x) cos(x)-x*exp(x);
% a=-2; b=-1;
% tol=.5*10^(-5);
% [root, err, froot]=bisection(f,a,b,tol);
%-------------------------------
% f=@(x) x^3-2*x-8; xexact=2.331;
% g= @(x) (2*x+8)^(1/3);
% dg=@(x) 1/(3*(2*x+8)^(2/3));
% a=1; tol=.5e-5;
% [root,froot,itr,xlist]=fixedpt(f,g,dg,a,tol);
% e=abs(xlist-xexact);
% q=zeros(length(e)-2,1);
% for n=2:length(e)-1
% q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
% end
% for i=1: length(q)
%     fprintf('the rate of convergence is =%4.2f \n',q(i));
% end
%-----------------Newton method----
f=@(x) x^2-9; xexact=3;
df=@(x) 2*x;
a=2;
[root,itr,xlist]=Newton(f,df,a,tol);
e=abs(xlist-xexact);
q=zeros(length(e)-2,1);
for n=2:length(e)-1
q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
end
for i=1: length(q)
    fprintf('the rate of convergence is =%4.2f \n',q(i));
end
```

So let us define the tolerance also. So I will define the same tolerance, now it is done.

(Refer Slide Time: 12:52)



```
 Initial app., a=2.000000
 At itr=1, a=3.250000, f(a)=1.562500
 At itr=2, a=3.009615, f(a)=0.057785
 At itr=3, a=3.000015, f(a)=0.000092
 At itr=4, a=3.000000, f(a)=0.000000
the rate of convergence is =2.35
the rate of convergence is =1.98
the rate of convergence is =2.00
>>
```

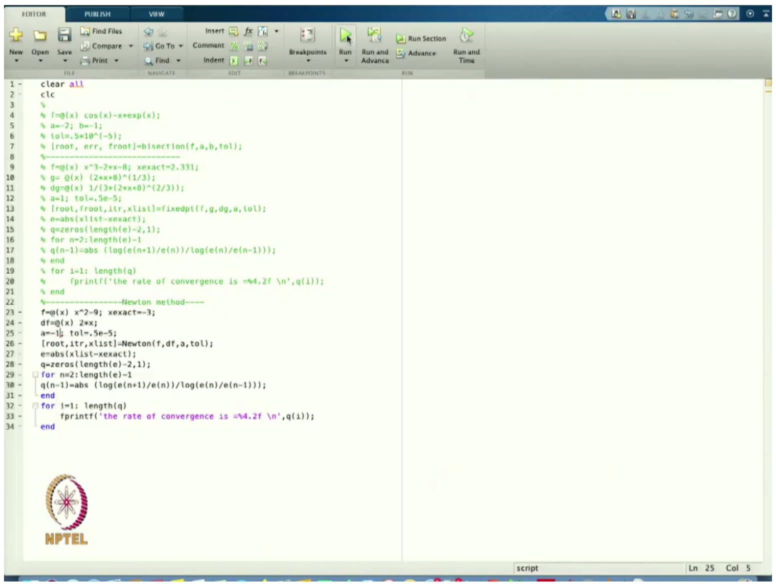
Name	Value
a	2
df	@(x)2*x
e	[1,0.2500,0.0096
f	@(x)x^2-9
i	3
itr	4
n	4
q	[2.3502;1.9764;1
root	3.0000
tol	5.0000e-06
xexact	3
xlist	[2,3.2500,3.0096

And that is the solution. So from here you can see that I started with the a = 2 and then my roots approximation to the roots is the first approximation it is giving a = 3.25 that is the value of the function. And after the fourth iteration it gives you the value 3 exactly. And that is the value of the function and the rate of convergence it started with 2.35, then 1.98. And after that it is given 2.

```
%----------------Newton method----

 f=@(x) x^3-8; xexact=2;

 df=@(x) 3*x^2;

 a=1; tol=.5e-5;

 [root,itr,xlist]=Newton(f,df,a,tol);
```

And we also know that this rate of convergence for the Newton-Raphson method is of order 2. So, in this case it is coming exactly 2.

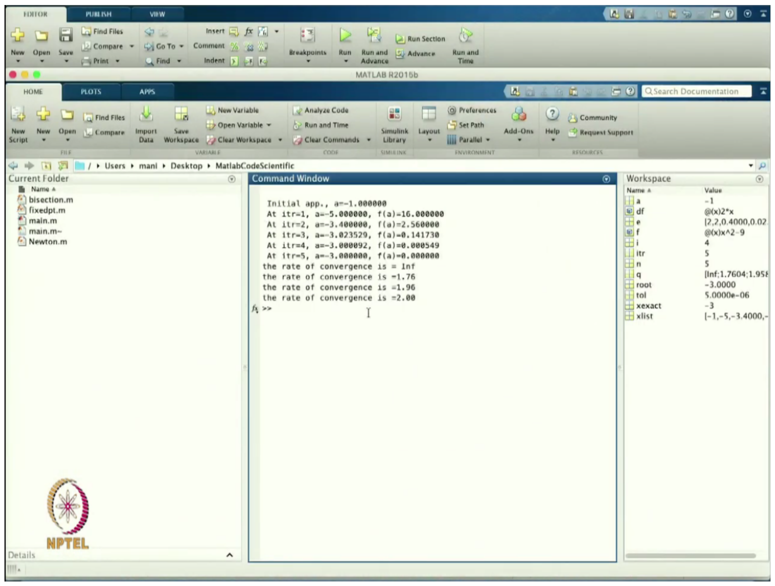
(Refer Slide Time: 13:32)



```
clear all
clc
%
% f=@(x) cos(x)-x*exp(x);
% a=-2; b=-1;
% tol=.5*10^(-5);
% [root, err, froot]=bisection(f,a,b,tol);
%---------------------------
% f=@(x) x^3-2*x-8; xexact=2.331;
% g= @(x) (2*x+8)^(1/3);
% dg=@(x) 1/(3*(2*x+8)^(2/3));
% a=1; tol=.5e-5;
% [root,froot,itr,xlist]=fixedpt(f,g,dg,a,tol);
% e=abs(xlist-xexact);
% q=zeros(length(e)-2,1);
% for n=2:length(e)-1
% q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
% end
% for i=1: length(q)
%     fprintf('the rate of convergence is =%4.2f \n',q(i));
% end
%---------------Newton method----
f=@(x) x^2-9; xexact=-3;
df=@(x) 2*x;
a=-1; tol=.5e-5;
[root,itr,xlist]=Newton(f,df,a,tol);
e=abs(xlist-xexact);
q=zeros(length(e)-2,1);
for n=2:length(e)-1
q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
end
for i=1: length(q)
    fprintf('the rate of convergence is =%4.2f \n',q(i));
end
```

Because in this case, I am also getting the exact root that is equal to 3, because in the previous well case, we are getting the exact root is equal to 2.331. So, that is also the approximation root of that, but in this case it is giving you the same value. So, maybe I can change this one and let us take minus 3 and I will start at maybe -1, let us see what would happen.

(Refer Slide Time: 14:07)



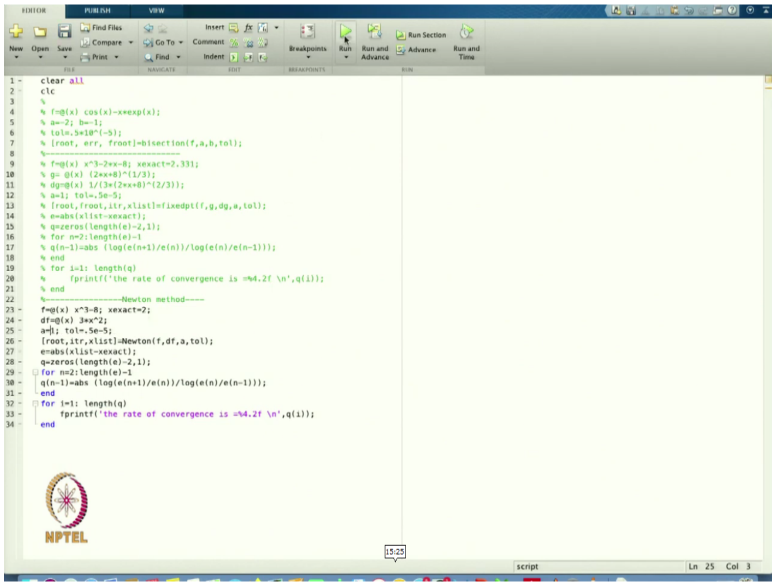
```
 Initial app., a=-1.000000
 At itr=1, a=-5.000000, f(a)=16.000000
 At itr=2, a=-3.400000, f(a)=2.560000
 At itr=3, a=-3.023529, f(a)=0.141730
 At itr=4, a=-3.000092, f(a)=0.000549
 At itr=5, a=-3.000000, f(a)=0.000000
the rate of convergence is = Inf
the rate of convergence is =1.76
the rate of convergence is =1.96
the rate of convergence is =2.00
>>
```

Name	Value
a	-1
df	@(x)2*x
e	[2,2,0.4000,0.02
f	@(x)x^2-9
i	4
itr	5
n	5
q	[Inf;1.7604;1.95
root	-3.0000
tol	5.0000e-06
xexact	-3
xlist	[-1,-5,-3.4000,-

So, in this case, it is giving the 2 that infinity like it is coming like, because somewhere we have to find the value of absolute value. So, it started with 1, then -5, then -3, -3, -3 and then it came to 3. because the initial approximation, if you see from the Newton-Raphson method, we have a sufficient condition for the convergence. So, that sufficient condition we have to take care whenever we start with the initial condition.

So, here I am starting just with the a raise to minus 1 and then later on we found that we are reaching towards the root and root in - 3 and then the rate of convergence here also coming 2, so that we have to take care.

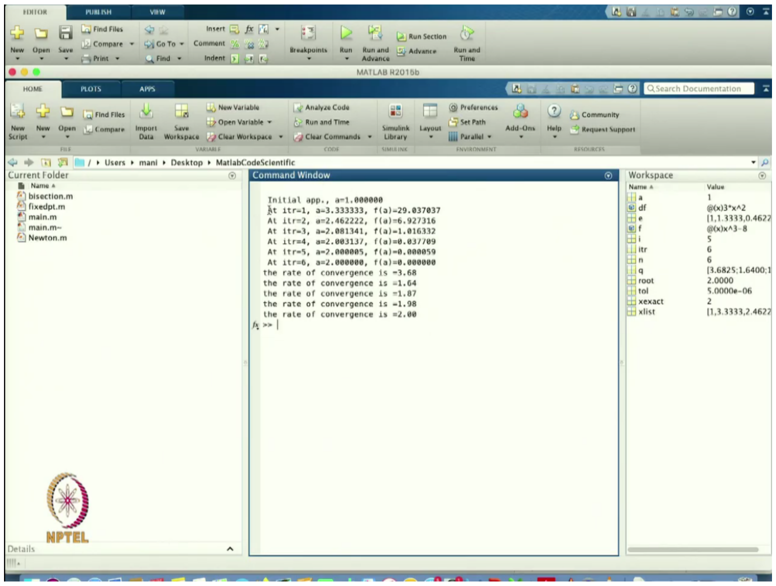
(Refer Slide Time: 14:55)



```
clear all
clc
%
% f=@(x) cos(x)-x*exp(x);
% a=-2; b=-1;
% tol=.5*10^(-5);
% [root, err, froot]=bisection(f,a,b,tol);
%---------------------------
% f=@(x) x^3-2*x-8; xexact=2.331;
% g= @(x) (2*x+8)^(1/3);
% dg=@(x) 1/(3*(2*x+8)^(2/3));
% a=1; tol=.5e-5;
% [root,froot,itr,xlist]=fixedpt(f,g,dg,a,tol);
% e=abs(xlist-xexact);
% q=zeros(length(e)-2,1);
% for n=2:length(e)-1
% q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
% end
% for i=1: length(q)
%     fprintf('the rate of convergence is =%4.2f \n',q(i));
% end
%---------------Newton method----
f=@(x) x^3-8; xexact=2;
df=@(x) 3*x^2;
a=1; tol=.5e-5;
[root,itr,xlist]=Newton(f,df,a,tol);
e=abs(xlist-xexact);
q=zeros(length(e)-2,1);
for n=2:length(e)-1
q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
end
for i=1: length(q)
    fprintf('the rate of convergence is =%4.2f \n',q(i));
end
```

Maybe I can start with some other equation and then let us write it. So, in this case what is happening and getting my exact root that is 2 this will be 3 star x raise to power 2. And then I will start with suppose 1 let us see this one.

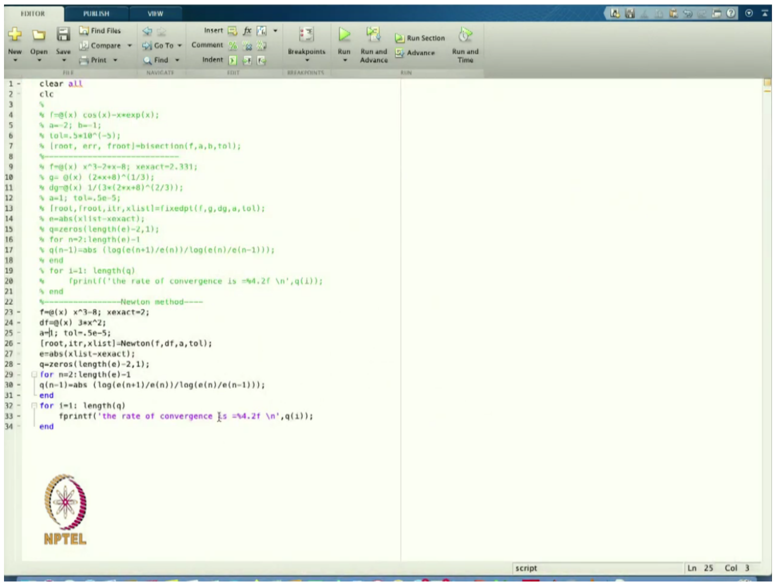
(Refer Slide Time: 15:28)



```
Initial app., a=1.000000
At itr=1, a=3.333333, f(a)=29.037037
At itr=2, a=2.462222, f(a)=6.927316
At itr=3, a=2.081341, f(a)=1.016332
At itr=4, a=2.003137, f(a)=0.037709
At itr=5, a=2.000005, f(a)=0.000059
At itr=6, a=2.000000, f(a)=0.000000
the rate of convergence is =3.68
the rate of convergence is =1.64
the rate of convergence is =1.87
the rate of convergence is =1.98
the rate of convergence is =2.00
>>
```

And then I will find out the root here or that is the equation. So, I will start with 1 then it is giving the first iteration and after the 6 iteration I am getting the root that is 2 and the rate of convergence is 2.

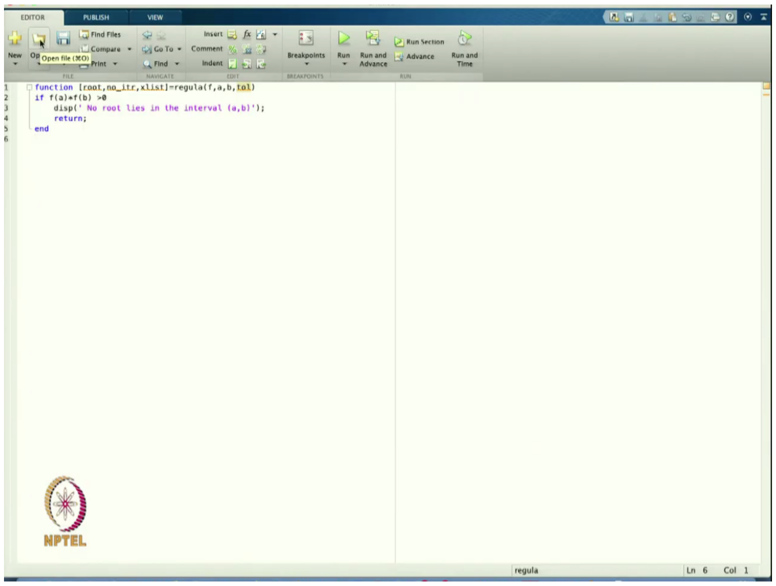
(Refer Slide Time: 15:52)



```
clear all
clc
%
% f=@(x) cos(x)-x*exp(x);
% a=-2; b=-1;
% tol=.5*10^(-5);
% [root, err, froot]=bisection(f,a,b,tol);
%---------------------------
% f=@(x) x^3-2*x-8; xexact=2.331;
% g= @(x) (2*x+8)^(1/3);
% dg=@(x) 1/(3*(2*x+8)^(2/3));
% a=1; tol=.5e-5;
% [root,froot,itr,xlist]=fixedpt(f,g,dg,a,tol);
% e=abs(xlist-xexact);
% q=zeros(length(e)-2,1);
% for n=2:length(e)-1
% q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
% end
% for i=1: length(q)
%     fprintf('the rate of convergence is =%4.2f \n',q(i));
% end
%----------------Newton method----
f=@(x) x^3-8; xexact=2;
df=@(x) 3*x^2;
a=1; tol=.5e-5;
[root,itr,xlist]=Newton(f,df,a,tol);
e=abs(xlist-xexact);
q=zeros(length(e)-2,1);
for n=2:length(e)-1
q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
end
for i=1: length(q)
    fprintf('the rate of convergence is =%4.2f \n',q(i));
end
```

So, in this case we can see that Newton-Raphson method, we started with the simple equation and after doing this one we are able to find that the rate of convergence for the Newton-Raphson is 2, so that is all about how we can find the Newton-Raphson method. Now we can write the new code also.

(Refer Slide Time: 16:09)



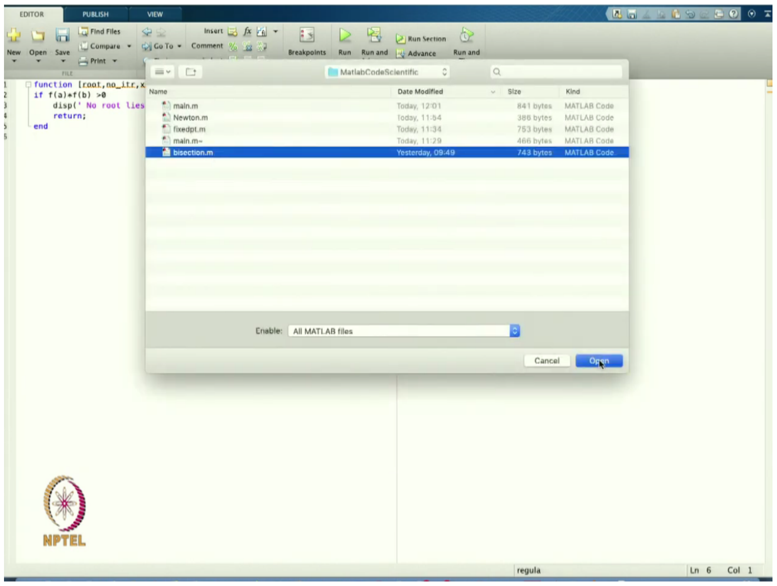
```
function [root,no_itr,xlist]=regula(f,a,b,tol)
if f(a)*f(b) >0
    disp(' No root lies in the interval (a,b)');
    return;
end
```

So let us start with the new code. And that code we are going to define is basically Regula-Falsi. So, let us do the code of Regula-Falsi. So in this case of I will define the function and then I will find the value of root and iteration, the number of iteration and then xlist and then I will find Regula and then in this Regula I will pass my function f the initial two guess because here we have to give the two guesses and then the tolerance.

So, this is the function Regula we have to define and then, so in this case also I can write that what is the value of this one.

```
function [root,no_itr,xlist]=regula(f,a,b,tol)

if f(a)*f(b) >0

    disp(' No root lies in the interval (a,b)');

    return;

end
```

(Refer Slide Time: 18:27)



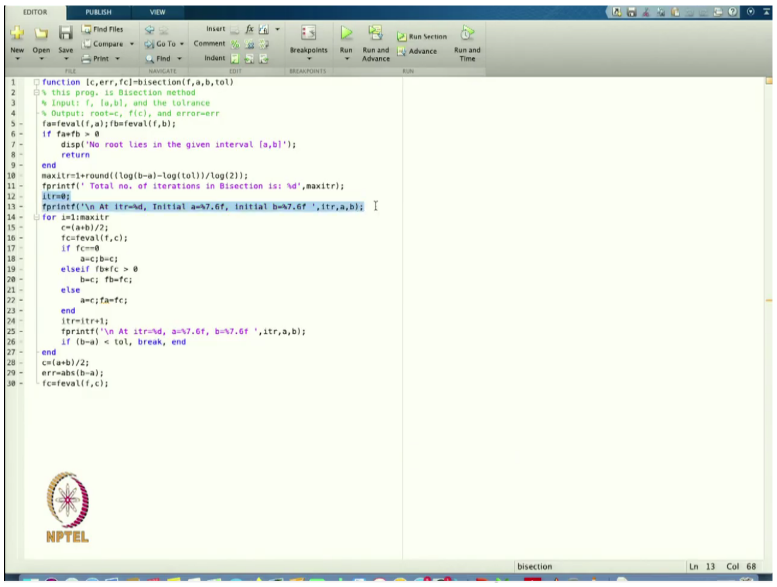
```
function [root,no_itr,x
if f(a)*f(b) >0
    disp(' No root lies
    return;
end
```

Name	Date Modified	Size	Kind
main.m	Today, 12:01	841 bytes	MATLAB Code
Newton.m	Today, 11:54	386 bytes	MATLAB Code
fixedpt.m	Today, 11:34	753 bytes	MATLAB Code
main.m~	Today, 11:29	466 bytes	MATLAB Code
bisection.m	Yesterday, 09:49	743 bytes	MATLAB Code

Enable: All MATLAB files

So, instead of writing again and again the same thing I maybe just copy so, I will open the bisection method.

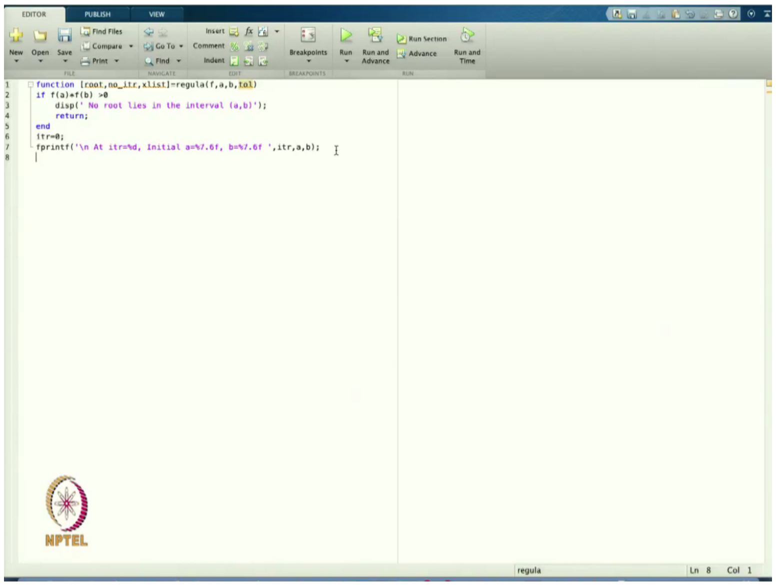
(Refer Slide Time: 18:31)



```
function [c,err,fc]=bisection(f,a,b,tol)
% this prog. is Bisection method
% Input: f, [a,b], and the tolrance
% Output: root=c, f(c), and error=err
fa=feval(f,a);fb=feval(f,b);
if fa*fb > 0
    disp('No root lies in the given interval [a,b]');
    return
end
maxitr=1+round((log(b-a)-log(tol))/log(2));
fprintf(' Total no. of iterations in Bisection is: %d',maxitr);
itr=0;
fprintf('\n At itr=%d, Initial a=%7.6f, initial b=%7.6f ',itr,a,b);
for i=1:maxitr
    c=(a+b)/2;
    fc=feval(f,c);
    if fc==0
        a=c;b=c;
    elseif fb*fc > 0
        b=c; fb=fc;
    else
        a=c;fa=fc;
    end
    itr=itr+1;
    fprintf('\n At itr=%d, a=%7.6f, b=%7.6f ',itr,a,b);
    if (b-a) < tol, break, end
end
c=(a+b)/2;
err=abs(b-a);
fc=feval(f,c);
```

So, I will just because in the bisection method, we had the advantage that we can find the number of iteration we are going to do to get the to achieve that accuracy of the root but in this case, we cannot do that. So, I will start with that iteration 0 and that is the value.

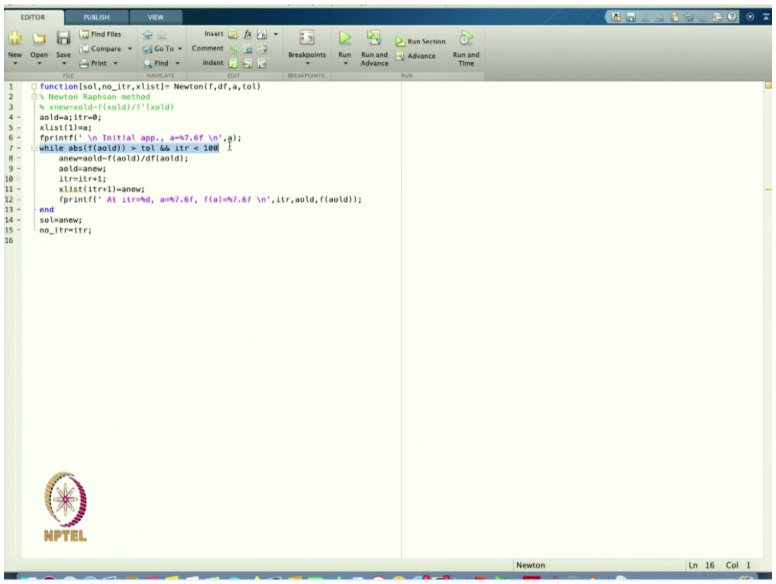
(Refer Slide Time: 18:57)



```
function [root,no_itr,xlist]=regula(f,a,b,tol)
if f(a)*f(b) >0
    disp(' No root lies in the interval (a,b)');
    return;
end
itr=0;
fprintf('\n At itr=%d, Initial a=%7.6f, b=%7.6f ',itr,a,b);
```

So, I just copy paste from here and I can apply here, so I start with the iteration equal to 0 and then at the iteration 0 initial value a and b, I can define. So, this is the value it will be shown. Now, I will again I the same thing.

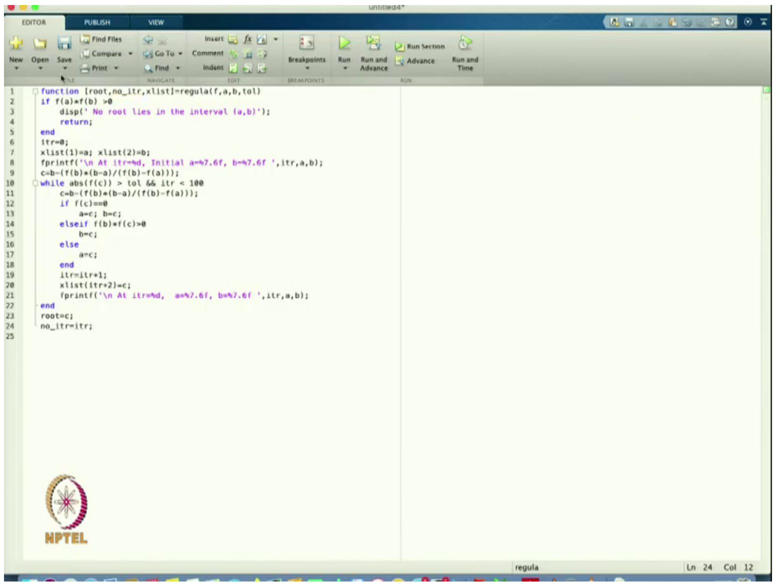
(Refer Slide Time: 19:27)



```
function[sol,no_itr,xlist]= Newton(f,df,a,tol)
% Newton Raphson method
% xnew=xold-f(xold)/f'(xold)
aold=a;itr=0;
xlist(1)=a;
fprintf(' \n Initial app., a=%7.6f \n',a);
while abs(f(aold)) > tol && itr < 100
    anew=aold-f(aold)/df(aold);
    aold=anew;
    itr=itr+1;
    xlist(itr+1)=anew;
    fprintf(' At itr=%d, a=%7.6f, f(a)=%7.6f \n',itr,aold,f(aold));
end
sol=anew;
no_itr=itr;
```

So, I will copy this point Newton-Raphson.

(Refer Slide Time: 19:36)



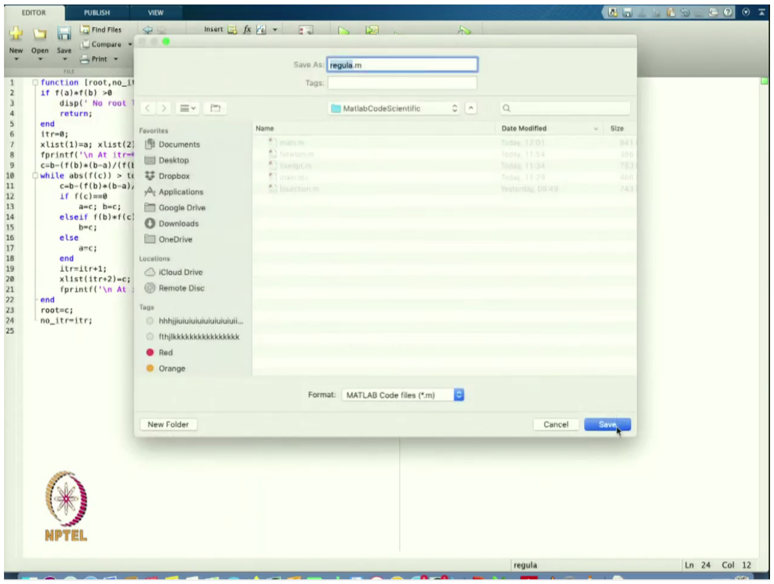
```
function [root,no_itr,xlist]=regula(f,a,b,tol)
if f(a)*f(b) >0
    disp(' No root lies in the interval (a,b)');
    return;
end
itr=0;
xlist(1)=a; xlist(2)=b;
fprintf('\n At itr=%d, Initial a=%7.6f, b=%7.6f ',itr,a,b);
c=b-(f(b)*(b-a)/(f(b)-f(a)));
while abs(f(c)) > tol && itr < 100
    c=b-(f(b)*(b-a)/(f(b)-f(a)));
    if f(c)==0
        a=c; b=c;
    elseif f(b)*f(c)>0
        b=c;
    else
        a=c;
    end
    itr=itr+1;
    xlist(itr+2)=c;
    fprintf('\n At itr=%d,  a=%7.6f, b=%7.6f ',itr,a,b);
end
root=c;
no_itr=itr;
```

```
function [root,no_itr,xlist]=regula(f,a,b,tol)

if f(a)*f(b) >0

    disp(' No root lies in the interval (a,b)');

    return;

end

itr=0;

xlist(1)=a; xlist(2)=b;

fprintf('\n At itr=%d, Initial a=%7.6f, b=%7.6f ',itr,a,b);

c=b-(f(b)*(b-a)/(f(b)-f(a)));

while abs(f(c)) > tol && itr < 100

    c=b-(f(b)*(b-a)/(f(b)-f(a)));

    if f(c)==0

        a=c; b=c;

    elseif f(b)*f(c)>0
```

```
        b=c;
    else
        a=c;
    end
    itr=itr+1;
    xlist(itr+2)=c;
    fprintf('\n At itr=%d,  a=%7.6f, b=%7.6f ',itr,a,b);
end
root=c;
no_itr=itr;
```

So this is the iteration initial a. So this is not the value of these iterations a and b I am getting and then I will put the end of this. So, from here I will get my root that is, that is equal to c and the number of iterations so, that is equal to itr. So, that is the value we are going to get.

(Refer Slide Time: 24:25)



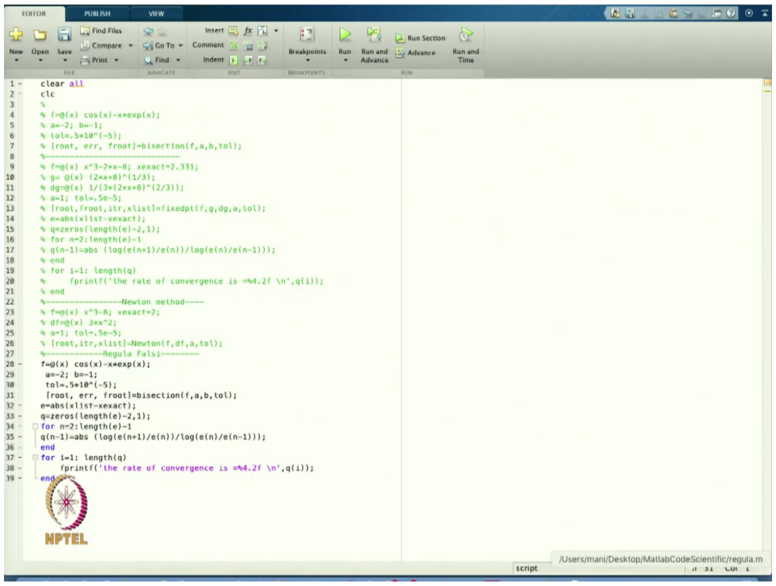
```
function [root,no_it
if f(a)*f(b) >0
    disp(' No root 
    return;
end
itr=0;
xlist(1)=a; xlist(2
fprintf('\n At itr=
c=b-(f(b)*(b-a)/(f(
while abs(f(c)) > t
    c=b-(f(b)*(b-a)/
    if f(c)==0
        a=c; b=c;
    elseif f(b)*f(c
        b=c;
    else
        a=c;
    end
    itr=itr+1;
    xlist(itr+2)=c;
    fprintf('\n At 
end
root=c;
no_itr=itr;
```

Save As: regula.m

Format: MATLAB Code files (*.m)

And I will save this function as Regula.

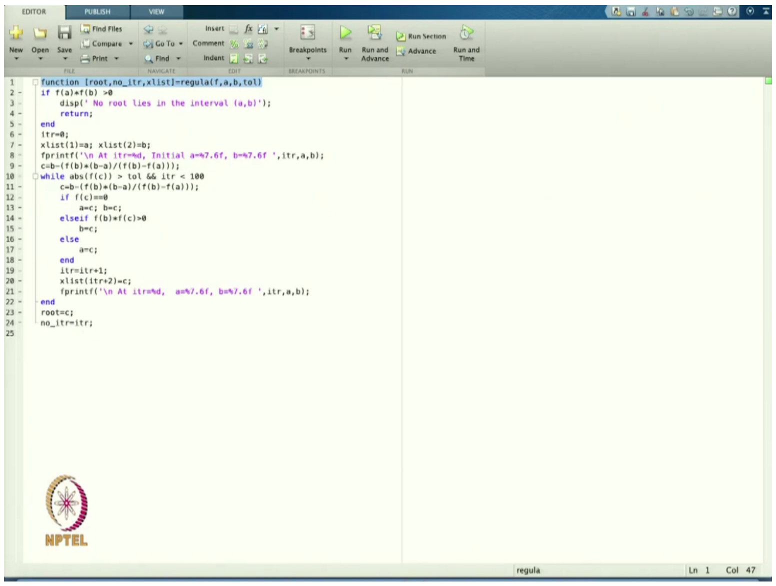
(Refer Slide Time: 24:35)



```
clear all
clc
%
% f=@(x) cos(x)-x*exp(x);
% a=-2; b=-1;
% tol=.5*10^(-5);
% [root, err, froot]=bisection(f,a,b,tol);
%---------------------------
% f=@(x) x^3-2*x-8; xexact=2.331;
% g= @(x) (2*x+8)^(1/3);
% dg=@(x) 1/(3*(2*x+8)^(2/3));
% a=1; tol=.5e-5;
% [root,froot,itr,xlist]=fixedpt(f,g,dg,a,tol);
% e=abs(xlist-xexact);
% q=zeros(length(e)-2,1);
% for n=2:length(e)-1
% q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
% end
% for i=1: length(q)
%     fprintf('the rate of convergence is =%4.2f \n',q(i));
% end
%---------------Newton method----
% f=@(x) x^3-8; xexact=2;
% df=@(x) 3*x^2;
% a=1; tol=.5e-5;
% [root,itr,xlist]=Newton(f,df,a,tol);
%-----------Regula Falsi--------
f=@(x) cos(x)-x*exp(x);
 a=-2; b=-1;
 tol=.5*10^(-5);
 [root, err, froot]=bisection(f,a,b,tol);
e=abs(xlist-xexact);
q=zeros(length(e)-2,1);
for n=2:length(e)-1
q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
end
for i=1: length(q)
    fprintf('the rate of convergence is =%4.2f \n',q(i));
end
```

Now, I will go to the main program. So, this is the program we are going to get so, in this case also we are we have to in this one I just make the comment and all other I will so, in the Bisection method, I know that two initial approximations are needed, so I will use this one. So I will take my f here, and then my a and b is minus 1 then is equal to tolerance I am taking 0.5 and then from here I am calling this Regula-Falsi method.

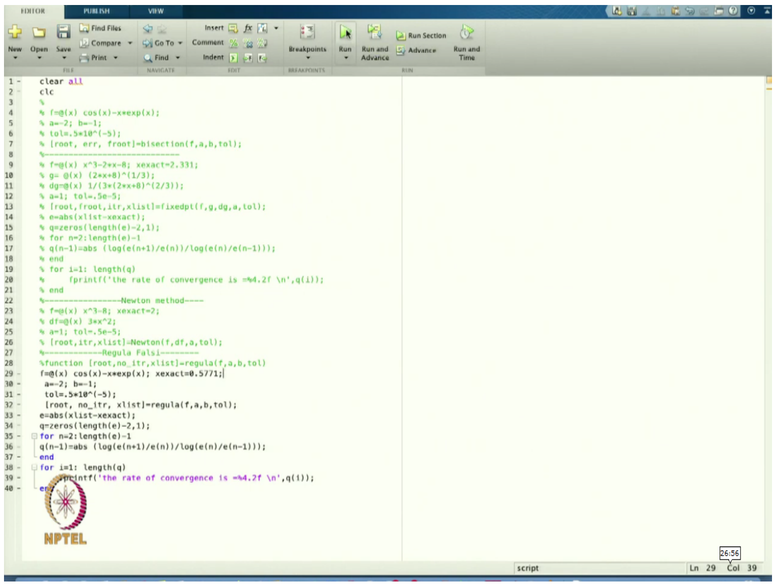
(Refer Slide Time: 25:32)



```
function [root,no_itr,xlist]=regula(f,a,b,tol)
if f(a)*f(b) >0
    disp(' No root lies in the interval (a,b)');
    return;
end
itr=0;
xlist(1)=a; xlist(2)=b;
fprintf('\n At itr=%d, Initial a=%7.6f, b=%7.6f ',itr,a,b);
c=b-(f(b)*(b-a)/(f(b)-f(a)));
while abs(f(c)) > tol && itr < 100
    c=b-(f(b)*(b-a)/(f(b)-f(a)));
    if f(c)==0
        a=c; b=c;
    elseif f(b)*f(c)>0
        b=c;
    else
        a=c;
    end
    itr=itr+1;
    xlist(itr+2)=c;
    fprintf('\n At itr=%d,  a=%7.6f, b=%7.6f ',itr,a,b);
end
root=c;
no_itr=itr;
```

And the Regula-Falsi I am so, that is the three conditions are there we just copy.

(Refer Slide Time: 25:42)



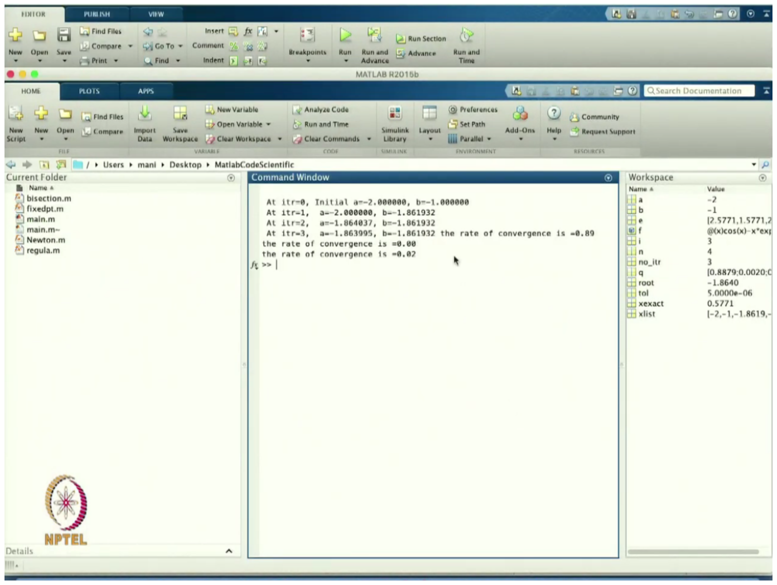
```
clear all
clc
%
% f=@(x) cos(x)-x*exp(x);
% a=-2; b=-1;
% tol=.5*10^(-5);
% [root, err, froot]=bisection(f,a,b,tol);
%---------------------------
% f=@(x) x^3-2*x-8; xexact=2.331;
% g= @(x) (2*x+8)^(1/3);
% dg=@(x) 1/(3*(2*x+8)^(2/3));
% a=1; tol=.5e-5;
% [root,froot,itr,xlist]=fixedpt(f,g,dg,a,tol);
% e=abs(xlist-xexact);
% q=zeros(length(e)-2,1);
% for n=2:length(e)-1
% q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
% end
% for i=1: length(q)
%     fprintf('the rate of convergence is =%4.2f \n',q(i));
% end
%---------------Newton method----
% f=@(x) x^3-8; xexact=2;
% df=@(x) 3*x^2;
% a=1; tol=.5e-5;
% [root,itr,xlist]=Newton(f,df,a,tol);
%------------Regula Falsi--------
%function [root,no_itr,xlist]=regula(f,a,b,tol)
f=@(x) cos(x)-x*exp(x); xexact=0.5771;
 a=-2; b=-1;
 tol=.5*10^(-5);
 [root, no_itr, xlist]=regula(f,a,b,tol);
e=abs(xlist-xexact);
q=zeros(length(e)-2,1);
for n=2:length(e)-1
q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
end
for i=1: length(q)
    fprintf('the rate of convergence is =%4.2f \n',q(i));
end
```

And here you can write.

```
%------------Regula Falsi--------

%function [root,no_itr,xlist]=regula(f,a,b,tol)

 f=@(x) cos(x)-x*exp(x); xexact=0.5771;

  a=0; b=1; tol=.5*10^(-5);

  [root, no_itr, xlist]=regula(f,a,b,tol);

 e=abs(xlist-xexact);

 q=zeros(length(e)-2,1);

 for n=2:length(e)-1

 q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));

 end

 for i=1: length(q)

 fprintf('the rate of convergence is =%4.2f \n',q(i));

% end
```

So, that is suppose I am taking is 0.5771. So that was the approximation, approximate root we were getting in the case of Bisection method. So let us check that whether we are able to get this result or not. So, I will run this one and let us see.

(Refer Slide Time: 27:05)

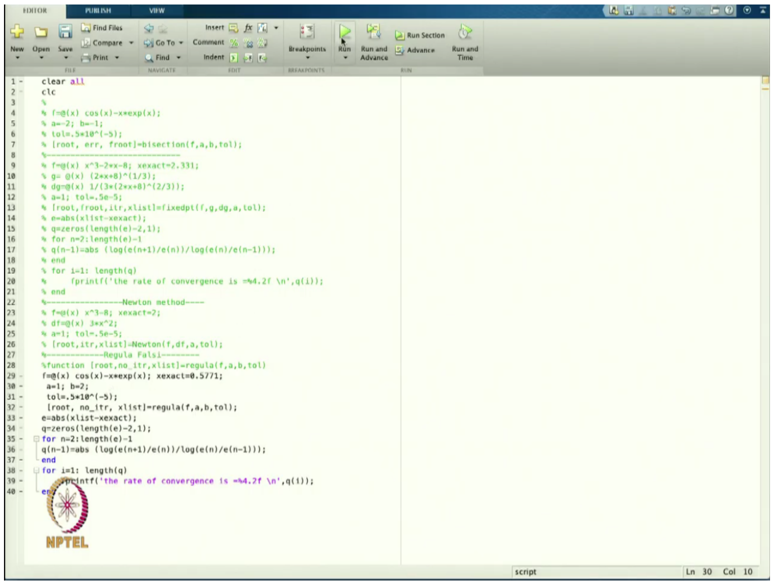


```
 At itr=0, Initial a=-2.000000, b=-1.000000
 At itr=1,  a=-2.000000, b=-1.861932
 At itr=2,  a=-1.864037, b=-1.861932
 At itr=3,  a=-1.863995, b=-1.861932 the rate of convergence is =0.89
the rate of convergence is =0.00
the rate of convergence is =0.02
>> 
```

Name	Value
a	-2
b	-1
e	[2.5771,1.5771,2
f	@(x)cos(x)-x*exp
i	3
n	4
no_itr	3
q	[0.8879;0.0020;0
root	-1.8640
tol	5.0000e-06
xexact	0.5771
xlist	[-2,-1,-1.8619,-

So that is a we are getting, so I started -2 and -1 and after that, okay so in this case I am getting this root -1.86. Okay?

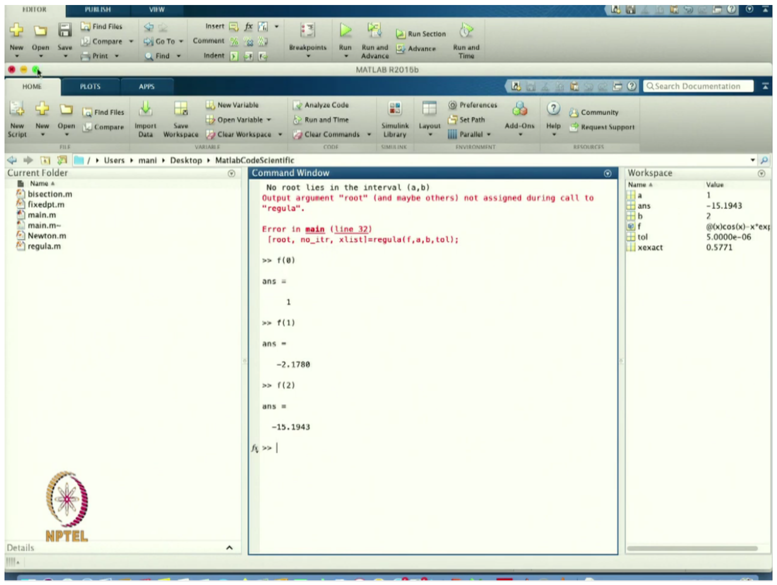
(Refer Slide Time: 27:22)



```
clear all
clc
%
% f=@(x) cos(x)-x*exp(x);
% a=-2; b=-1;
% tol=.5*10^(-5);
% [root, err, froot]=bisection(f,a,b,tol);
%---------------------------
% f=@(x) x^3-2*x-8; xexact=2.331;
% g= @(x) (2*x+8)^(1/3);
% dg=@(x) 1/(3*(2*x+8)^(2/3));
% a=1; tol=.5e-5;
% [root,froot,itr,xlist]=fixedpt(f,g,dg,a,tol);
% e=abs(xlist-xexact);
% q=zeros(length(e)-2,1);
% for n=2:length(e)-1
% q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
% end
% for i=1: length(q)
%     fprintf('the rate of convergence is =%4.2f \n',q(i));
% end
%---------------Newton method----
% f=@(x) x^3-8; xexact=2;
% df=@(x) 3*x^2;
% a=1; tol=.5e-5;
% [root,itr,xlist]=Newton(f,df,a,tol);
%-----------Regula Falsi--------
%function [root,no_itr,xlist]=regula(f,a,b,tol)
f=@(x) cos(x)-x*exp(x); xexact=0.5771;
 a=1; b=2;
 tol=.5*10^(-5);
 [root, no_itr, xlist]=regula(f,a,b,tol);
e=abs(xlist-xexact);
q=zeros(length(e)-2,1);
for n=2:length(e)-1
q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
end
for i=1: length(q)
    fprintf('the rate of convergence is =%4.2f \n',q(i));
end
```

That means I am doing let us see here, 1, 2. Let us see.

(Refer Slide Time: 27:38)

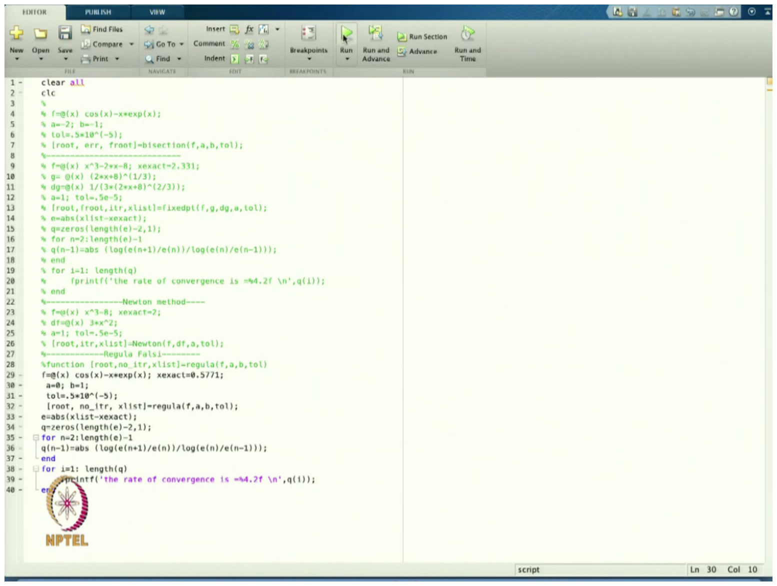


```
 No root lies in the interval (a,b)
Output argument "root" (and maybe others) not assigned during call to
"regula".

Error in main (line 32)
 [root, no_itr, xlist]=regula(f,a,b,tol);

>> f(0)

ans =

     1

>> f(1)

ans =

   -2.1780

>> f(2)

ans =

  -15.1943

>> |
```

```
Name      Value
a         1
ans       -15.1943
b         2
f         @(x)cos(x)-x*exp
tol       5.0000e-06
xexact    0.5771
```

What is my f? So I just put the value of f, 0. It is coming 1, f1, 2. Okay. So I have to choose 0 and 1.

(Refer Slide Time: 27:57)

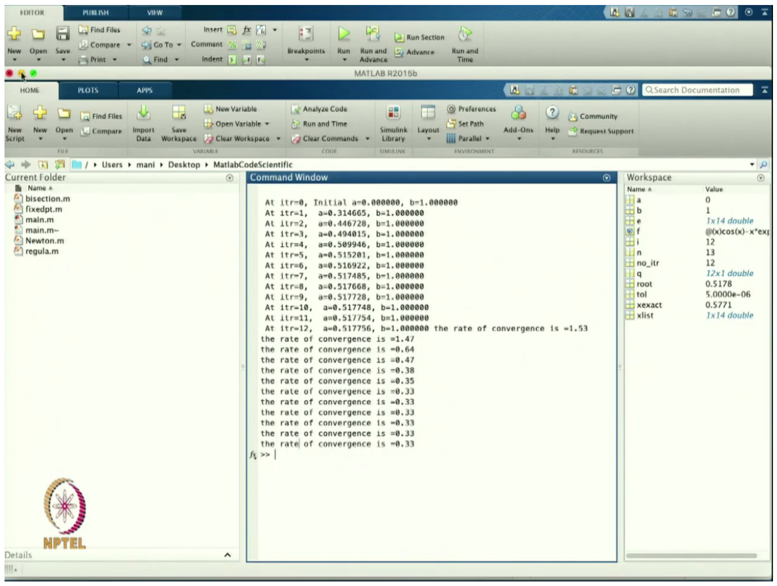


```
clear all
clc
%
% f=@(x) cos(x)-x*exp(x);
% a=-2; b=-1;
% tol=.5*10^(-5);
% [root, err, froot]=bisection(f,a,b,tol);
%---------------------------
% f=@(x) x^3-2*x-8; xexact=2.331;
% g= @(x) (2*x+8)^(1/3);
% dg=@(x) 1/(3*(2*x+8)^(2/3));
% a=1; tol=.5e-5;
% [root,froot,itr,xlist]=fixedpt(f,g,dg,a,tol);
% e=abs(xlist-xexact);
% q=zeros(length(e)-2,1);
% for n=2:length(e)-1
% q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
% end
% for i=1: length(q)
%     fprintf('the rate of convergence is =%4.2f \n',q(i));
% end
%----------------Newton method----
% f=@(x) x^3-8; xexact=2;
% df=@(x) 3*x^2;
% a=1; tol=.5e-5;
% [root,itr,xlist]=Newton(f,df,a,tol);
%-----------Regula Falsi--------
%function [root,no_itr,xlist]=regula(f,a,b,tol)
f=@(x) cos(x)-x*exp(x); xexact=0.5771;
 a=0; b=1;
 tol=.5*10^(-5);
 [root, no_itr, xlist]=regula(f,a,b,tol);
e=abs(xlist-xexact);
q=zeros(length(e)-2,1);
for n=2:length(e)-1
q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
end
for i=1: length(q)
    fprintf('the rate of convergence is =%4.2f \n',q(i));
en
```

```
script                                   Ln  30   Col  10
```

So let us choose 0 and 1. So let us choose this one.

(Refer Slide Time: 28:15)



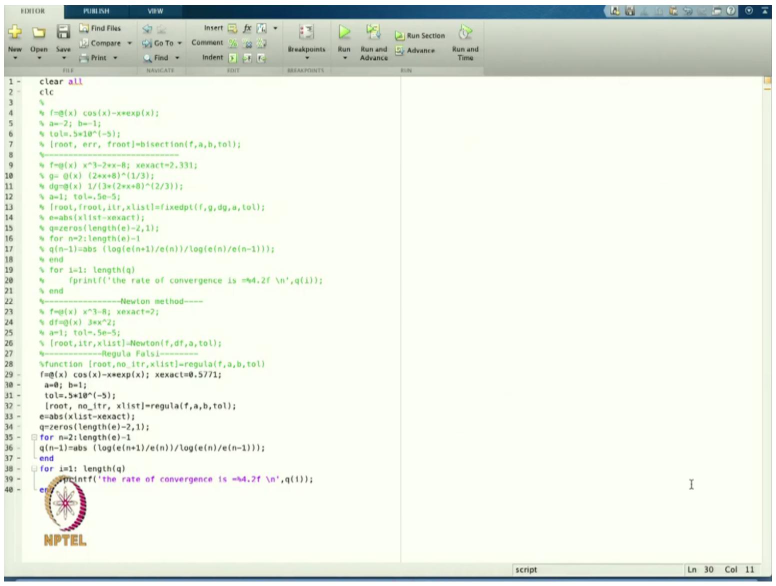
```
 At itr=0, Initial a=0.000000, b=1.000000
 At itr=1,  a=0.314665, b=1.000000
 At itr=2,  a=0.446728, b=1.000000
 At itr=3,  a=0.494015, b=1.000000
 At itr=4,  a=0.509946, b=1.000000
 At itr=5,  a=0.515201, b=1.000000
 At itr=6,  a=0.516922, b=1.000000
 At itr=7,  a=0.517485, b=1.000000
 At itr=8,  a=0.517668, b=1.000000
 At itr=9,  a=0.517728, b=1.000000
 At itr=10,  a=0.517748, b=1.000000
 At itr=11,  a=0.517754, b=1.000000
 At itr=12,  a=0.517756, b=1.000000 the rate of convergence is =1.53
the rate of convergence is =1.47
the rate of convergence is =0.64
the rate of convergence is =0.47
the rate of convergence is =0.38
the rate of convergence is =0.35
the rate of convergence is =0.33
the rate of convergence is =0.33
the rate of convergence is =0.33
the rate of convergence is =0.33
the rate of convergence is =0.33
the rate of convergence is =0.33
>> 
```

Name	Value
a	0
b	1
e	1x14 double
f	@(x)cos(x)-x*exp
i	12
n	13
no_itr	12
q	12x1 double
root	0.5178
tol	5.0000e-06
xexact	0.5771
xlist	1x14 double

So now look and see that, what are the results? So let us see. Yeah, so from here, you can see that I am getting my initial approximation 0 and 1. And with the time as the iteration passes, I am getting my solution 0.517756. So that is the approximate solution I am getting. And the rate of convergence started with 1.53 and then it was reduced to 0.333. So in this case, for these initial conditions, I am getting the, the rate of convergence is 0.33.

Because this is a Regula-Falsi method and we know that the Regula-Falsi method the rate of convergence is 1.61 or 1.62. So but that is the maximum rate of conventions we can have, but in this case based on my initial condition and I am finding for this rate of convergence 0.33 for this case what it is, it will be definitely less than 1.62.

(Refer Slide Time: 29:22)



```
clear all
clc
%
% f=@(x) cos(x)-x*exp(x);
% a=-2; b=-1;
% tol=.5*10^(-5);
% [root, err, froot]=bisection(f,a,b,tol);
%---------------------------
% f=@(x) x^3-2*x-8; xexact=2.331;
% g= @(x) (2*x+8)^(1/3);
% dg=@(x) 1/(3*(2*x+8)^(2/3));
% a=1; tol=.5e-5;
% [root,froot,itr,xlist]=fixedpt(f,g,dg,a,tol);
% e=abs(xlist-xexact);
% q=zeros(length(e)-2,1);
% for n=2:length(e)-1
% q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
% end
% for i=1: length(q)
%     fprintf('the rate of convergence is =%4.2f \n',q(i));
% end
%---------------Newton method----
% f=@(x) x^3-8; xexact=2;
% df=@(x) 3*x^2;
% a=1; tol=.5e-5;
% [root,itr,xlist]=Newton(f,df,a,tol);
%-----------Regula Falsi--------
%function [root,no_itr,xlist]=regula(f,a,b,tol)
f=@(x) cos(x)-x*exp(x); xexact=0.5771;
 a=0; b=1;
 tol=.5*10^(-5);
 [root, no_itr, xlist]=regula(f,a,b,tol);
e=abs(xlist-xexact);
q=zeros(length(e)-2,1);
for n=2:length(e)-1
q(n-1)=abs (log(e(n+1)/e(n))/log(e(n)/e(n-1)));
end
for i=1: length(q)
    printf('the rate of convergence is =%4.2f \n',q(i));
en
```

And in this case, I am already told you that for this type of equation, we do not know what is the value of xexact. So, here also I am choosing my approximate value of this one. So, if I choose the approximate value of the solution, then also it may happen then the rate of convergence will become lesser than the whatever expected. So, that is the case here we are getting, okay?

So, this is the way we have found the Regula-Falsi method and then based on the Regula-Falsi method and the Newton method, we have shown that how the rate of convergence can be applied or can be approximated within the same program whatever the program we are writing for the Regula-Falsi and the Newton-Raphson method. So maybe in the next lecture, we will continue for this one. So thanks for viewing this one, thanks very much.