

Scientific Computing Using Matlab
Professor Vivek Aggarwal
Indian Institute of Technology, Delhi
Professor Mani Mehra
Delhi Technological University
Department of Mathematics
Lecture 10

Bisection method for solving nonlinear equations

Welcome back to this course. We, today we are going to start with Lecture number 10. So, in the previous lecture we have started with scientific computing and how we can find the root of an equation. So, we will go further. So, in the previous class, we have discussed the intermediate value theorem.

(Refer Slide Time: 0:33)

Lecture-10
24 Feb 20

IVT $f(x) = \cos x - x e^x = 0$ on $\cos x$
 $x \in \mathbb{R}$
 $f(0) = \cos 0 - 0 = 1 > 0$
 $f(1) = \cos 1 - e = .540 - 2.71 < 0$
 $I_0 = (0, 1)$
 $f(0) f(1) < 0 \Rightarrow$ Use IVT a real root lies in $(0, 1)$.

Bisection Method:- This method is based on the repeated use of IVT.
 $f(x) = 0$ \Rightarrow (a, b)
 $I_0 = (a, b)$
bisection $I_0 \Rightarrow m_1 = \frac{a+b}{2}$
 $I_1 = (a, m_1)$ if $f(a) f(m_1) < 0$ $I_2 = m_2$
 $I_1 = (m_1, b)$ if $f(m_1) f(b) < 0$ $I_3 = m_3$
 $I_0 \supset I_1 \supset I_2 \supset I_3 \dots$

After 2 times process, we valid our app root in the sub interval $\frac{b_0 - a_0}{2^2}$ $a_0 = 0$ $b_0 = 5$

$I_1 = \frac{b_0 - a_0}{2}$
 $I_2 = \frac{b_0 - a_0}{2^2}$

Bisection method can be written as

$m_{k+1} = a_k + \frac{1}{2}(b_k - a_k) \quad k=0,1,2,\dots$

$(a_{k+1}, b_{k+1}) = \begin{cases} (a_k, m_{k+1}) & \text{if } f(a_k)f(m_{k+1}) < 0 \\ (m_{k+1}, b_k) & \text{if } f(m_{k+1})f(b_k) < 0 \end{cases}$

Iteration table:

iteration	a_{k-1}	b_{k-1}	$m_k = \frac{a_{k-1} + b_{k-1}}{2}$	$f(m_k)f(a_{k-1})$
1	0	1	$m_1 = \frac{0+1}{2} = 0.5$	$f(0.5)f(0) < 0$
2		m_1		
3				
4				
5				

$f(x) = \cos x - x e^x = 0$
 $f(0) = 1 > 0$
 $f(1) < 0$
 $a_0 = 0$
 $b_0 = 1$

$\cos(x) - 0.5 x e^{0.5}$
 $0.877 - (0.5) x e^{0.5}$

And then we have discussed that with the help of graphics, we can find out where the root lies. Now, we will take the help of the intermediate value theorem. And suppose I have an equation, so suppose I take a function $f(x) = \cos x - x e^x$. Suppose this function I take and I want to find the root.

So, in this case, I do not know where the root will lie. Either because that is now in this case it involves $\cos x$, x and e^x . So, one way is that, that I will plot function $\cos x$, I will plot function $x e^x$ and I will see where this intersects, so that will be the place we have the root.

Another one is that I want to take the help of the intermediate value theorem. So, in this case, so I take $f(x=0)$. So, that will be $\cos 0 - 0 = 1$. Now, I want to calculate what is $f(1)$. So, $f(1)$ means $\cos 1 - e$. And $\cos 1$ is basically if you see, suppose I have a calculator here, and I want 1 and then I put \cos of 1, so it gives the value 0.991. So that is almost equal to 1. But we know that the $\cos 1$, the value of the \cos is 1 only when x is equal to 0. So, in this case what is happening?

That is the problem that we have taken 1 as a degree. But I just told you that we cannot take 1 as a degree because whenever we deal with the x , that is a real number, we have to convert this into the radians. And now I can find the value of 1 and now \cos . So, from here you can see that $\cos r$, r means in the radians, the value is coming 0.54. So, it is coming 0.540. So, from here it can be written $0.540 - e$ and the value of the e is 2.71, so in this case this value will be less than 0, and this value is greater than 0.

So, from here I can say that $f(0) f(1) < 0$. So, I can, from here using intermediate, using intermediate value theorem we can say that a real root lies between 0 and 1. So, in this interval

the real root lies. So, to find out the roots, we have started with the first method and that is called the bisection method. So that is a very famous method, the bisection method. So, this method is based on the repeated use of intermediate value theorems.

So, suppose we have an equation $f(x) = 0$, so in this case we will find out the initial I_0 , so let us suppose this is equal to $[a, b]$. So, in this case, my $I_0 = [0, 1]$. So, this is my I_0 . Then I will do that, with the help of this one. So, I will bisect this one, I_0 . So, I will get my bisection, so I will call it m_1 . So that is $(a + b) / 2$. So, this is the bisection we got. And I will get another sub interval. So, this sub interval can be a m_1 if $f(a) f(m_1) < 0$.

So, it means the root is lying here, or it can be $[m_1, b]$. Because we know that $m_1 \in [a, b]$. So, it can be this also if $f(m_1) f(b) < 0$. So, now this I , I will get my sub interval that is called the I_m , I_1 . Similarly, now you can go for I_2 . So, I_2 will be another m_2 , I_3 It will give you another m_3 and so on. So, in this case, I will get that this is my I_0 , I started with this one. Then I will have I_1 , that will be, I_1 would be the subset of I_0 . Then I will get I_2 , then I will get I_3 , and this is the sequence of sub intervals I will get as we do the calculation.

So, using this one, I will get the sequence of sub intervals and after q times processes, process, we will get our approximate root. So, after this one I will get my approximate root in, in the sub interval I_q . So, the sub interval I_q will be basically, I start with $b_0 - a_0$ and its interval, the length will be this one, of I_q will be $(b_0 - a_0) / 2^q$, where $a_0 = a$ and $b_0 = b$. So, that is the starting point we take. And then we divide by this one, I will get, so my I_1 , I know that is $(b_0 - a_0) / 2$.

So, this is the first sub interval I got. Second interval suppose I got $(b_0 - a_0) / 2^2$. Again, I take the sub interval and so on. So in this way we are decreasing the length of our sub interval. So, if suppose I get the interval I_q in which I go to the root, so this is the length of the sub interval $(b_0 - a_0) / 2^q$. So, from here I can write down that my bisection method can be written as, as my $m_{k+1} = a_k + 1/2 (b_k - a_k)$ where $k = 0, 1, 2$ and so on.

$$(a_{k+1}, b_{k+1}) = (a_k, m_{k+1}) \text{ if } f(a_k) f(m_{k+1}) < 0$$

$$(a_{k+1}, b_{k+1}) = (m_{k+1}, b_k) \text{ if } f(b_k) f(m_{k+1}) < 0$$

And based on this value of the bisect, if it is using the intermediate value theorem, this can happen. And this is my sub interval, the new sub interval at the $(k+1)$ th step.

So, this is the way we can find the, this bisection method is given for the given interval $[a, b]$. Now, suppose how it works, so let us do the work of this one. So, in general what do we do? We make a table as given below for $f(x) = \cos(x) - x e^x$, $a_0 = 0$ and $b_0 = 1$.

k	a_{k-1}	b_{k-1}	$m_k = (a_{k-1} + b_{k-1})/2$	$f(m_k)f(a_{k-1})$
1	0	1	0.5	$f(0.5)f(0) > 0$
2	0.5	1	0.75	

This table can be used for the Bisection method.

So, from here, I can say that my $a_0 = 0$ and $b_0 = 1$. And the root will lie between 0 and 1. So I will start with here, so I will take first 0 and 1. Then with the help of this, I will get my $m_1 = 0.5$. So, based on this, I will find the value of $f(0.5)f(0)$. And $f(0)$ is 1 that is positive.

So, I will try to find the value of 0.5, then based on this one I will see that my m_1 will go here or here. So maybe we can find the value of this, because this is just I am using the calculator, so I will find out, so I am doing everything in the radians. So, this the value 0.5 I have to find. So, this is $0.5\cos$, so that is the value of 0.5.

So, this thing we can do in just one go by making a code in the matlab. So let us write the code in the matlab.

(Refer Slide Time: 14:00)

\Rightarrow we know that $Tol = 5 \times 10^{-4} = 0.0005$
 $I_n = \frac{b_n - a_n}{2^n} \leq Tol$
 $\log\left(\frac{b_n - a_n}{2^n}\right) \leq \log(Tol) \Rightarrow \frac{\log(b_n - a_n) - \log(Tol)}{\log(2)} \leq n$
 $\Rightarrow \frac{\log(b_n - a_n) - \log(Tol)}{\log(2)} \leq n$
Matlab Code

So, based on this first I will do that, that is how we can stop. So, based on this, now we will do that, how much is the permissible error in this case. Now we know that the b minus a naught is equal to 2 raise to power k or 2 raise to power n . B naught minus a naught 2 raise to power n .

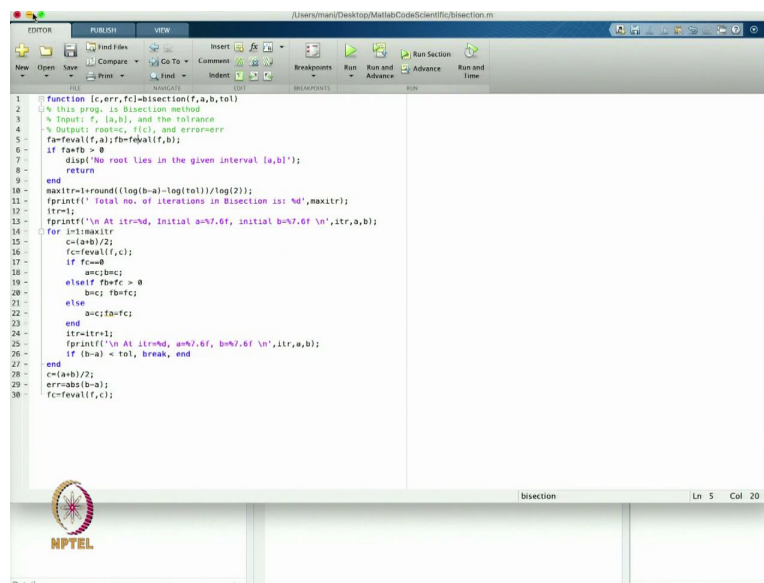
Suppose this is the nth iteration and I got my root. So, that suppose this is less than equal to some tolerance. So, this is the tolerance I have given.

It can be a tolerance, suppose I have given the tolerance 0.5 into 10 raise to power minus 4. So, that is point 00005. So, it is giving me the root that is correct up to four decimal places. So, in this case, I am finding my root that should be accurate up to four decimal places. So, suppose I take this tolerance and suppose this tolerance, based on this tolerance my iteration is stopped and I got my root.

$$n \geq \frac{\log(b_0 - a_0) - \log(tol)}{\log(2)}$$

So, based on this one, I can say that this scheme is basically how many number of iterations is needed to find the roots in my nth value, based on this tolerance. So, based on this one, I can say that how many number of iterations is going to take if I am going to use the bisection method to find out the root of the equation. So, let us start with the matlab code. So, we can directly go to the matlab. Yes, so after discussion with the bisection method, let us write the code for, bisection method.

(Refer Slide Time: 17:57)



```

1 function [c,err,fc]=bisection(f,a,b,tol)
2 % this prog. is Bisection method
3 % Input: f, [a,b], and the tolerance
4 % Output: root=c, f(c), and error=err
5 fa=fval(f,a);fb=fval(f,b);
6 if fa*fb > 0
7     disp('No root lies in the given interval [a,b]');
8     return
9 end
10 maxitr=ceil(log(b-a)-log(tol))/log(2));
11 fprintf(' Total no. of iterations in Bisection is: %d',maxitr);
12 itr=1;
13 fprintf('\n At itr=%d, Initial a=%f, Initial b=%f \n',itr,a,b);
14 for i=1:maxitr
15     c=(a+b)/2;
16     fc=fval(f,c);
17     if fc==0
18         a=c;b=c;
19     elseif fb*fc > 0
20         b=c; fb=fc;
21     else
22         a=c; fa=fc;
23     end
24     itr=itr+1;
25     fprintf('\n At itr=%d, a=%f, b=%f \n',itr,a,b);
26     if (b-a) < tol, break, end
27 end
28 c=(a+b)/2;
29 err=abs(b-a);
30 fc=fval(f,c);
  
```

```

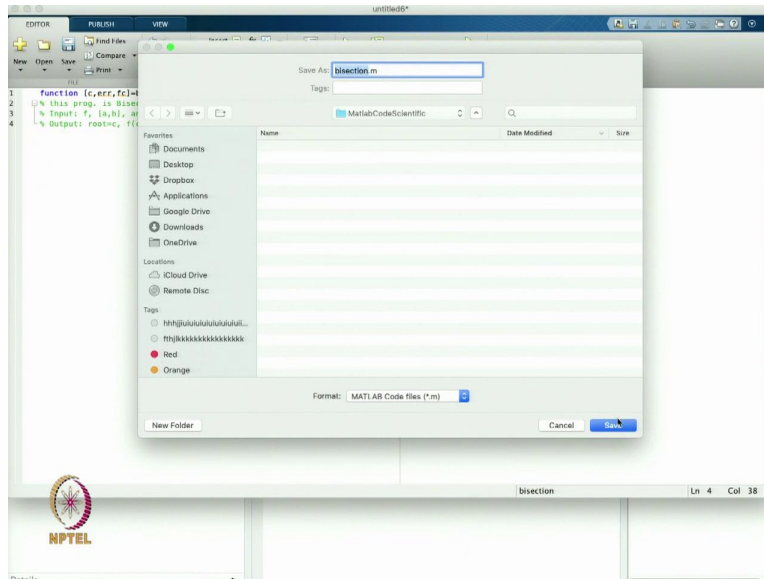
function [c,err,fc]=bisection(f,a,b,tol)
% this prog. is Bisection method
  
```

```

% Input: f, [a,b], and the tolerance
% 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);

```

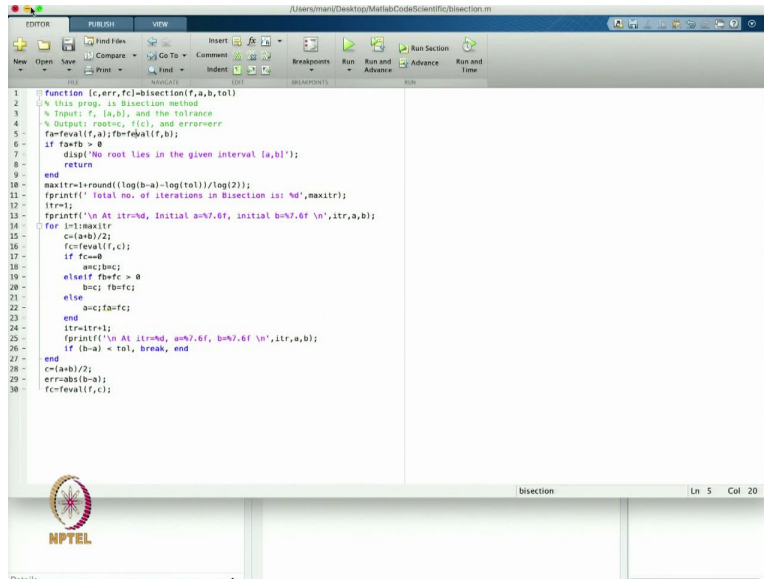
Now, what I want to call this program from some script file. So, now I have to start a new file.
(Refer Slide Time: 20:25)



So, this is the new file. In, in the new file we always want that the if the previous, any previous file is opened, that should be closed. So, what I will do is, I will start always by clear all. And clc. So, this will I have to write. Now, what I want to do is that suppose I want to find out, now I need a function f to start with. So, I will write the function f is equal to, I will write the anonymous function. So, this is the function I just defined.

And now based on this one, now I want to give the value of a and b . That which value of a and b should be given, so that I will get my solution. So, in this case, in the previous example, suppose I give start with value of a is equal to 0, I do not know where the value is this and I will give the value b is equal to 1. So, let us start with this one. And then I now call this one. So, I want root error and y root or f root. The value of f at root. So, this is the output I am going to get. Now, I will call my function bisection.

(Refer Slide Time: 20:37)

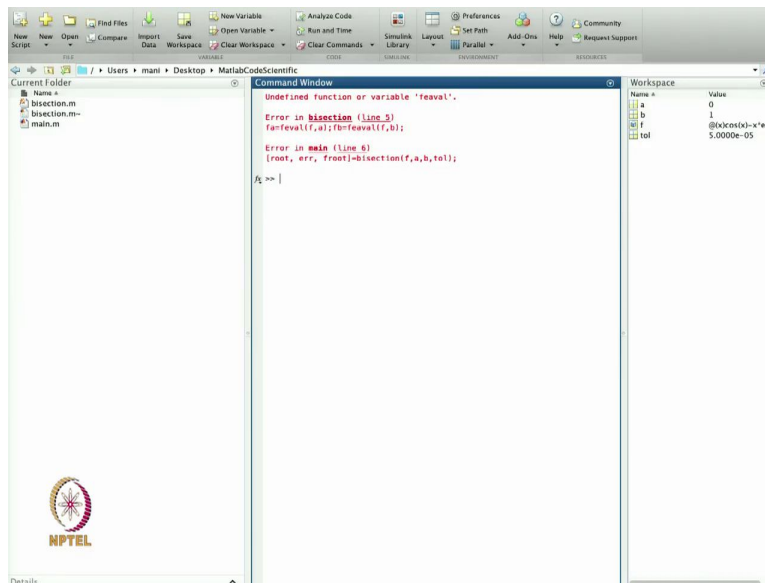
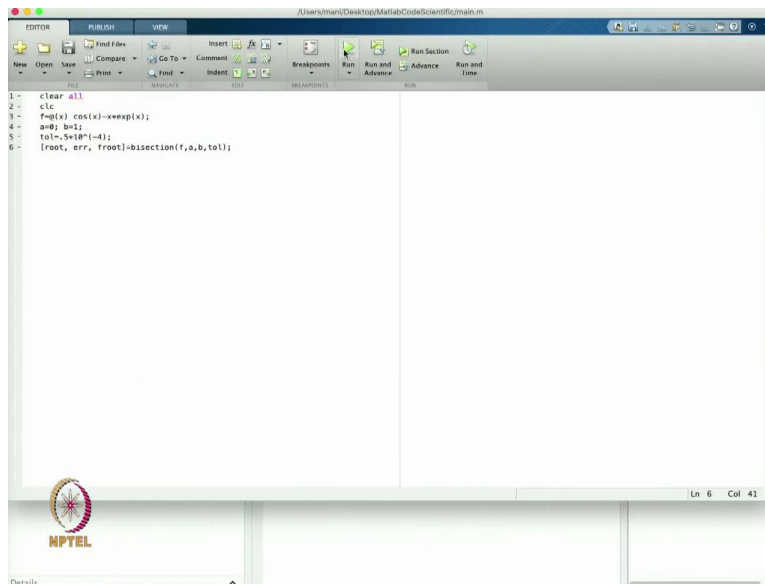


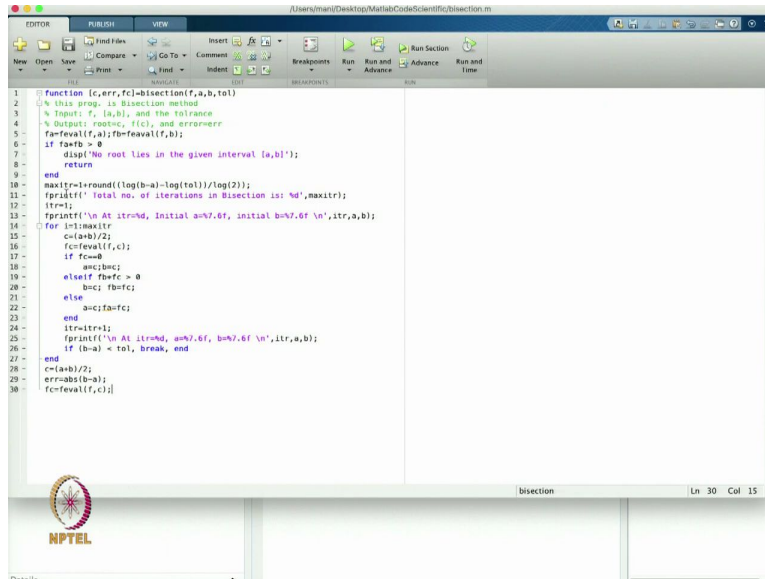
And then I will pass the value of f , then a , then b and then tolerance. So, this tolerance I have to define here. Now, what I do is that I will define my tolerance. So, tolerance I am defining, maybe 0.5×10^{-4} . So, this is the tolerance I am defining. That I should get my solution correct up to four decimals. So, I will save this function as the main function or calling function. So, I will write it as the main function and I will save it. Now, suppose I want to run this one, so let us run this one. So, it will give you that error.

```
clear all
clc
%
% f=@(x) cos(x)-x*exp(x);
% a=0; b=1;
% tol=.5*10^(-4);
% [root, err, froot]=bisection(f,a,b,tol);
```

I started with the total number of iteration is 15, at iteration 1 the initial value I have started with $a = 0$ and $b = 1$. Then what is happening? At iteration two, this will be $a = (0+1)/2$. So, $a = 0.5$, $b=1$. At iteration three, it will be this one. So, after doing all this iteration ultimately in the end, so we have a 16 iteration, but maximum iteration 15, because the first iteration I have just taken starting from the 1, because otherwise these are my initial iteration, so I should start with the 0 also. So, in this case, if I write this one, so and the, and the final end, I will get my solution that is 0.5177. So, that is my solution. After doing this 15 iterations.

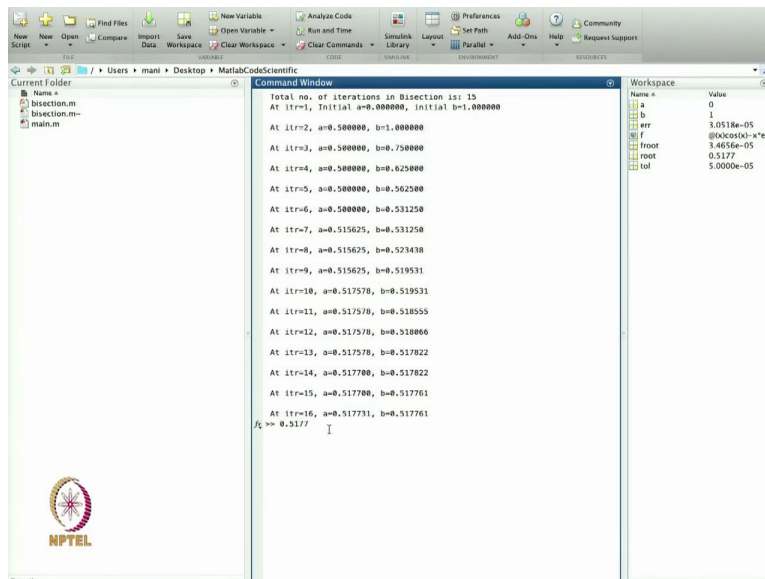
(Refer Slide Time: 30:52)





```

1 function [c,err,fc]=bisection(f,a,b,tol)
2 % this prog. is Bisection method
3 % Input: f, [a,b], and the tolerance
4 % Output: rootc, f(c), and errorerr
5 fa=feval(f,a);fb=feval(f,b);
6 if fa*fb > 0
7     disp('No root lies in the given interval [a,b]');
8     return
9 end
10 maxitr=round((log(b-a)-log(tol))/log(2));
11 fprintf('Total no. of iterations in Bisection is: %d',maxitr);
12 itr=1;
13 fprintf('\n At itr=%d, Initial a=%7.6f, initial b=%7.6f \n',itr,a,b);
14 for itr=1:maxitr
15     c=(a+b)/2;
16     fc=feval(f,c);
17     if fc==0
18         a=c;b=c;
19     elseif fb*fc > 0
20         b=c;fb=fc;
21     else
22         a=c;fa=fc;
23     end
24     itr=itr+1;
25     fprintf('\n At itr=%d, a=%7.6f, b=%7.6f \n',itr,a,b);
26     if (b-a) < tol, break, end
27 end
28 c=(a+b)/2;
29 err=abs(b-a);
30 fc=feval(f,c);
  
```



Command Window

```

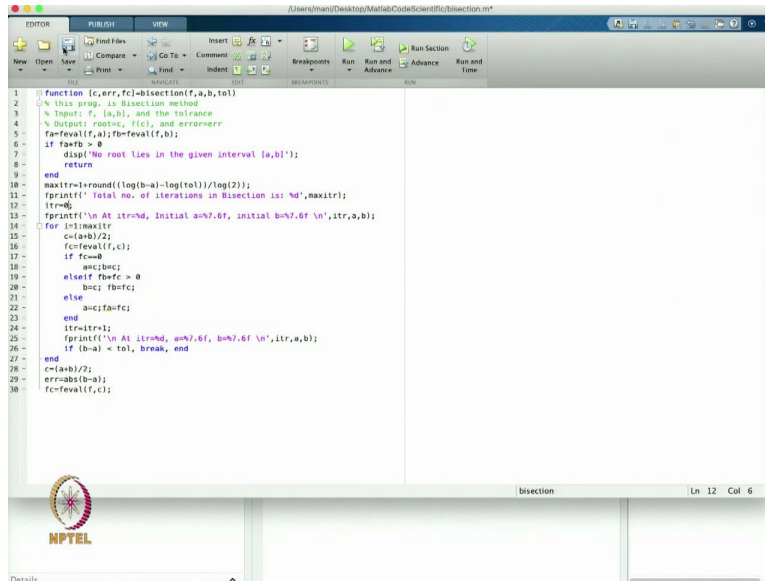
Total no. of iterations in Bisection is: 15
At itr=1, Initial a=0.000000, initial b=1.000000
At itr=2, a=0.500000, b=1.000000
At itr=3, a=0.500000, b=0.750000
At itr=4, a=0.500000, b=0.625000
At itr=5, a=0.500000, b=0.562500
At itr=6, a=0.500000, b=0.531250
At itr=7, a=0.515625, b=0.531250
At itr=8, a=0.515625, b=0.523438
At itr=9, a=0.515625, b=0.519531
At itr=10, a=0.517578, b=0.519531
At itr=11, a=0.517578, b=0.518555
At itr=12, a=0.517578, b=0.518066
At itr=13, a=0.517578, b=0.517822
At itr=14, a=0.517700, b=0.517822
At itr=15, a=0.517700, b=0.517761
At itr=16, a=0.517731, b=0.517761
f_c >> 0.5177
  
```

Workspace

Name	Value
a	0
b	1
err	5.0518e-05
f	@(x)cos(x)*exp(x)
froot	3.4656e-05
root	0.5177
tol	5.0000e-05

So, maybe I can change the number of iterations here. So this is my code. And I can start with 0. So, 0 means at iteration 0, means the initial value, I am giving this value and after this, it is calculating the value of this function. So, this is f, I am doing this one and now, I can call from here. So, now I can, maybe I can change my accuracy, so I will give the value tolerance value is equal to maybe 3. Because once I change the tolerance, the number of iterations will also change. And suppose I run the code, I will get, new value of this one. So, in this case, the maximum iterations are 12.

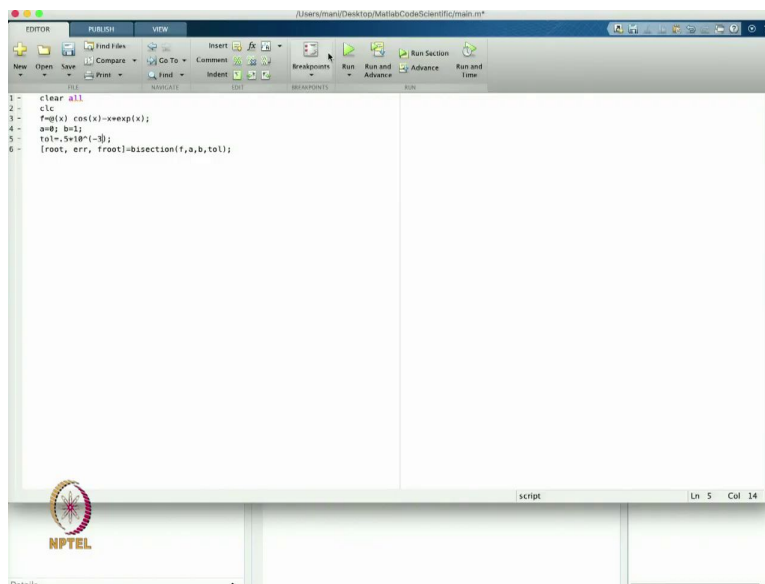
(Refer Slide Time: 35:10)



The image shows a MATLAB Editor window titled "bisection.m". The script defines a function `bisection(f,a,b,tol)` that implements the bisection method. It includes comments in Chinese and English, and uses `fprintf` for output. The script calculates the number of iterations and updates the interval `[a,b]` until the error is within the tolerance `tol`.

```
1 function [r,err,fc]=bisection(f,a,b,tol)
2 % this prog. is Bisection method
3 % Input: f, [a,b], and the tolerance
4 % Output: root r, f(c), and error err
5 fa=feval(f,a);fb=feval(f,b);
6 if fa*fb > 0
7     disp('No root lies in the given interval [a,b]');
8     return
9 end
10 maxitr=1+round((log(b-a)-log(tol))/log(2));
11 fprintf('Total no. of iterations in Bisection is: %d',maxitr);
12 itr=0;
13 fprintf('\n At itr=%d, initial a=%7.6f, initial b=%7.6f \n',itr,a,b);
14 for itr=1:maxitr
15     c=(a+b)/2;
16     fc=feval(f,c);
17     if fc==0
18         a=c;b=c;
19     elseif fb*fc > 0
20         b=c;fb=fc;
21     else
22         a=c;fa=fc;
23     end
24     itr=itr+1;
25     fprintf('\n At itr=%d, a=%7.6f, b=%7.6f \n',itr,a,b);
26     if (b-a) < tol, break, end
27 end
28 c=(a+b)/2;
29 err=abs(b-a);
30 fc=feval(f,c);
```

The status bar at the bottom indicates "Ln 12 Col 6".



The image shows a MATLAB Editor window titled "main.m". The script defines a function `main` that calls the `bisection` function. It includes comments in Chinese and English, and uses `fprintf` for output. The script calculates the number of iterations and updates the interval `[a,b]` until the error is within the tolerance `tol`.

```
1 clear all
2 clc
3 f=@(x) cos(x)-x*exp(x);
4 a=0; b=1;
5 tol=5e10*(-1);
6 [root, err, froot]=bisection(f,a,b,tol);
```

The status bar at the bottom indicates "Ln 5 Col 14".

The Command Window displays the following output:

```

Total no. of iterations in Bisection is: 12
At itr=0, Initial a=0.000000, Initial b=1.000000
At itr=1, a=0.500000, b=1.000000
At itr=2, a=0.500000, b=0.750000
At itr=3, a=0.500000, b=0.625000
At itr=4, a=0.500000, b=0.562500
At itr=5, a=0.500000, b=0.531250
At itr=6, a=0.515625, b=0.531250
At itr=7, a=0.515625, b=0.523438
At itr=8, a=0.515625, b=0.519531
At itr=9, a=0.517578, b=0.519531
At itr=10, a=0.517578, b=0.518055
At itr=11, a=0.517578, b=0.518066

```

The Workspace window shows the following variables:

Name	Value
a	0
b	1
err	4.8828e-04
f	@(x)cos(x)-x*exp(x)
froot	-1.9745e-04
root	0.5178
tol	5.0000e-04

And this starting from the 0. So, this is my initial approximation and after sometime you will get, you will see that that is my solution. So, at iteration 11, you will get a is equal to 0.175 and b is equal to 0.18066. So, in this case, it is not reaching up to the tolerance, but we have started our code with the number of iterations and the number of iterations I am taking always maximum iteration.

So, it will not run more than the maximum number of iterations. So, in this code, the maximum number of iterations I have defined. And then maximum iterations is not giving me the solution up to this tolerance.

(Refer Slide Time: 36:43)

The Editor window shows the following code:

```

1 - clear all
2 - clc
3 - f=@(x) cos(x)-x*exp(x);
4 - a=0; b=1;
5 - tol=5e10*(-5);
6 - [root, err, froot]=bisection(f,a,b,tol);

```

The status bar at the bottom indicates the file is named 'script' and the cursor is at Line 5, Column 14.

The Command Window displays the following output:

```

Total no. of iterations in Bisection is: 19
At itr=0, Initial a=0.000000, Initial b=1.000000
At itr=1, a=0.500000, b=1.000000
At itr=2, a=0.500000, b=0.750000
At itr=3, a=0.500000, b=0.625000
At itr=4, a=0.500000, b=0.562500
At itr=5, a=0.500000, b=0.531250
At itr=6, a=0.515625, b=0.531250
At itr=7, a=0.515625, b=0.523438
At itr=8, a=0.515625, b=0.519531
At itr=9, a=0.515758, b=0.519531
At itr=10, a=0.517578, b=0.518555
At itr=11, a=0.517578, b=0.518066
At itr=12, a=0.517578, b=0.517822
At itr=13, a=0.517700, b=0.517822
At itr=14, a=0.517700, b=0.517761
At itr=15, a=0.517731, b=0.517761
At itr=16, a=0.517746, b=0.517761
At itr=17, a=0.517754, b=0.517761
At itr=18, a=0.517754, b=0.517757

```

The Workspace window shows the following variables:

Name	Value
a	0
b	1
err	3.8147e-06
f	@(x)cos(x)-x*exp(x)
froot	5.6459e-06
root	0.5178
tol	5.0000e-06

Or maybe if I want to increase that I will take the 5 number of, the number of iterations will increase and then, so that is my solution now. So, in this case, the number of iterations becomes 19 and that is my initial conditions, initial approximations and with the time after iteration 18, my solution is this one. So that is the solution we are getting. So, 5.1, 0.5177, 0.5177. So, up to 4 digit or 5 digit is giving the accuracy. So that is I am going to get, now I will, somebody says that okay I want to change the initial approximation.
(Refer Slide Time: 37:30)

The Editor window shows the following code:

```

1 - clear all
2 - clc
3 - f=@(x) cos(x)-x*exp(x);
4 - a=0; b=1;
5 - tol=5e-6;
6 - [root, err, froot]=bisection(f,a,b,tol);

```

The screenshot shows the MATLAB environment. The Command Window displays the following output:

```

Total no. of iterations in Bisection is: 20
At itr=0, Initial a=-1.000000, Initial b=1.000000
At itr=1, a=0.000000, b=1.000000
At itr=2, a=0.500000, b=1.000000
At itr=3, a=0.500000, b=0.750000
At itr=4, a=0.500000, b=0.625000
At itr=5, a=0.500000, b=0.562500
At itr=6, a=0.500000, b=0.531250
At itr=7, a=0.515625, b=0.531250
At itr=8, a=0.515625, b=0.523438
At itr=9, a=0.515625, b=0.519531
At itr=10, a=0.517578, b=0.519531
At itr=11, a=0.517578, b=0.518555
At itr=12, a=0.517578, b=0.518066
At itr=13, a=0.517578, b=0.517822
At itr=14, a=0.517700, b=0.517822
At itr=15, a=0.517700, b=0.517761
At itr=16, a=0.517731, b=0.517761
At itr=17, a=0.517746, b=0.517761
At itr=18, a=0.517754, b=0.517761
At itr=19, a=0.517754, b=0.517757

```

The Workspace shows the following variables and their values:

Name	Value
a	-1
b	1
f	3.8147e-06
root	0.5178
tol	5.0000e-06

So, instead of a I want to start with maybe -1. And then b is equal to 1. So, let us see what will happen. So, in this case, if you just see, the iteration becomes 20 and it is starting from - 1 and the b is 1. So, in this case, after sometime, and in the end after the 20 iteration, at iteration 19 you will get this value. So, that is the value we are getting. Because in the code we have written that. This is the code. So, I have given you that for, if I is equal to iteration, maximum iteration it reaches then do, but in between I have also written that if b - a is less than tolerance, then use a stop.

So, in this case you see that the number of maximum iterations is 20, but after the 19th iteration, we go to the solution. Because here the value of b minus a is less than tolerance, so it will break from the loop and it will come out. So, that is why this code we have written like that. So, based on this one, I can find the value of, so this is and this is my workspace, a b f, so in this case I can find, because somebody wants to see that I do not know the value of f, where this value is coming.

So, I have defined my f and now suppose I want to find out what is the value, so maybe you can check f, you can try with values of f maybe at minus 2. So, that gives the value minus this. Then I can try here, so it gives the plus value. It gives the minus value then the plus value. 1, 2, so the one root is lying here between f2 and f1. So, this one I can check also by the code f0. So, f0 gives the, f at 0, so that is the root we can find from the inbuilt function and that is the root of this one. So, based on this one, I can just verify that whether my solution is giving, code is giving the solution, so I can find from there that this is my root.

Or maybe I can define the root near -1. So near -1, I can find another root and this is my root. So,

in this case, based on this one, the real value of this, I can find the roots based on that where this point is lying. So, because from here you can see that $f - 2$ is giving this value and $f - 1$ is giving this value. So, one root is also lying in between. So let us in the code, so let us write this one.

```
f=@(x) cos(x)-x*exp(x);  
% a=-2; b=-1;  
% tol=.5*10^(-5);
```

So, that is the main. So, in this case I will define maybe -2 and -1, and let us run this code. So that is the value up to the 19. So, the total number of iterations is 19. And after the 18 iterations, you can see that my root is coming and that is a is equal to minus 1.8639. So, this is another real root we can define, we can find with the help of this code. So, that is the initial code on the bisection method.

So, maybe today we have started with the basics, we have used our basic knowledge of matlab to write the first code, that is the bisection method. And using this bisection method, we can find the root of various equations, based on this one. So, in the next lectures we will start with, we will continue with this one. So, thanks for viewing, thanks very much.